

INGINERIA ȘI MANAGEMENTUL SISTEMELOR TEHNICE

Note de curs

Corina Bîrleanu
Mihai Dragomir

U.T.PRESS
Cluj-Napoca, 2026
ISBN 978-606-787-865-8



Corina BÎRLEANU

Mihai DRAGOMIR

INGINERIA ȘI MANAGEMENTUL SISTEMELOR TEHNICE

Note de curs



U.T. PRESS

Cluj-Napoca, 2026

ISBN 978-606-737-865-8



Editura U.T.PRESS
Str. Observatorului nr. 34
400775 Cluj-Napoca
Tel.: 0264-401.999
e-mail: utpress@biblio.utcluj.ro
www.utcluj.ro/editura

Recenzia: Prof.dr.ing. Bogdan Mocan

Conf.dr.ing. Mircea Fulea

Pregătire format electronic on-line: Gabriela Groza

Copyright © 2026 Editura U.T.PRESS

Reproducerea integrală sau parțială a textului sau ilustrațiilor din această carte este posibilă numai cu acordul prealabil scris al editurii U.T.PRESS.

ISBN 978-606-737-865-8

Cuvânt înainte

- Lucrarea „Ingineria și managementul sistemelor tehnice. Note de curs” se adresează studenților Universității Tehnice din Cluj-Napoca care studiază ingineria și managementul sistemelor tehnice în cadrul programelor de studii din domeniile ingineresti.
- Materialul urmează ciclul de viață al unui sistem tehnic: de la gândirea sistemică și managementul riscului, la analiza nevoilor, explorarea și definirea conceptului, arhitectura sistemului și, în final, dezvoltarea inginerescă, integrarea și testarea.
- Fiecare capitol cuprinde conceptele de bază, exemple aplicate, concluzii și teme de aplicare. Trei exemple traversează întreaga lucrare: automobilul modern, controlul traficului aerian și controlul accesului într-o clădire.
- Trimiterile bibliografice sunt indicate în text între paranteze drepte, de exemplu [1, cap. 3], și corespund listei de la finalul lucrării. Figurile preluate sau adaptate din alte lucrări au sursa indicată în legendă; cele fără mențiune sunt elaborate de autori. Pentru aprofundare se recomandă lucrările [3]–[6] și [12].

Cuprins

<u>Partea I. Fundamentele ingineriei sistemelor</u>	6	<u>Partea a II-a. Dezvoltarea conceptului și dezvoltarea ingineriască</u>	90
<u>1. De ce avem nevoie de ingineria sistemelor</u>	7	<u>8. Analiza nevoilor și analiza operațională</u>	91
<u>2. Ingineria sistemelor ca profesie</u>	20	<u>9. Analiza funcțională</u>	107
<u>3. Structura sistemelor complexe</u>	31	<u>10. Cerințe funcționale și validarea lor</u>	116
<u>4. Interfețe și interacțiuni</u>	41	<u>11. Explorarea conceptului de sistem</u>	129
<u>5. Managementul ingineriei sistemelor</u>	49	<u>12. Definirea conceptului de sistem</u>	142
<u>6. Managementul riscurilor: concepte fundamentale</u>	60	<u>13. Arhitectura sistemului</u>	152
<u>7. Identificarea, evaluarea și atenuarea riscurilor</u>	70	<u>14. Modelarea arhitecturii: SysML și diagrame</u>	163
<u>Recapitularea Părții I</u>	89	<u>15. Dezvoltarea ingineriască</u>	181
		<u>16. Integrarea, testarea și gestionarea modificărilor</u>	198
		<u>Sinteza lucrării</u>	208
		<u>Bibliografie</u>	212

PARTEA I

Fundamentele ingineriei sistemelor

Gândirea sistemică, profesia de inginer de sistem, structura sistemelor complexe, interfețele, managementul ingineriei sistemelor și managementul riscurilor

CAPITOLUL 1

De ce avem nevoie de ingineria sistemelor

Cuprinsul capitolului

1.1 Inginerie, sistem, ingineria sistemelor

1.2 Sistemul este mai mult decât suma componentelor

1.3 Sursele complexității

1.4 Rolul inginerului de sistem

1.5 De ce eșuează proiectele complexe

1.6 Ingineria sistemelor și managementul proiectului

1.7 Studiu de caz: automobilul și ținta UE 2035

Concluzii · Teme de aplicare

1.1 Inginerie, sistem, ingineria sistemelor

Pentru a înțelege rolul ingineriei sistemelor pornim de la trei definiții de lucru, formulate în literatura de referință [1, cap. 1].

Inginerie

Aplicarea principiilor științifice în scopuri practice: proiectarea, realizarea și operarea eficientă a structurilor, echipamentelor și sistemelor.

Sistem

Un ansamblu de componente interconectate care colaborează pentru atingerea unui obiectiv comun.

Ingineria sistemelor

Disciplina care ghidează dezvoltarea sistemelor prea complexe pentru a fi proiectate ca un singur element.

- Manualul INCOSE descrie ingineria sistemelor ca o abordare **transdisciplinară și integratoare**, care permite realizarea, utilizarea și retragerea cu succes a sistemelor ingineresti, folosind principii sistemice și metode științifice, tehnologice și de management [3].
- „A ghida” înseamnă a conduce, a coordona și a direcționa deciziile tehnice pe baza înțelegerii sistemului ca întreg; ingineria sistemelor asigură coerența dintre cerințe, soluții tehnice și rezultate.

1.2 Sistemul este mai mult decât suma componentelor

- Proprietățile unui sistem nu se regăsesc în niciuna dintre componentele sale luate separat; ele apar din **interacțiunile** dintre acestea. Același set de componente, conectat diferit, produce un alt sistem.
- De aceea ingineria sistemelor privește sistemul ca întreg și analizează simultan:
 - interacțiunile interne, dintre componente și subsisteme;
 - interacțiunile externe, cu alte sisteme și cu mediul de operare.
- Standardul ISO/IEC/IEEE 15288 definește sistemul ca o combinație de elemente care interacționează, organizate pentru a atinge unul sau mai multe scopuri declarate, și descrie procesele ciclului său de viață [2].

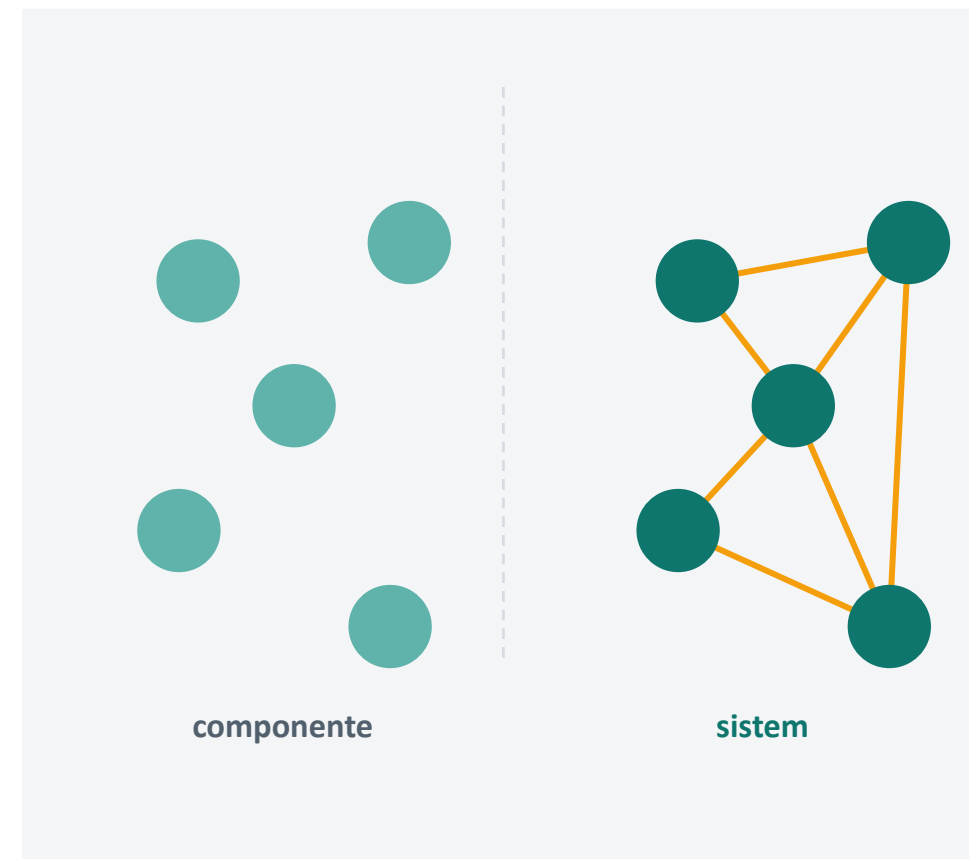


Fig. 1.1. Aceleași componente: colecție (stânga) și sistem (dreapta).

1.3 Sursele complexității

Ingineria sistemelor devine necesară atunci când sistemul este prea complex pentru a fi proiectat, construit sau produs ca un singur element [1, cap. 1]. Complexitatea provine din patru surse principale:

Număr mare de componente

Mii sau zeci de mii de elemente, imposibil de urmărit simultan.

Interdependențe

Modificarea unui element se propagă în elementele legate de el.

Tehnologii avansate

Soluții noi, cu comportament incomplet cunoscut și riscuri tehnice.

Constrângeri

Cost, termen, reglementări și cerințe de siguranță care trebuie echilibrate.

Sistemele complexe au comportament emergent și dificil de anticipat; interacțiunile dintre elemente contează adesea mai mult decât elementele însele [7]. De aceea complexitatea nu se „rezolvă” prin optimizarea fiecărei componente, ci prin gestionarea interacțiunilor.

Sistemul privit din exterior și din interior

Ingineria sistemelor nu se ocupă doar de proiectarea tehnică, ci și de factorii externi care influențează decisiv sistemul [1, cap. 1]. Sistemul se analizează din două direcții complementare:

Din exterior

- nevoile clienților și ale utilizatorilor
- mediul operațional
- interfețele cu alte sisteme
- suportul logistic și mentenanța
- pregătirea operatorilor

Din interior

- structura sistemului
- interacțiunile dintre componente
- integrarea subsistemelor

Toți acești factori trebuie să se regăsească în documentul de cerințe al sistemului, tratat în Partea a II-a.

1.4 Rolul inginerului de sistem

- Ingineria sistemelor face legătura dintre disciplinele tradiționale: mecanică, electrică, electronică, software, producție. Pentru ca sistemul să funcționeze, fiecare element trebuie să funcționeze corect **în combinație cu celelalte**, nu izolat; de aceea componentele nu pot fi proiectate independent unele de altele [1, cap. 1].
- Inginerul de sistem ghidează proiectarea fiecărui element astfel încât interacțiunile dintre subsisteme să fie compatibile și să se susțină reciproc.
- Rolul devine critic atunci când:
 - subsistemele sunt proiectate de echipe diferite;
 - componentele provin de la companii diferite;
 - cineva trebuie să răspundă de integrarea tehnică a întregului sistem.

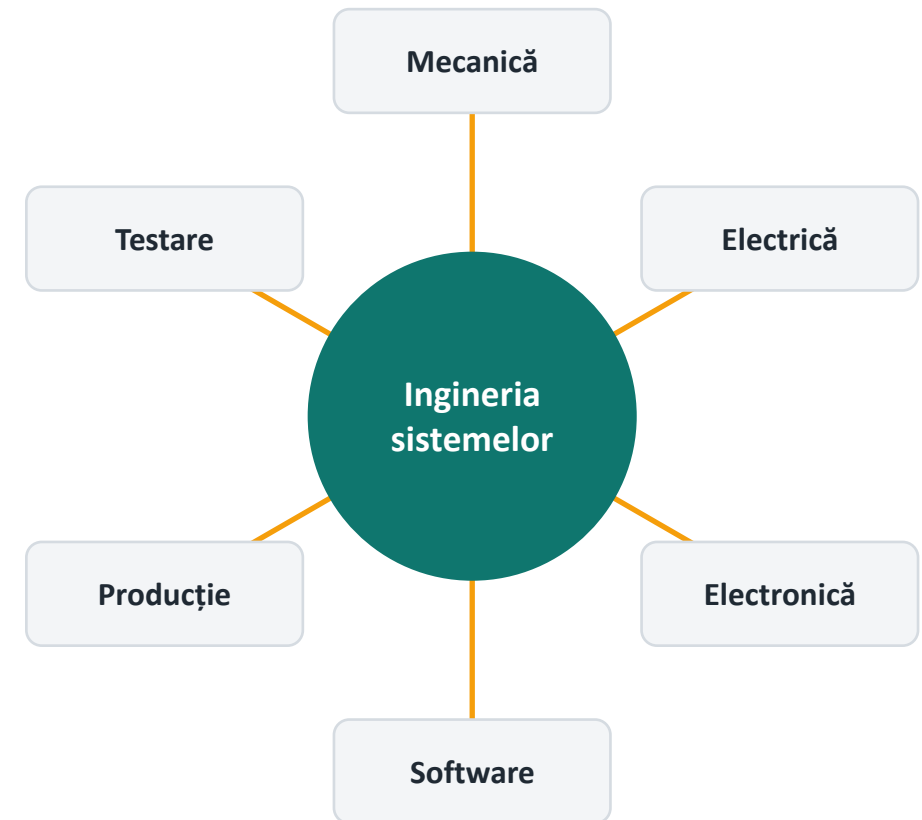


Fig. 1.2. Ingineria sistemelor leagă disciplinele tehnice.

1.5 De ce eșuează proiectele complexe

Experiența industrială arată că proiectele complexe eșuează rar din lipsă de tehnologie și mult mai des din **lipsă de integrare** [1, cap. 1], [3]. Cauzele tipice:

Cerințe neclare sau contradictorii

Decizii locale, fără viziune de ansamblu

Interfețe prost definite între subsisteme

Probleme descoperite târziu, la integrare

Exemple din industrie

Automotive	subsisteme proiectate separat, probleme la integrarea electronicii și a software-ului
Energie	echipamente performante, dar incompatibile cu infrastructura existentă
IT	aplicații livrate „la timp”, dar respinse de utilizatori
Producție	linii automatizate care funcționează local, dar nu ating capacitatea planificată

În toate cazurile, componentele funcționează, dar sistemul nu.

Ce aduce ingineria sistemelor

- 1 Viziune de ansamblu asupra sistemului
 - 2 Gestionarea cerințelor și a compromisurilor
 - 3 Reducerea riscurilor tehnice
 - 4 Integrarea coerentă a subsistemelor
 - 5 Performanță pe tot ciclul de viață
- Puterea ingineriei sistemelor constă în capacitatea de a privi dincolo de componente și de a echilibra cerințe concurente pe întregul ciclu de viață al sistemului [1, cap. 1].
 - Lipsa acestei perspective duce la costuri ridicate de operare, mentenanță dificilă și riscuri crescute în exploatare.

Optimizarea fiecărei componente nu garantează performanța sistemului.

1.6 Ingineria sistemelor și managementul proiectului

Ingineria sistemelor: CE construim și CUM funcționează

- definește obiectivele tehnice
- ghidează proiectarea și integrarea
- evaluează performanța tehnică

Managementul proiectului: CÂND livrăm și CÂT costă

- planificarea activităților
- bugetul și contractele
- relația cu clientul

Cine ce face într-un proiect tehnic

Stakeholderii

definesc obiectivele și criteriile de succes

Managerul de proiect

planifică; gestionează riscuri, costuri, constrângeri

Inginerii specialiști

proiectează și implementează subsistemele

Inginerul de sistem

coordonează ansamblul și asigură integrarea

Un proiect poate fi livrat la timp și în buget (managementul proiectului a reușit), dar sistemul să nu funcționeze corect (ingineria sistemelor a lipsit). Succesul apare doar prin integrarea celor două funcții; relația lor este detaliată în Capitolul 5.

1.7 Firul roșu al lucrării: automobilul modern

- Automobilul este exemplul care traversează întreaga lucrare. Este un singur produs care este, simultan, **mecanic, electric, electronic, software și strict reglementat**.
- Performanța lui depinde de integrarea precisă a componentelor și de echilibrul dintre cost, siguranță, emisii și confort.
- Chiar dacă principiile de bază sunt cunoscute de mult timp, tehnologiile sunt avansate, compromisurile sunt complexe, iar deciziile sunt de tip sistemic. Industria auto a adoptat explicit principiile ingineriei sistemelor în dezvoltarea grupurilor de propulsie [8].



Fig. 1.3. Automobilul: un sistem mecanic, electric, electronic și software. Sursa: [https://www.boomplay.com/buzz/3265000].

Studiu de caz: ținta UE 2035 pentru autoturisme

- Regulamentul (UE) 2023/851 a stabilit pentru autoturismele noi o reducere cu 100% a emisiilor de CO₂ începând din 2035 [9]. În decembrie 2025, Comisia Europeană a propus relaxarea țintei la 90%, propunere aflată în procedura legislativă [10].
- Indiferent de forma finală, decizia este o **problemă sistemică**, nu doar tehnologică. Soluțiile (vehicule electrice cu baterie, pile de combustie cu hidrogen, carburanți sintetici) diferă prin eficiența energetică, infrastructură, cost și impact ecologic.
- Cu energia produsă de aceeași turbină eoliană pot fi alimentate aproximativ 1600 de vehicule electrice, 600 cu hidrogen sau 250 cu carburanți sintetici [11].
- Trebuie echilibrate: reducerea emisiilor, eficiența energetică, fezabilitatea economică și infrastructura disponibilă.

Nu există un răspuns „corect”, ci un compromis argumentat. Aceasta este esența ingineriei sistemelor.

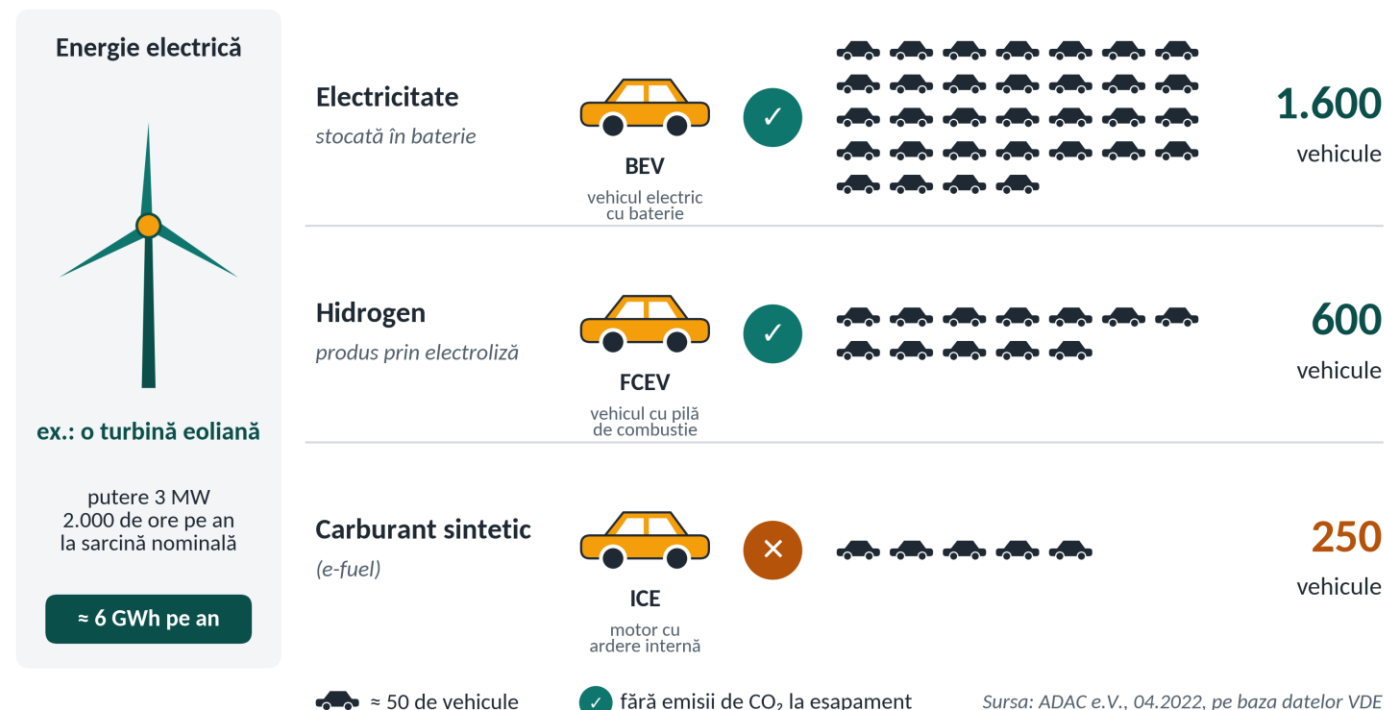


Fig. 1.4. Câte vehicule pot fi alimentate cu energia unei turbine eoliene. Sursa: adaptat după ADAC e.V., date VDE [11].

Concluzii

- 1 Un sistem nu este suma componentelor sale: proprietățile lui apar din interacțiuni.
- 2 Proiectele complexe eșuează mai ales din lipsă de integrare, nu din lipsă de tehnologie.
- 3 Ingineria sistemelor răspunde la întrebările CE și CUM; managementul proiectului, la CÂND și CÂT.
- 4 Deciziile reale, precum ținta UE 2035, sunt compromisuri sistemice.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Explicați într-un paragraf afirmația „Ingineria sistemelor se concentrează asupra sistemului în ansamblu”.

Tema 2

Ce caracteristici ale unui avion aparțin sistemului ca întreg și nu se regăsesc în niciunul dintre elementele lui?

Tema 3

Formulați două argumente pro și două contra pentru folosirea celor mai noi tehnologii într-un sistem complex, cu exemple.

CAPITOLUL 2

Ingineria sistemelor ca profesie

Cuprinsul capitolului

2.1 O profesie recunoscută

2.2 Organizarea unui proiect complex

2.3 Inginerul șef de sistem (CSE)

2.4 Parcursul profesional

2.5 Orientarea profesională: știință, matematică, inginerie

2.6 Modelul „T” al inginerului de sistem

2.7 Exemplu: deciziile CSE într-un proiect auto

Concluzii · Teme de aplicare

2.1 O profesie recunoscută

- Ingineria sistemelor este astăzi o profesie de sine stătătoare, esențială în proiectele tehnice complexe [1, cap. 1].
- Rolul apare din nevoia de coordonare interdisciplinară, de integrare tehnică și de decizii luate la nivelul întregului sistem.
- Inginerul de sistem **nu înlocuiește specialiștii**, ci îi coordonează spre un sistem funcțional. În Capitolul 1 am văzut de ce este nevoie de ingineria sistemelor; în acest capitol vedem cine o practică și cum se ajunge la acest rol.

Coordonare interdisciplinară

Integrare tehnică

Decizii la nivel de sistem

2.2 Organizarea unui proiect complex

Într-un proiect mare, echipele tehnice raportează inginerului șef de sistem (CSE, *Chief Systems Engineer*), iar acesta raportează managerului de proiect (PM). Controlul proiectului acoperă planificarea, costurile, personalul și relația contractuală [1, cap. 5].

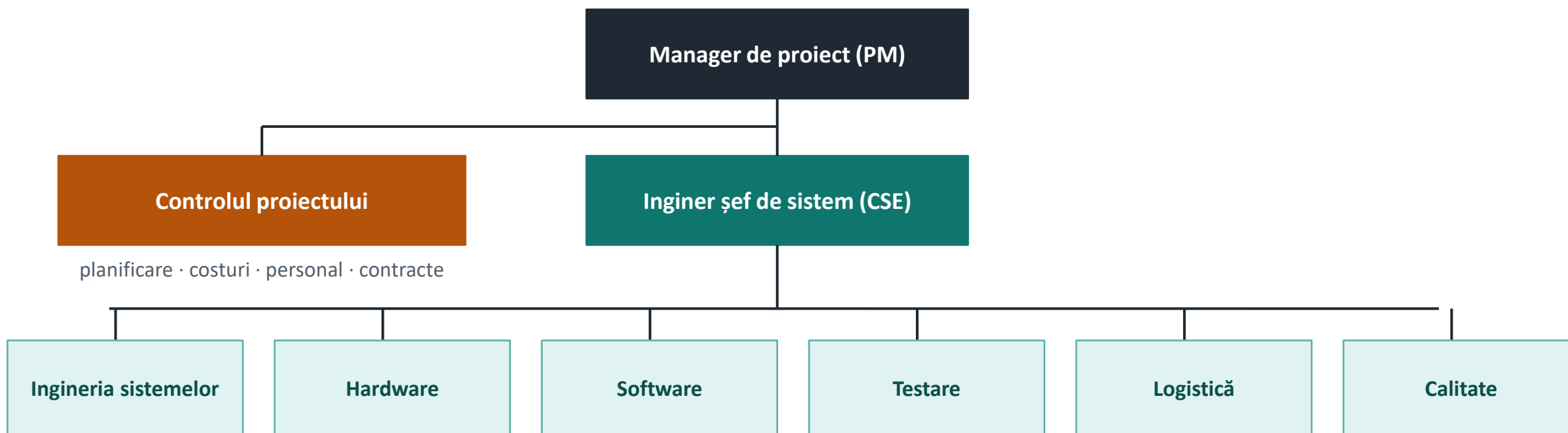


Fig. 2.1. Organizarea ilustrativă a unui proiect complex. Sursa: adaptat după [1, cap. 5].

2.3 Inginerul șef de sistem (CSE)

Managerul de proiect

- planificare și buget
- relația cu clientul

Inginerul șef de sistem

- răspunde de toate aspectele tehnice
- coordonează subsistemele
- conduce echipele de inginerie

În proiectele mari, CSE este partenerul tehnic al managerului de proiect: **interfața dintre tehnic și management** [1, cap. 1]. El este, în același timp, inginer de sistem, lider tehnic și coordonator al subsistemelor. Timpul său se împarte, orientativ, astfel:



Obiectivul principal al CSE este soluția tehnică corectă pentru client, nu optimizarea locală.

2.4 Parcursul profesional



- Ingineria sistemelor nu este un rol de început de carieră, ci unul de **maturitate profesională** [1, cap. 1]. Parcursul tipic începe cu specializarea într-un domeniu tehnic și continuă prin acumularea de experiență practică (OJT, *on-the-job training*).
- Pe măsură ce experiența crește, inginerul ajunge să coordoneze echipe și, în final, sisteme complexe. De aceea, de regulă, nu devii inginer de sistem imediat după facultate.

Harta carierei în ingineria sistemelor

- Ingineria sistemelor se află la intersecția a trei domenii: **tehnic, managerial și financiar** [1, cap. 1].
- De aici se deschid patru direcții de carieră:
 - spre **specializare tehnică**: director tehnic (CTO);
 - spre **zona financiară**: director financiar (CFO);
 - spre **management tehnic**: manager de program;
 - spre **conducerea sistemului**: inginer șef de sistem (CSE).
- Domeniul este atractiv tocmai pentru că păstrează deschise toate aceste trasee.

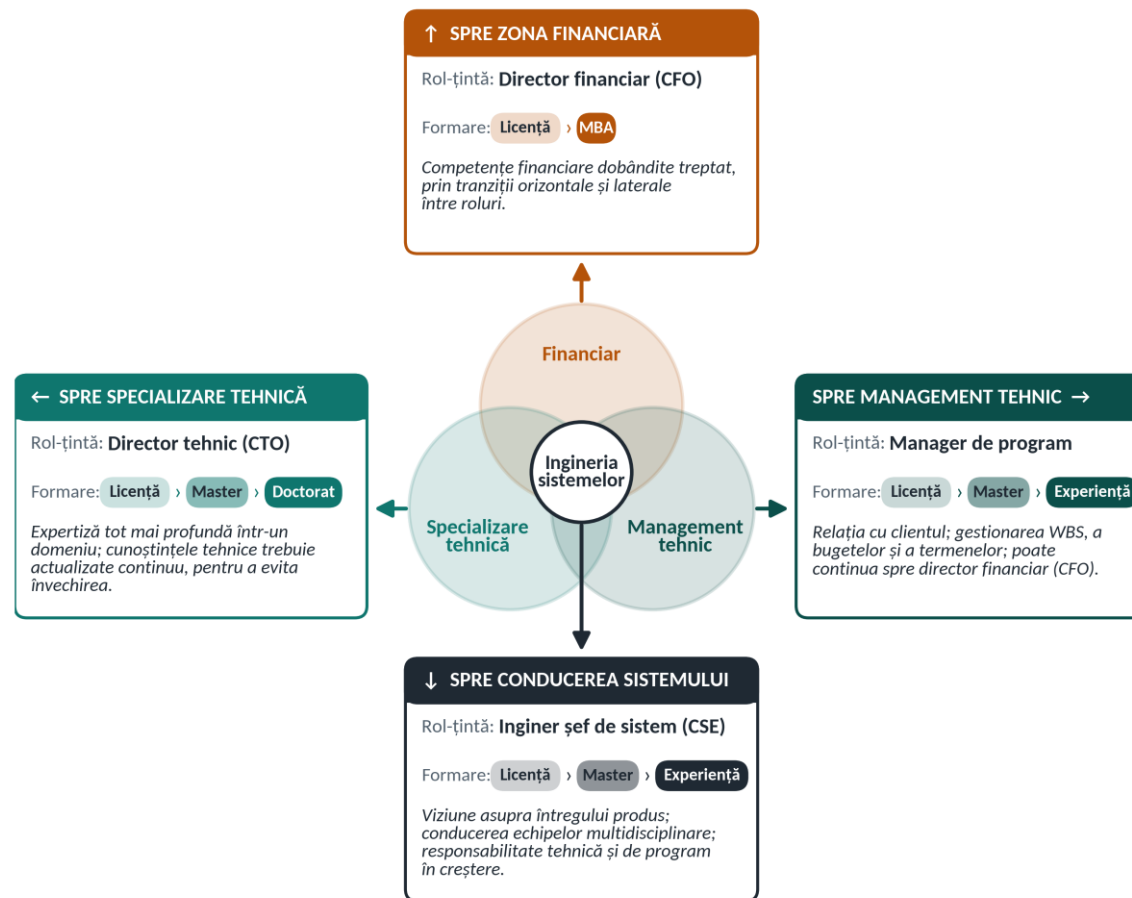


Fig. 2.2. Oportunități de carieră și creștere profesională. Sursa: adaptat după [1, cap. 1].

2.5 Orientarea profesională: știință, matematică, inginerie

- Profesioniștii tehnici gândesc diferit [1, cap. 1]:
 - **omul de știință** caută DE CE și CUM funcționează lumea;
 - **matematicianul** caută relații logice: „dacă A, atunci B”;
 - **inginerul** creează un produs care funcționează.
- Fiecare profesionist este o combinație a celor trei orientări, reprezentabilă ca un punct într-un triunghi (de exemplu, 70% știință, 20% matematică, 10% inginerie).
- Orientarea înseamnă interes și mod de gândire, nu cunoștințe. Inginerul de sistem combină cele trei perspective și privește sistemul ca întreg.

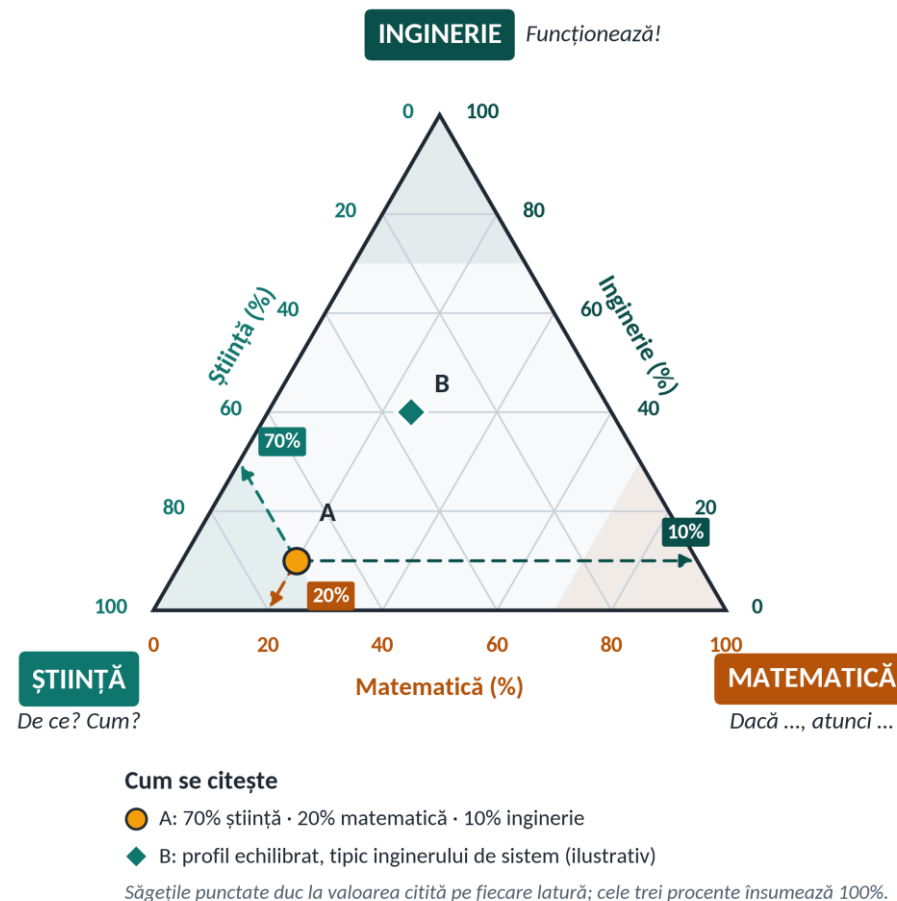


Fig. 2.3. Orientarea profesioniștilor tehnici. Sursa: adaptat după [1, cap. 1].

2.6 Modelul „T” al inginerului de sistem

- Inginerul de sistem are **profunzime** într-un domeniu tehnic (bara verticală a literei T) și **lărgime** în domeniile conexe (bara orizontală) [1, cap. 1].
- Treptele carierei, orientativ:
 - contributor tehnic: 1–4 ani;
 - membru în echipă: 5–8 ani;
 - inginer de sistem: 9–12 ani;
 - inginer de sistem senior: 13–20 de ani;
 - lider în ingineria sistemelor: peste 20 de ani.
- Succesul se măsoară prin performanța echipei și a sistemului livrat.

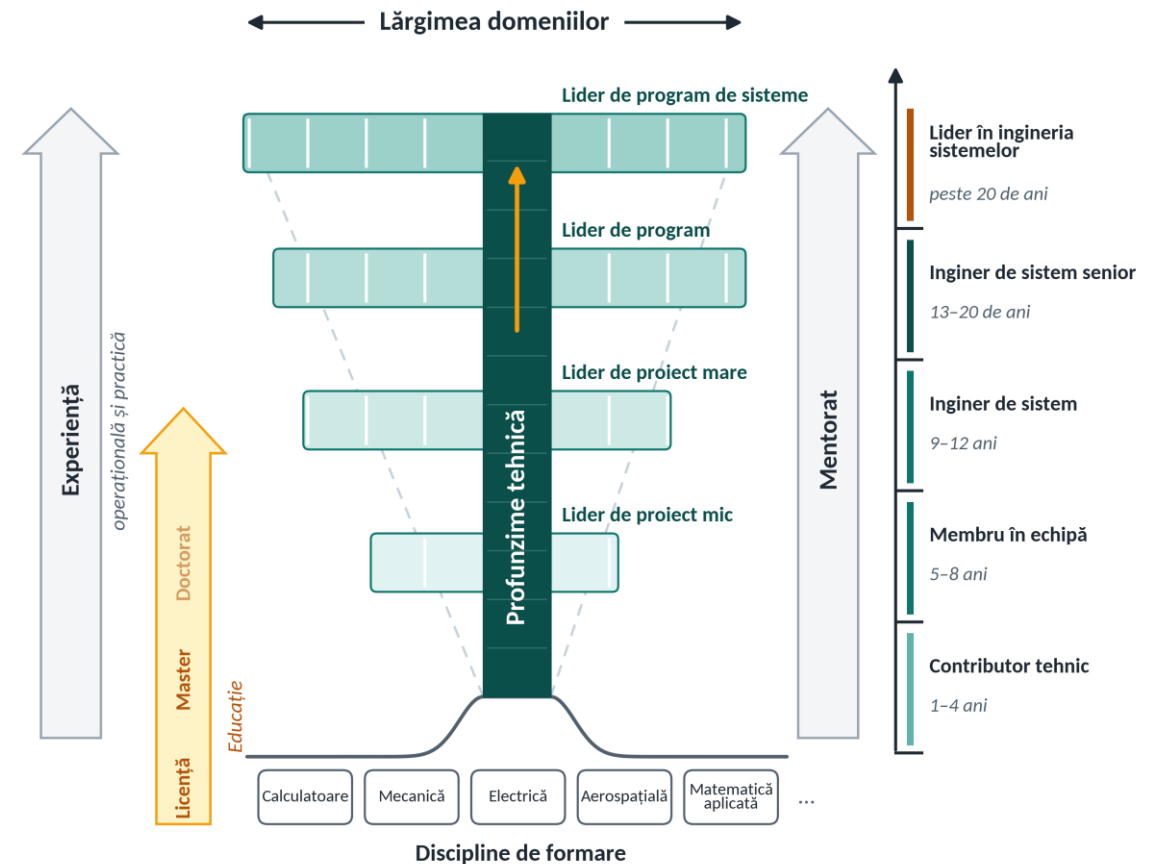


Fig. 2.4. Modelul de dezvoltare a carierei în ingineria sistemelor. Sursa: adaptat după [1, cap. 1].

2.7 Exemplu: deciziile CSE într-un proiect auto

Specialiștii optimizează fiecare componentă; inginerul șef de sistem cântărește efectul fiecărei decizii asupra întregului:

Baterie mai mare

- + autonomie mai mare
- masă și cost mai mari

Mai mulți senzori

- + siguranță sporită
- software mai complex

Materiale mai ușoare

- + consum mai mic
- cost de producție mai mare

Niciuna dintre aceste decizii nu poate fi luată corect privind o singură componentă. Exemplul anticipează analiza compromisurilor din explorarea conceptului (Partea a II-a).

Concluzii

1

Inginerul de sistem coordonează specialiștii, nu îi înlocuiește.

2

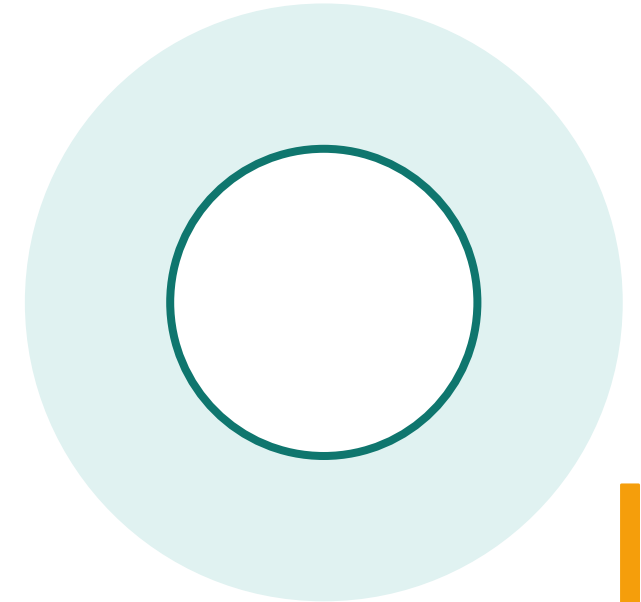
CSE răspunde de partea tehnică; managerul de proiect, de plan, buget și relația cu clientul.

3

Ingineria sistemelor este un rol de maturitate profesională, construit prin experiență.

4

Modelul „T”: profunzime într-un domeniu și lărgime în domeniile conexe.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Descrieți-vă orientarea profesională (x% știință, y% matematică, z% inginerie). Cum era la absolvire și de ce s-a schimbat?

Tema 2

Alegeți un proiect pe care îl cunoașteți. Cine a avut rolul de CSE sau cine ar fi trebuit să îl aibă?

CAPITOLUL 3

Structura sistemelor complexe

Cuprinsul capitolului

3.1 Organizarea ierarhică a sistemelor

3.2 Sistem sau subsistem: rolul nivelului de analiză

3.3 Inginerul de sistem și specialistul în proiectare

3.4 Structura funcțională și structura fizică

3.5 Limitele sistemului și mediul

3.6 Diagrama de context

Concluzii · Teme de aplicare

3.1 Organizarea ierarhică a sistemelor

- Un automobil are zeci de mii de piese. Nimeni nu le poate gândi simultan; ele se organizează pe **niveluri ierarhice** [1, cap. 3].
- Fiecare sistem este format din subsisteme, acestea din componente, iar componentele se descompun în subcomponente și piese.
- Ierarhia ajută:
 - la proiectare: fiecare nivel se proiectează și se verifică separat;
 - la alocarea responsabilităților: fiecare echipă știe ce nivel îi aparține;
 - la reducerea complexității: se lucrează pe un nivel, fără a le ține pe toate în minte.

Ierarhia este principalul instrument de control al complexității.

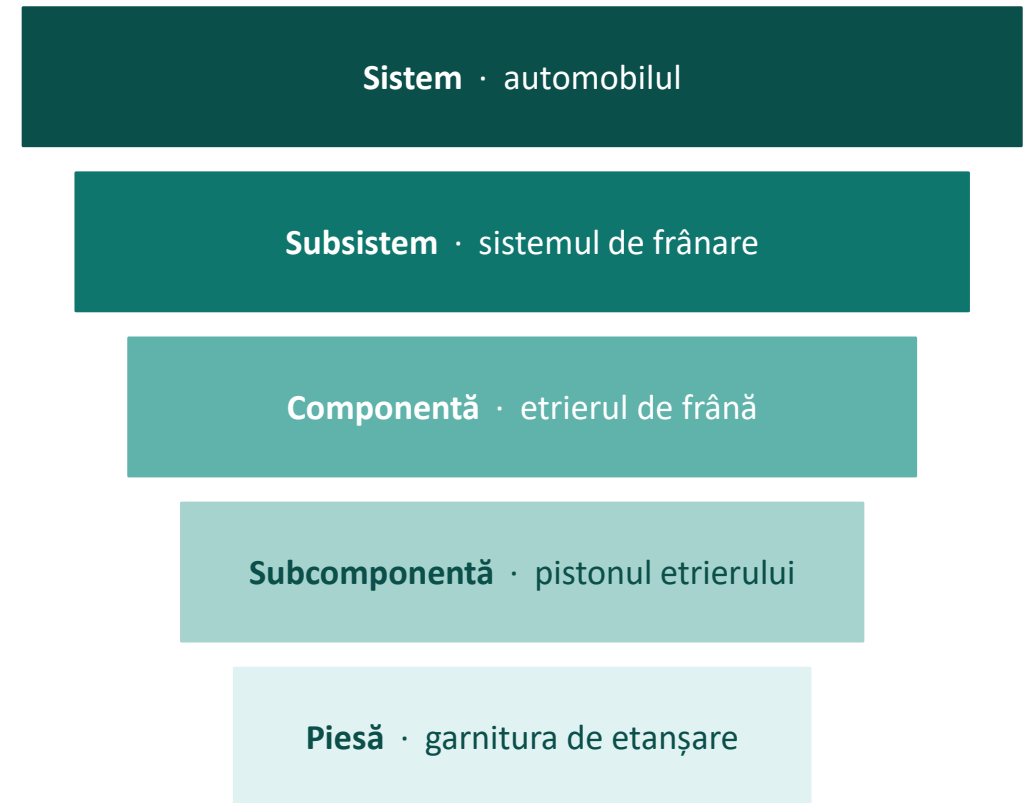


Fig. 3.1. Nivelurile ierarhice ale unui sistem, exemplificate pe automobil.

3.2 Sistem sau subsistem: rolul nivelului de analiză

- Aceeași entitate poate fi **sistem** la un nivel de analiză și **subsistem** la nivelul superior [1, cap. 3].
- Automobilul este un sistem, dar și un subsistem al transportului rutier. La fel, avionul comercial este un sistem, dar și un subsistem al transportului aerian.
- Aceleași cinci niveluri apar în domenii foarte diferite: sisteme de comunicații, sisteme informatice, prelucrarea materialelor, sisteme aerospațiale [1, cap. 3].
- Consecința practică: înainte de orice analiză trebuie stabilit **nivelul** la care lucrăm.

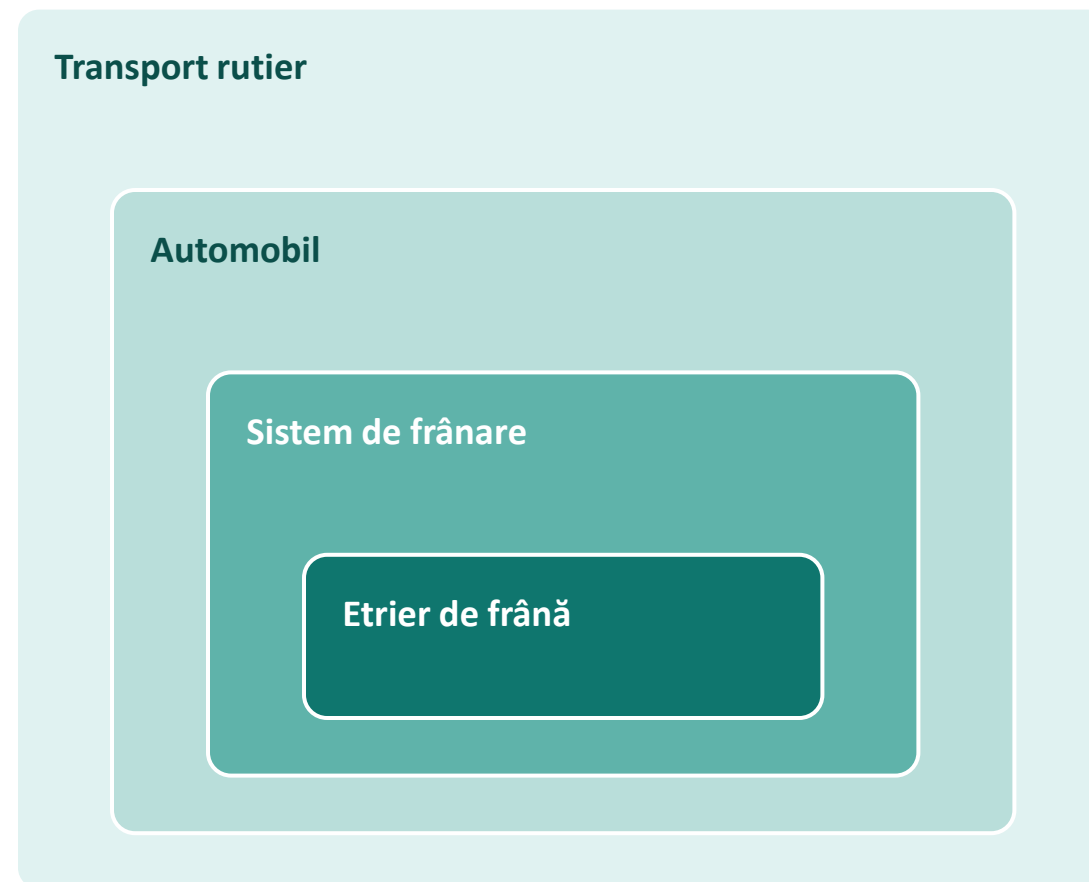


Fig. 3.2. Sisteme incluse unele în altele: nivelul de analiză decide ce este „sistemul”.

3.3 Inginerul de sistem și specialistul în proiectare

Inginerul de sistem

- lucrează „de sus în jos”, pe orizontală
- acoperă mai multe domenii, la nivel de sistem și subsistem
- se concentrează pe integrare și interfețe

Specialistul în proiectare

- lucrează „de jos în sus”, pe verticală
- un singur domeniu, până la subcomponente și piese
- optimizează componente

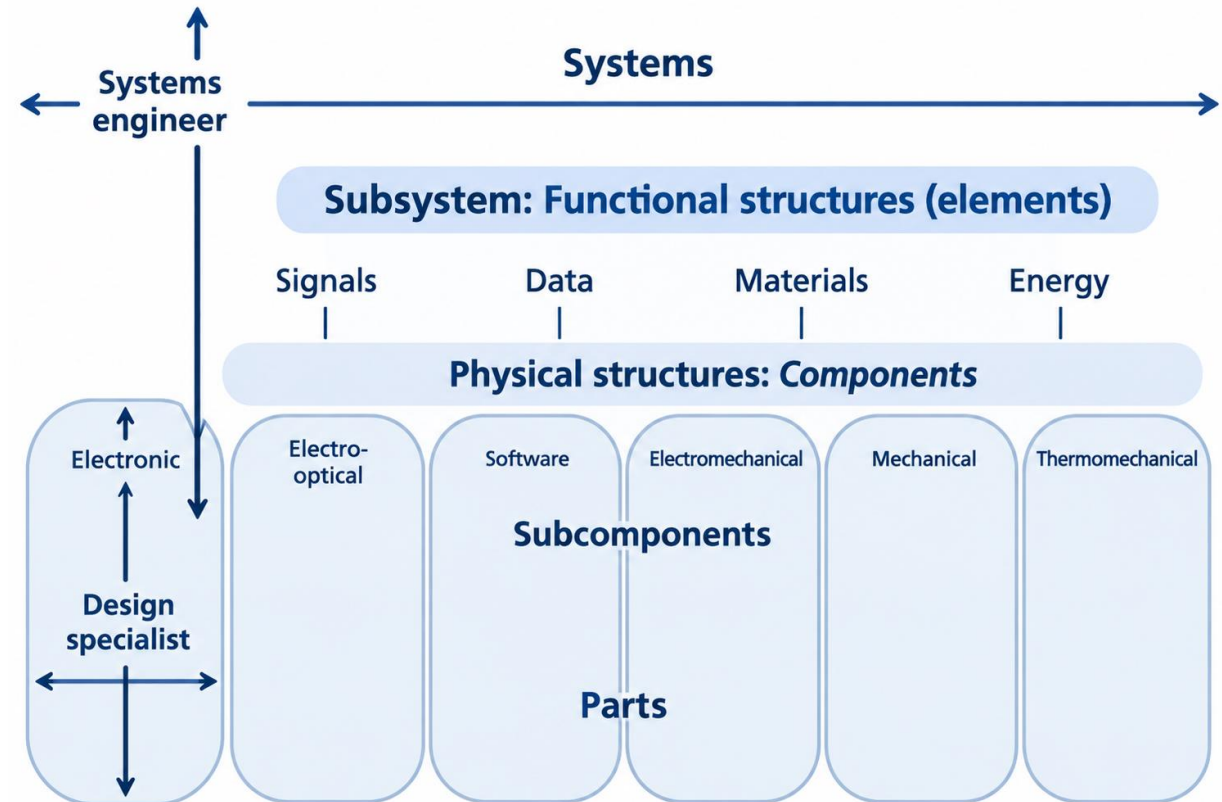


Fig. 3.3. Domeniile de cunoaștere ale inginerului de sistem și ale specialistului.
Sursa: adaptat după [1, cap. 3].

Cele două roluri se completează și se întâlnesc la interfețe [1, cap. 3]; este aceeași idee ca în modelul „T” din Capitolul 2.

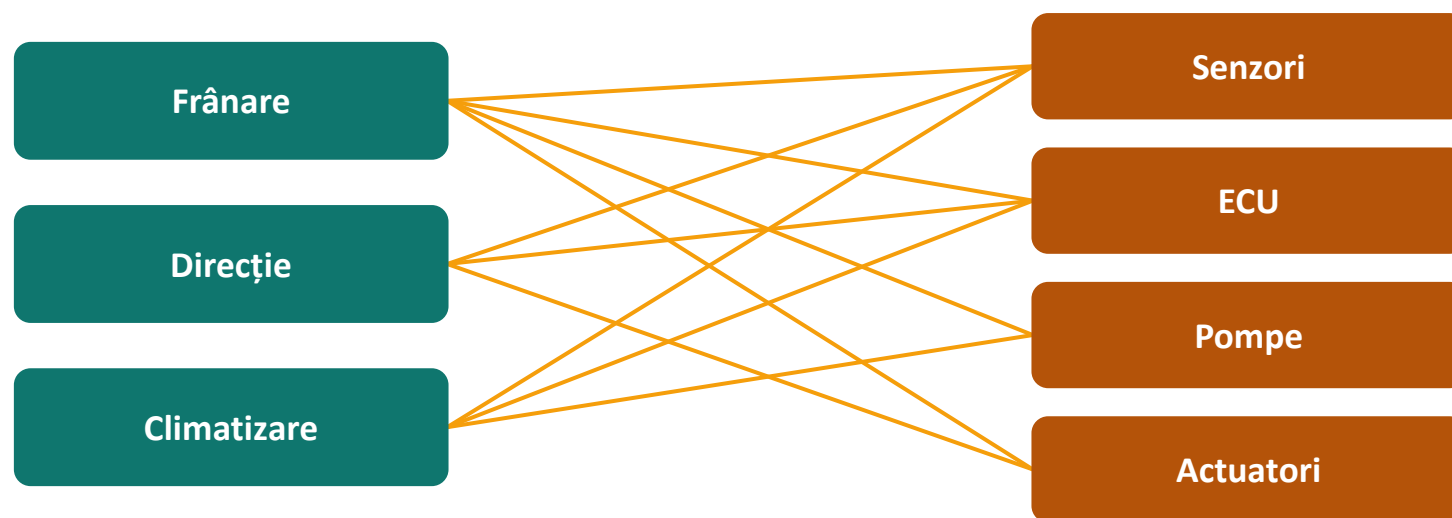
3.4 Structura funcțională și structura fizică

A. Structura funcțională: CE face sistemul

Elemente funcționale care prelucrează **semnale, date, materiale și energie** [1, cap. 3].

B. Structura fizică: DIN CE este construit

Componente **electronice, electro-optice, mecanice, electromecanice, termomecanice, software** [1, cap. 3].



funcții

componente fizice

Fig. 3.4. Corespondența funcții–componente la automobil.

Legăturile nu sunt unu-la-unu: o funcție folosește mai multe componente, iar o componentă servește mai multe funcții. De aceea cele două structuri trebuie gândite împreună. (ECU = unitate electronică de control.)

3.5 Limitele sistemului și mediul

- **Sistemul** este ceea ce proiectăm și livrăm. **Mediul** cuprinde tot ce interacționează cu sistemul fără să-i aparțină: utilizatori, mentenanță, surse de energie, condiții de mediu, amenințări [1, cap. 3].
- Trasarea graniței este critică, dificilă și reprezintă o responsabilitate a inginerului de sistem: ce este în interior trebuie proiectat; ce este în exterior impune cerințe și interfețe.
- Exemplu: șoferul **nu** face parte din sistemul „automobil”, dar interfața șofer–vehicul este critică. Volanul, pedalele, bordul și ecranele aparțin sistemului; omul care le folosește aparține mediului.

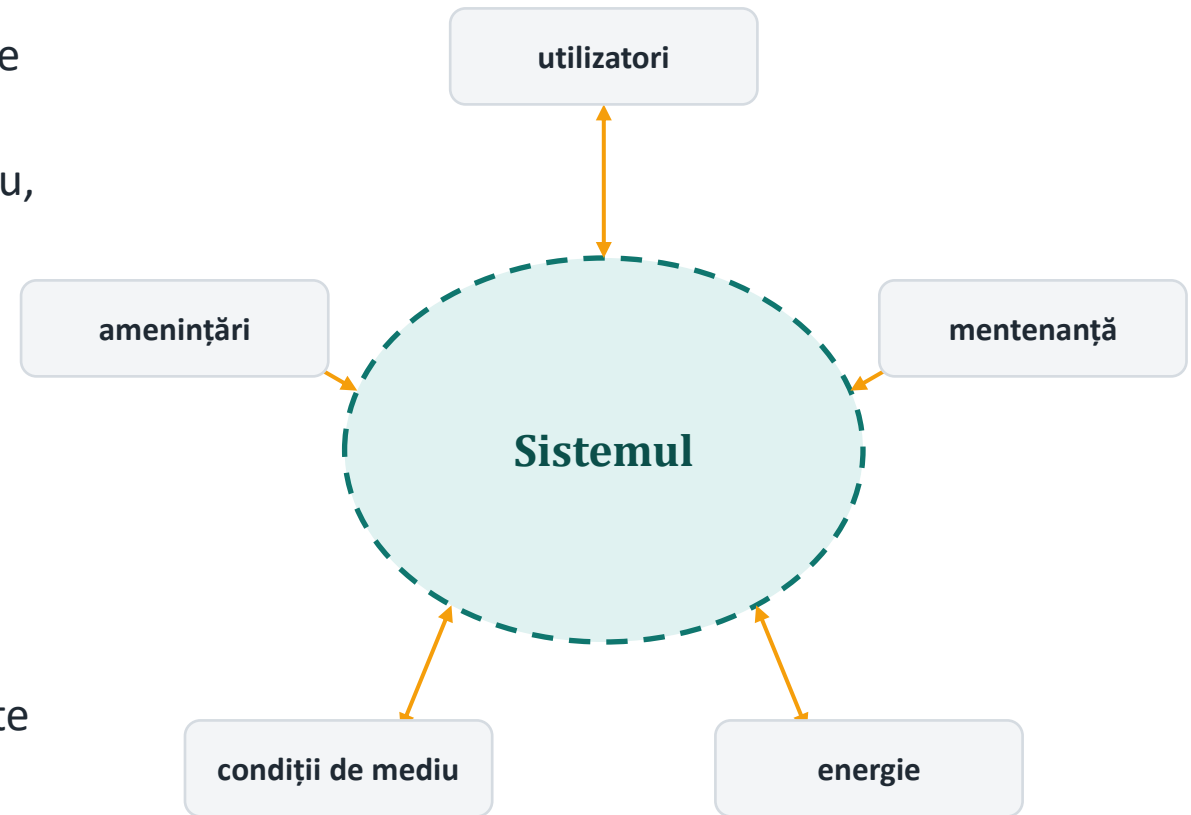


Fig. 3.5. Granița sistemului (linia punctată) și mediul său.

Unde se termină sistemul?



Fig. 3.6. Interfața șofer–vehicul: comenzile și afișajele aparțin sistemului. Sursa: [<https://www.linkedin.com/pulse/automotive-human-machine-interface-hmi-market-growth-trends-patil-zjvgf/>].

Șoferul nu face parte din sistem, dar interfața șofer–vehicul este critică pentru siguranță. Ea va fi tratată ca interfață om–mașină în Capitolul 4.

- Granița se stabilește explicit și se documentează, pentru că determină ce intră în responsabilitatea echipei de proiect și ce devine cerință sau interfață externă.

3.6 Diagrama de context

- Diagrama de context arată sistemul, entitățile externe și interacțiunile dintre ele: date, semnale, energie, materiale, activități [1, cap. 3].
- Ea **nu** arată interiorul sistemului, ci doar relațiile lui cu exteriorul.
- Este folosită pentru a clarifica limitele, a identifica interfețele și a comunica cu stakeholderii.
- Este un instrument foarte simplu, dar practic; vom reveni la el în Partea a II-a, în analiza nevoilor și în modelarea arhitecturii.

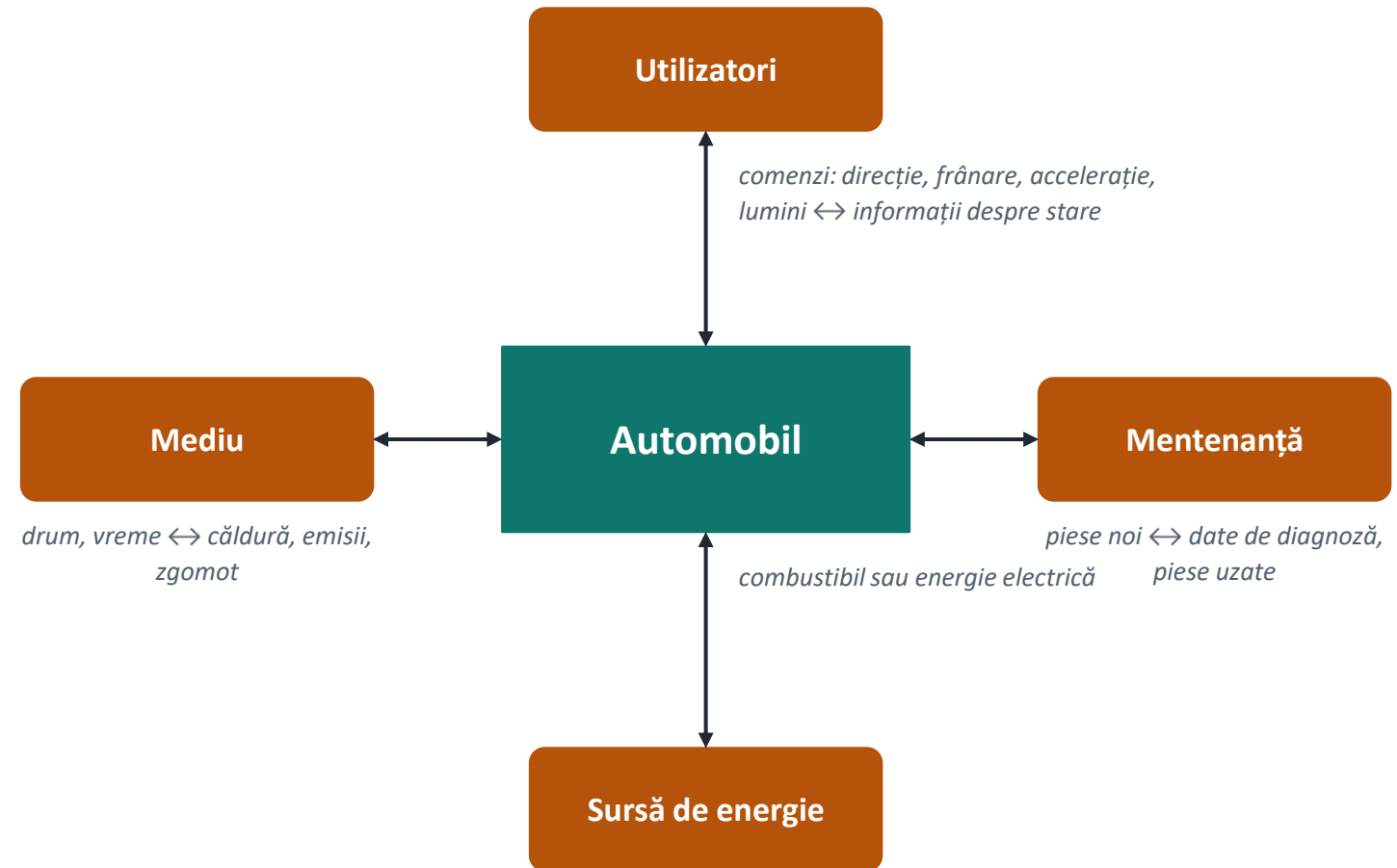


Fig. 3.7. Diagrama de context pentru un automobil. Sursa: adaptat după [1, cap. 3].

Concluzii

- 1 Sistem → subsistem → componentă → subcomponentă → piesă: ierarhia controlează complexitatea.
- 2 Ce numim „sistem” depinde de nivelul de analiză ales.
- 3 Structura funcțională spune CE face sistemul; cea fizică, DIN CE este construit.
- 4 Granița sistemului se trasează explicit, cu ajutorul diagramei de context.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Alegeți un produs pe care îl cunoașteți și descompuneți-l pe cele cinci niveluri ierarhice, de la sistem la piesă.

Tema 2

Desenați diagrama de context pentru un sistem de la locul de muncă. Ce entități externe ați identificat și ce fluxuri le leagă de sistem?

CAPITOLUL 4

Interfețe și interacțiuni

Cuprinsul capitolului

4.1 De la interacțiuni la interfețe

4.2 Interfețe externe și interne

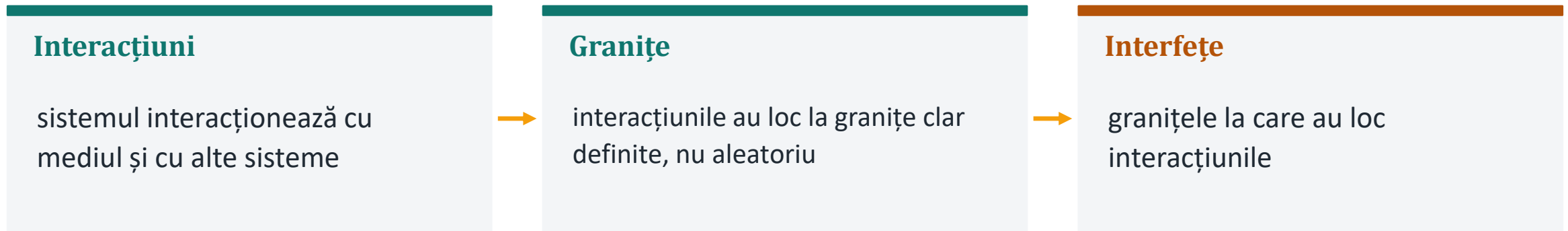
4.3 Clasificarea interfețelor

4.4 Interfețele cu mediul: avionul de pasageri

4.5 Interfața om–mașină

Concluzii · Teme de aplicare

4.1 De la interacțiuni la interfețe



- O **interfață** este elementul care permite interacțiunea dintre două părți ale unui sistem sau dintre sistem și mediu. Prin interfață se transferă energie, informație, forțe sau materiale [1, cap. 3].
- Exemple intuitive: mâinile șoferului și volanul (forță și informație, inclusiv vibrațiile care „spun” ceva despre drum); anvelopele și șoseaua (forțe de aderență).
- Identificarea interfețelor este o responsabilitate-cheie a inginerului de sistem.

4.2 Interfețe externe și interne

- **Interfețele externe** leagă sistemul de mediu și de alte sisteme.
- **Interfețele interne** leagă subsistemele și componentele între ele [1, cap. 3].
- Componentele sunt de regulă bine înțelese de specialiștii care le proiectează. Interfețele, în schimb, aparțin „nimănui” dacă nu sunt definite explicit; de aceea **majoritatea problemelor de integrare apar la interfețe, nu în componente.**
- Definirea și controlul interfețelor sunt, prin urmare, o activitate centrală a ingineriei sistemelor pe tot ciclul de viață.

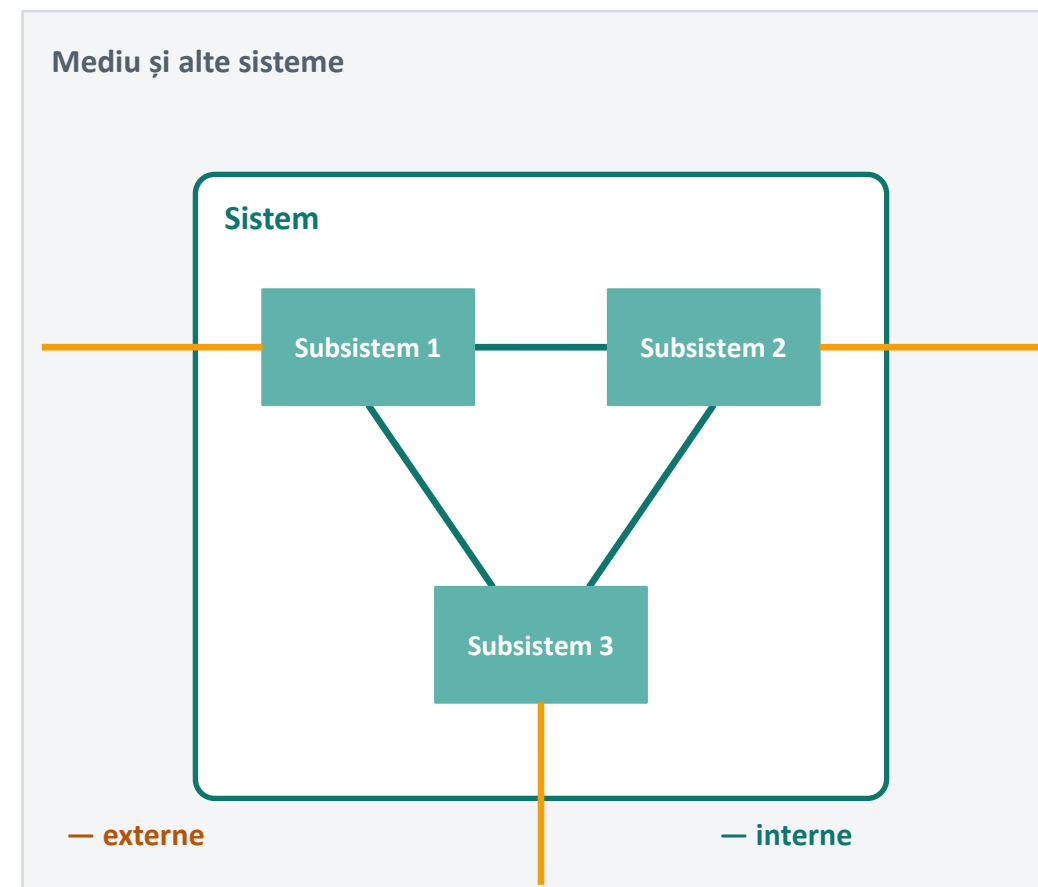


Fig. 4.1. Interfețe externe (portocaliu) și interne (verde).

4.3 Clasificarea interfețelor

După rolul lor principal, elementele de interfață se grupează în trei categorii [1, cap. 3]; clasificarea ajută la proiectarea și verificarea interfețelor.

Conectare

- leagă elementele și transmite fluxuri
ex.: cabluri, cuplaje, conectori

Izolare

- separă și protejează elementele
ex.: carcase, etanșări, ecranări

Conversie

- transformă o formă de energie sau semnal în alta
ex.: motoare, senzori, convertoare

Exemplu: interfețe în automobil

Tip	Exemple	Externă / internă
Conectare	mufa de încărcare; conectorul unității ECU	externă (vehicul–rețea); internă
Izolare	garnitura de chiulasă; carcasa bateriei	interne
Conversie	motorul electric; senzorul de viteză ABS	interne

4.4 Interfețele cu mediul: avionul de pasageri

- Avionul de pasageri interacționează cu cinci medii externe, fiecare cu interfețe și cerințe specifice [1, cap. 3]:
 - **mediul de zbor:** ploaie, vânt, turbulențe, comenzi de la turnul de control;
 - **mediul de aterizare:** sistemul ILS, șocuri și vibrații;
 - **mediul de suport la sol:** energie, combustibil;
 - **mediul de întreținere;**
 - **pasageri și încărcătură.**
- Sistemul interacționează simultan cu factori naturali, infrastructură și operatori umani. Interfețele sunt multiple, diferite ca natură și critice pentru siguranță.

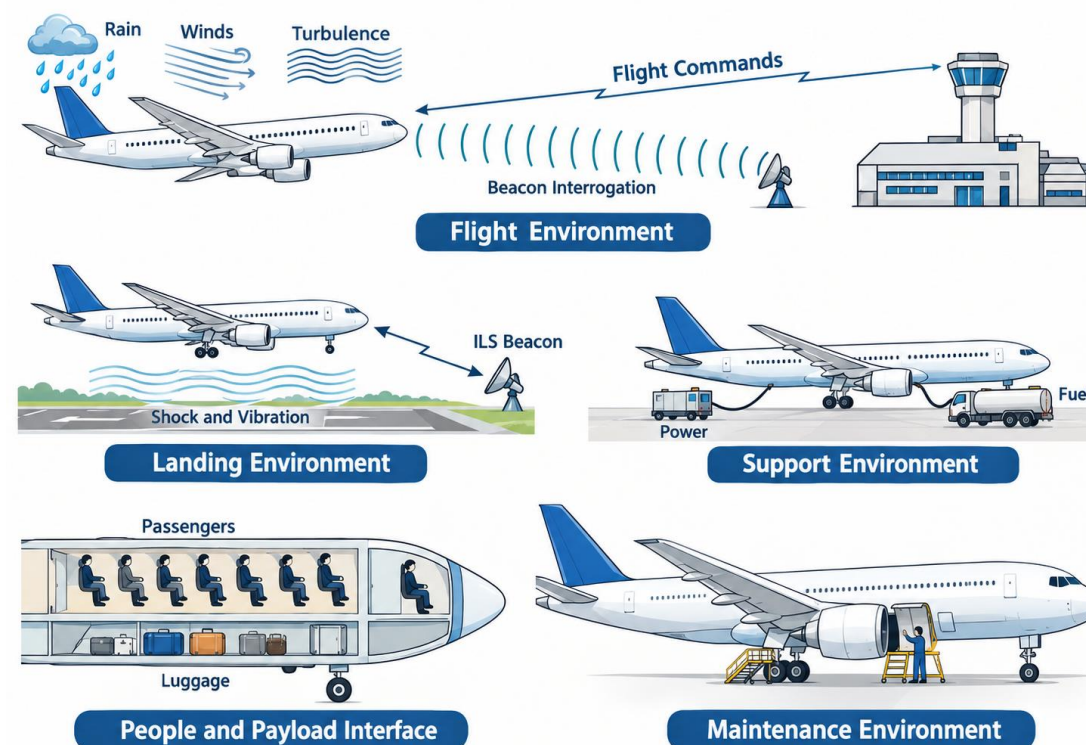


Fig. 4.2. Mediile externe ale unui avion de pasageri. Sursa: [1, cap. 3].

4.5 Interfața om–mașină (HMI)

- Operatorii umani apar în toate etapele de utilizare a sistemului: pilotaj, întreținere, suport la sol.
- Interfața om–mașină (HMI, *Human–Machine Interface*) este critică pentru controlul sistemului, pentru feedback-ul către operator și pentru siguranța operațională [1, cap. 3].
- Omul face parte din mediu, dar interfața om–mașină face parte din sistem (Capitolul 3). Proiectarea ei trebuie să țină seama de capacitățile și limitele operatorului.
- Automatizarea reduce sarcina operatorului, dar nu elimină rolul omului.

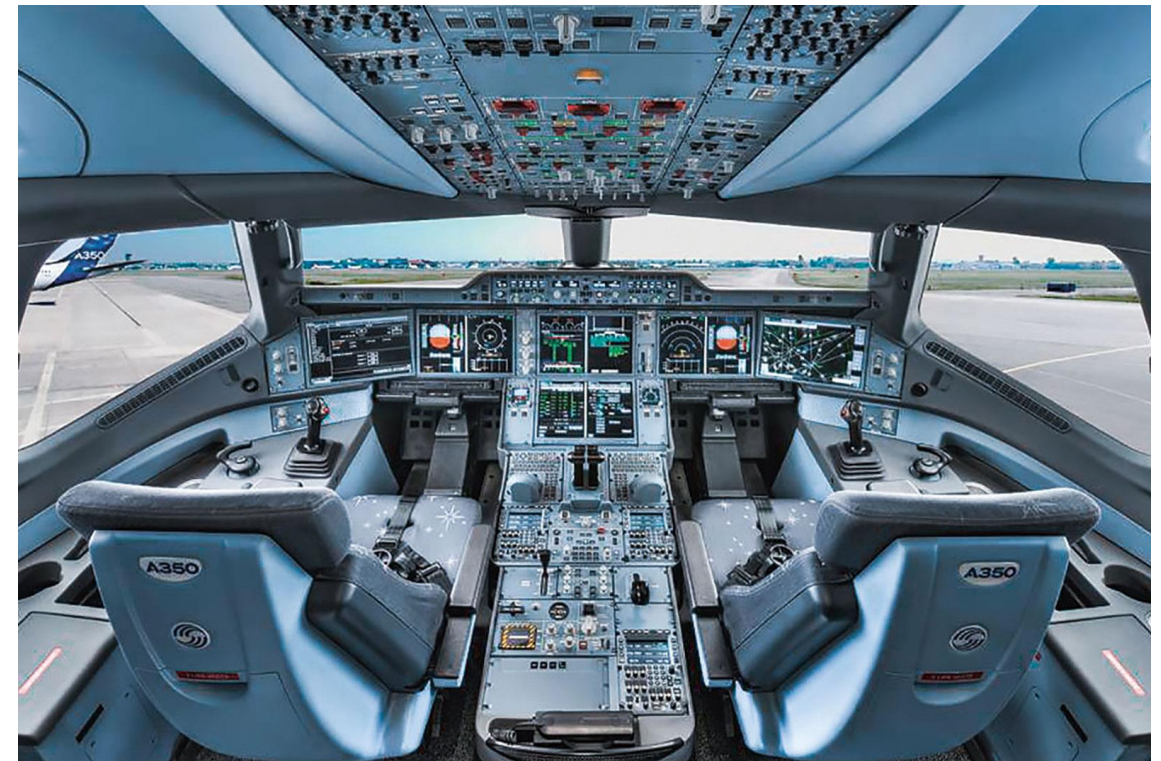


Fig. 4.3. Cockpitul unui avion. Sursa: [<https://aeroreport.de/de/good-to-know/gestern-heute-cockpit>].

Concluzii

1

Interfața este granița la care are loc o interacțiune.

2

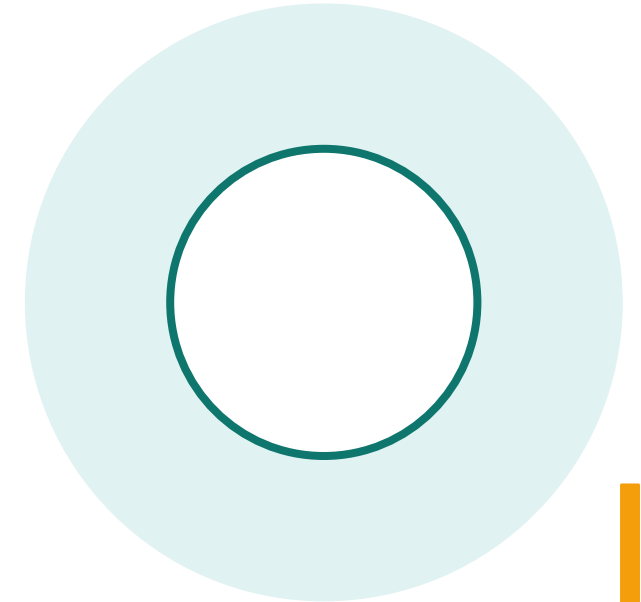
Interfețele externe leagă sistemul de mediu; cele interne, subsistemele între ele.

3

Trei tipuri de interfețe: de conectare, de izolare și de conversie.

4

Problemele de integrare apar, de regulă, la interfețe; interfața om–mașină este critică.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Pentru sistemul ales în Capitolul 3, identificați cel puțin trei interfețe critice, clasificați-le (conectare, izolare, conversie) și explicați ce s-ar întâmpla dacă una dintre ele ar fi proiectată greșit.

Tema 2

Alegeți un sistem cu operator uman. Descrieți rolul operatorului, explicați de ce acesta face parte din mediu și nu din sistem și dați un exemplu de risc generat de o interfață om–mașină slab proiectată.

CAPITOLUL 5

Managementul ingineriei sistemelor

Cuprinsul capitolului

5.1 Nevoia unui management tehnic

5.2 Ingineria sistemelor în managementul de proiect

5.3 Faza de propunere

5.4 Structura detaliată a lucrărilor (WBS)

5.5 Testarea, responsabilitate de sistem

5.6 Planul de management al ingineriei sistemelor (SEMP)

Concluzii · Teme de aplicare

5.1 Nevoia unui management tehnic

Proiectele complexe nu pot fi conduse doar prin planificare și buget. Este nevoie de o funcție care [1, cap. 5]:

ghidează deciziile tehnice

coordonează disciplinele

asigură integrarea sistemului

- Această funcție este **managementul ingineriei sistemelor**: componenta tehnică a managementului de proiect [1, cap. 5], [4].
- Ea nu dublează managementul proiectului, ci îl completează. Cele două împart o parte a responsabilităților și se diferențiază prin obiectul optimizării.
- Capitolul descrie instrumentele prin care se exercită: participarea la propunere, structura detaliată a lucrărilor (WBS), coordonarea testării și planul de management al ingineriei sistemelor (SEMP).

5.2 Ingineria sistemelor în managementul de proiect

Ingineria sistemelor

- arhitectura sistemului
- coordonare tehnică
- integrare și validare

Funcții comune

- alocarea resurselor
- definirea sarcinilor
- managementul riscului
- relația cu clientul

Planificare și control

- planificare și termene
- buget și contracte
- raportare

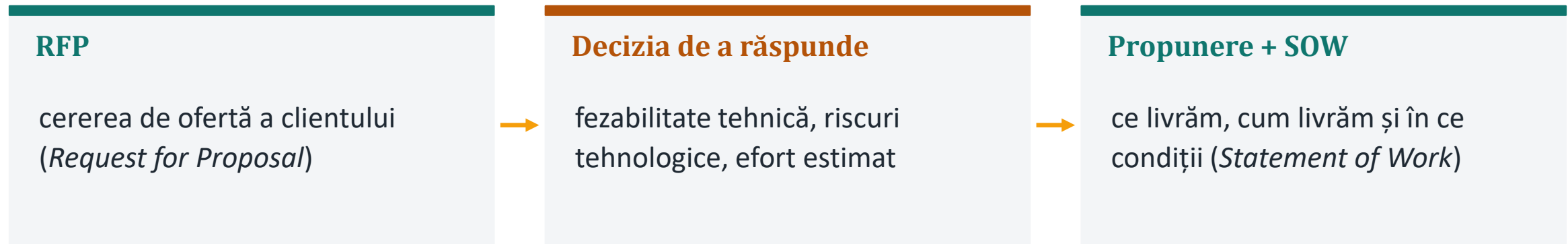
← CE construim și CUM funcționează

CÂND livrăm și CÂT costă →

Fig. 5.1. Ingineria sistemelor și planificarea/controlul în managementul de proiect. Sursa: adaptat după [1, cap. 5].

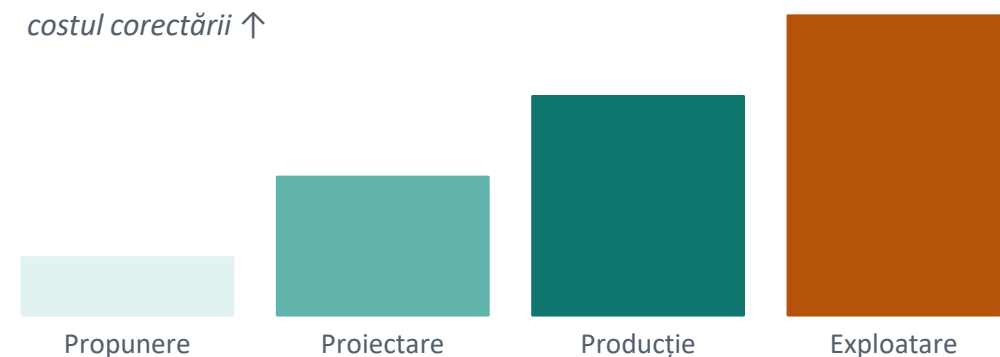
Managementul proiectului optimizează **timpul și costul**; ingineria sistemelor optimizează **funcționarea sistemului**. De aceea un proiect poate fi „la timp și în buget”, iar sistemul să nu funcționeze corect.

5.3 Faza de propunere



- Ingineria sistemelor începe încă din faza de propunere, înainte de semnarea contractului [1, cap. 5].
- Inginerul de sistem evaluează conceptele tehnice, contribuie la redactarea SOW și asigură **realismul tehnic** al propunerii: un angajament imposibil de respectat, asumat la propunere, devine o problemă a întregului proiect.

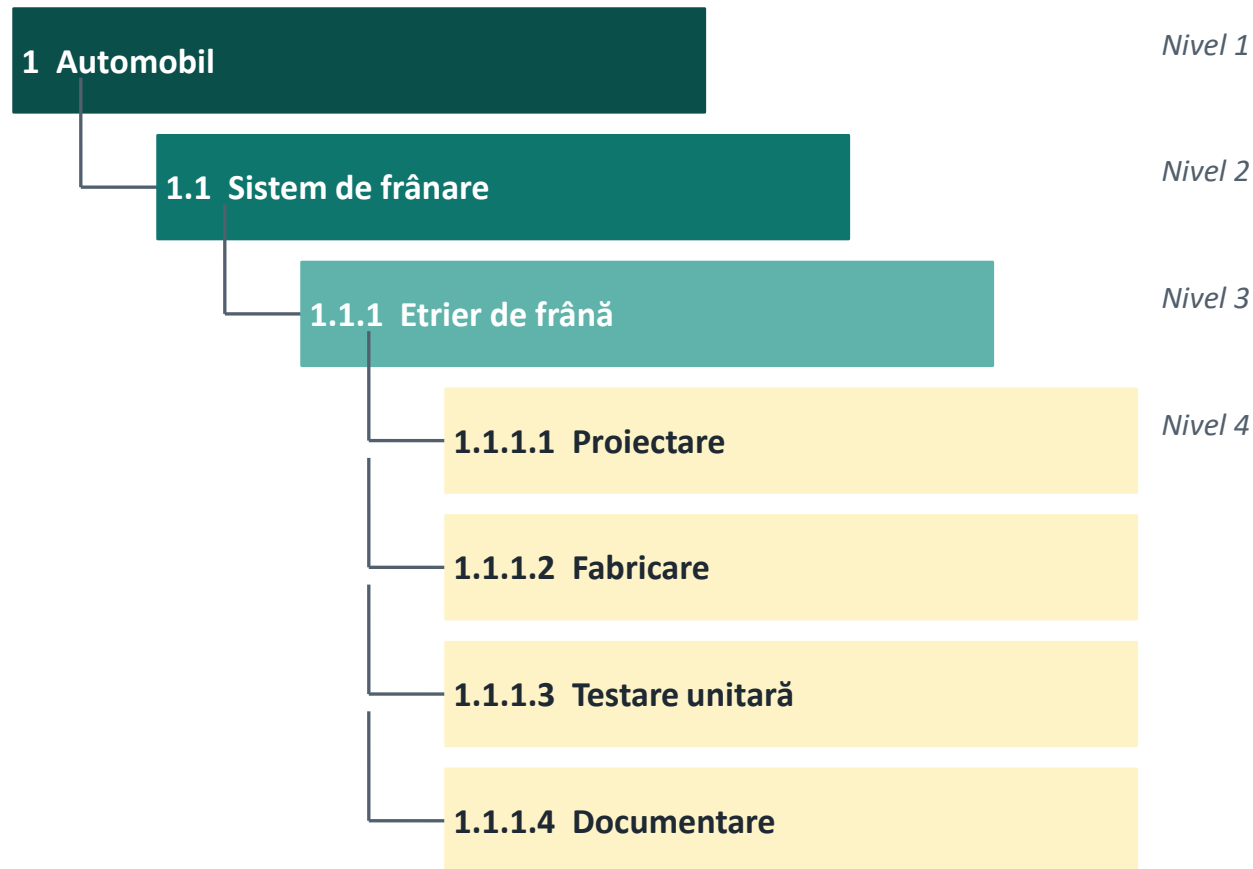
Problemele ignorate la propunere devin costuri mari în execuție: costul corectării unei erori crește cu cât aceasta este descoperită mai târziu în ciclul de viață [3], [4].



5.4 Structura detaliată a lucrărilor (WBS)

- WBS (*Work Breakdown Structure*) descompune ierarhic **întreaga muncă** a proiectului. Este, de regulă, o cerință contractuală și coloana vertebrală a planificării, a estimării costurilor și a urmăririi progresului [1, cap. 5], [4].
- Un WBS complet acoperă: produsul (hardware și software), suportul sistemului, testarea, managementul proiectului și ingineria sistemelor.
- Un WBS bun reflectă structura sistemului (Capitolul 3), dar **nu este o listă de componente**: elementele de pe ultimul nivel descriu activități, exprimate prin verbe de acțiune (proiectare, integrare, testare, documentare).
- Inginerul de sistem este implicat în construirea WBS de la început.

Exemplu: fragment de WBS pentru automobil



Nivelul 4 conține **activități, nu piese**: etrierul se proiectează, se fabrică, se testează și se documentează. Pe nivelul 3 apar și echipamentele de asamblare necesare integrării.

Fig. 5.2. Fragment de WBS pentru automobil. Sursa: adaptat după [1, cap. 5].

5.5 Testarea, responsabilitate de sistem



- Testarea nu înseamnă doar verificarea pieselor. Cea mai mare parte a efortului se desfășoară la nivel de sistem, urcând de la integrarea subsistemelor până la testarea în condiții reale de operare [1, cap. 13].
- Scopul este funcționarea **ca ansamblu**, nu doar conformitatea locală. Ingineria sistemelor planifică și coordonează această etapă; nivelurile de testare sunt reluate în Partea a II-a, la integrare și evaluare.

5.6 Planul de management al ingineriei sistemelor (SEMP)

- SEMP (*Systems Engineering Management Plan*) este documentul care [1, cap. 5], [5]:
 - **planifică** toate activitățile de inginerie a sistemelor;
 - **definește** roluri și responsabilități;
 - **coordonează** disciplinele tehnice și planurile de specialitate.
- SEMP începe ca o schiță, în faza de propunere, și este elaborat și actualizat pe tot parcursul dezvoltării: un **document dinamic**.
- Rezultatul: toți participanții știu ce au de făcut și cum se leagă munca lor de a celorlalți.

Locul SEMP în planul proiectului

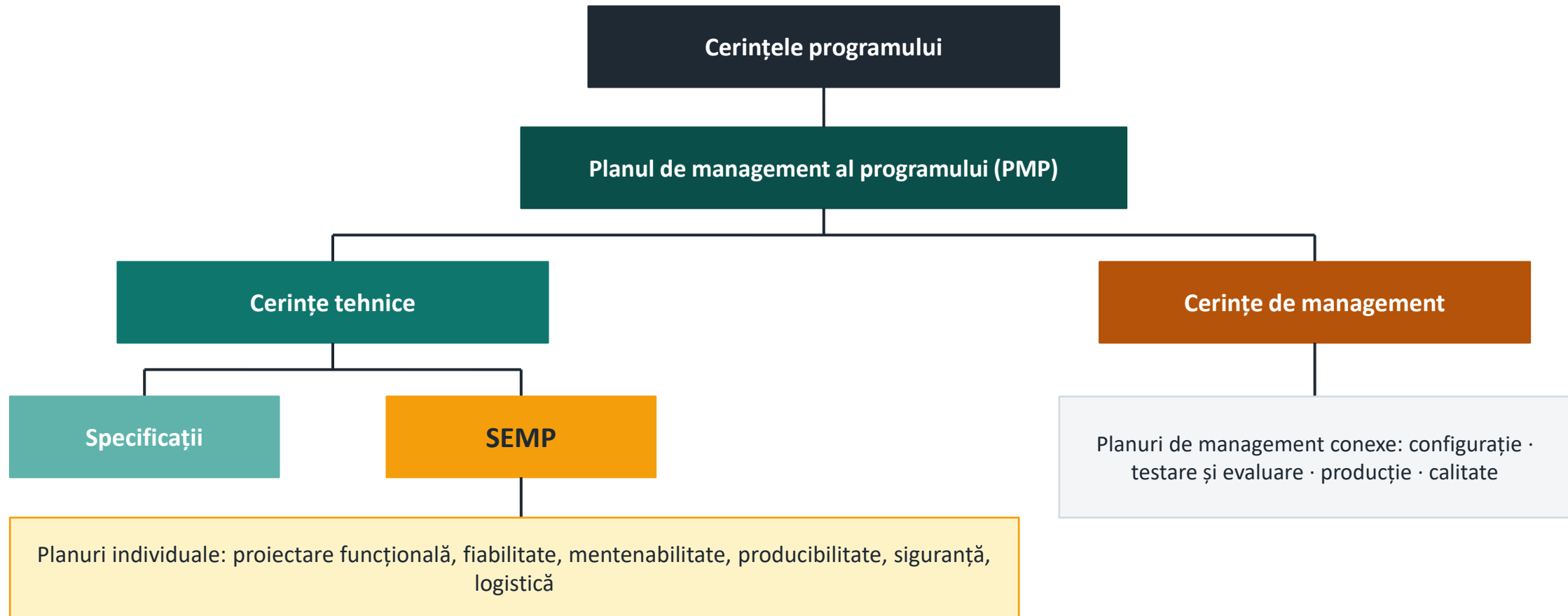


Fig. 5.3. Locul SEMP în structura planurilor unui proiect. Sursa: adaptat după [1, cap. 5].

Concluzii

- 1 Ingineria sistemelor este componenta tehnică a managementului de proiect, pe care îl completează.
- 2 Ea începe din faza de propunere: RFP, decizia de a răspunde, SOW.
- 3 WBS descompune munca, nu doar produsul, folosind verbe de acțiune.
- 4 SEMP leagă toate activitățile tehnice și se actualizează pe tot parcursul proiectului.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Pentru sistemul ales în Capitolul 3, construiți un WBS pe patru niveluri. Ultimul nivel trebuie exprimat prin verbe de acțiune.

Tema 2

Într-un proiect de la locul de muncă, identificați trei funcții comune și trei diferențe între managementul proiectului și ingineria sistemelor.

CAPITOLUL 6

Managementul riscurilor: concepte fundamentale

Cuprinsul capitolului

6.1 Riscul: efectul incertitudinii asupra obiectivelor

6.2 Amenințare, vulnerabilitate, risc

6.3 Componentele managementului riscurilor

6.4 Un proces continuu, pe tot ciclul de viață

6.5 Evoluția riscului și a efortului

6.6 Când trebuie gestionat riscul?

Concluzii · Teme de aplicare

6.1 Riscul: efectul incertitudinii asupra obiectivelor

„Toate organizațiile, indiferent cât de mari sau mici sunt, se confruntă cu factori interni și externi care creează incertitudini legate de posibilitatea de a-și atinge propriile obiective. Efectul acestei incertitudini reprezintă riscul, care este inherent în toate activitățile.” *Kevin W. Knight, membru al grupului de lucru care a elaborat ISO 31000*

- Standardul ISO 31000 definește riscul ca **efectul incertitudinii asupra obiectivelor** [13]. Accentul se mută de pe „eveniment negativ” pe efectul asupra a ceea ce proiectul își propune.
- În ingineria sistemelor, obiectivele afectate sunt **tehnice** (performanța), **de cost, de termen și de siguranță**.
- Riscul nu poate fi eliminat dintr-un proiect tehnic; el se **gestionează**, sistematic, prin identificare, evaluare și măsuri preventive și reactive (Capitolul 7).

6.2 Amenințare, vulnerabilitate, risc

Amenințare

- factor **extern** care poate produce un eveniment negativ
ex.: atac cibernetic, vreme extremă, reglementare nouă

Vulnerabilitate

- slăbiciune **internă** care permite amenințării să producă efecte
ex.: interfețe nevalidate, lipsa redundanței, proceduri neclare

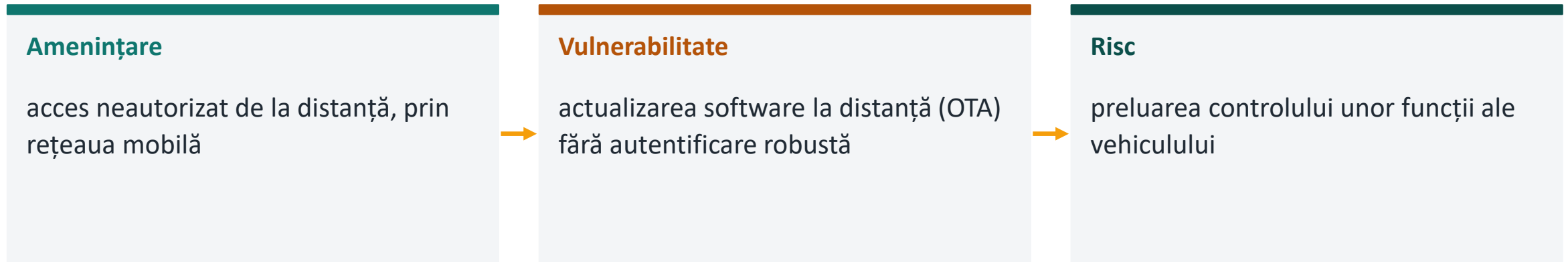
Risc

- probabilitatea ca o amenințare să exploateze o vulnerabilitate și să producă un **impact** negativ

Amenințare × Vulnerabilitate → Impact = Risc (schemă de raționament, nu relație matematică)

- Amenințarea este externă și, de regulă, nu o controlăm. Vulnerabilitatea este internă și ține de proiectarea sistemului.
- De aceea ingineria sistemelor acționează în primul rând asupra **vulnerabilităților**.

Aplicație: automobilul conectat



- Firul roșu al lucrării, automobilul, este astăzi un sistem conectat: software-ul său poate fi actualizat prin rețeaua mobilă (OTA, *Over-The-Air*).
- Securitatea cibernetică a vehiculelor este reglementată la nivel internațional, prin Regulamentul ONU nr. 155 [14]; exemplul ține deci și de riscurile generate de reglementări.

Unde intervine inginerul de sistem? Pe **vulnerabilitate**: amenințarea există oricum, dar interfața poate fi proiectată sigur.

6.3 Componentele managementului riscurilor

- Managementul riscurilor reunește mai multe activități, care protejează proiectul ca o „umbrelă” [1, cap. 5]:
 - cunoașterea amenințărilor și a vulnerabilităților;
 - identificarea și ierarhizarea riscurilor;
 - analiza probabilitate–efect;
 - estimarea costurilor;
 - adoptarea măsurilor preventive și reactive.
- Evaluarea răspunde la două întrebări: **cât de probabil** este riscul și **cât de grav** este impactul. Atenuarea cuprinde revizii, analize și teste suplimentare, prototipare, cerințe mai riguroase și soluții de rezervă.



Fig. 6.1. Componentele managementului riscurilor.

6.4 Un proces continuu, pe tot ciclul de viață

Riscul este inevitabil în proiectele de sisteme, deoarece apar tehnologii noi, interfețe complexe și constrângeri de timp și cost [1, cap. 5]. Un **plan de management al riscurilor** stabilește:

Reguli

cum identificăm și cum evaluăm riscurile

Responsabilități

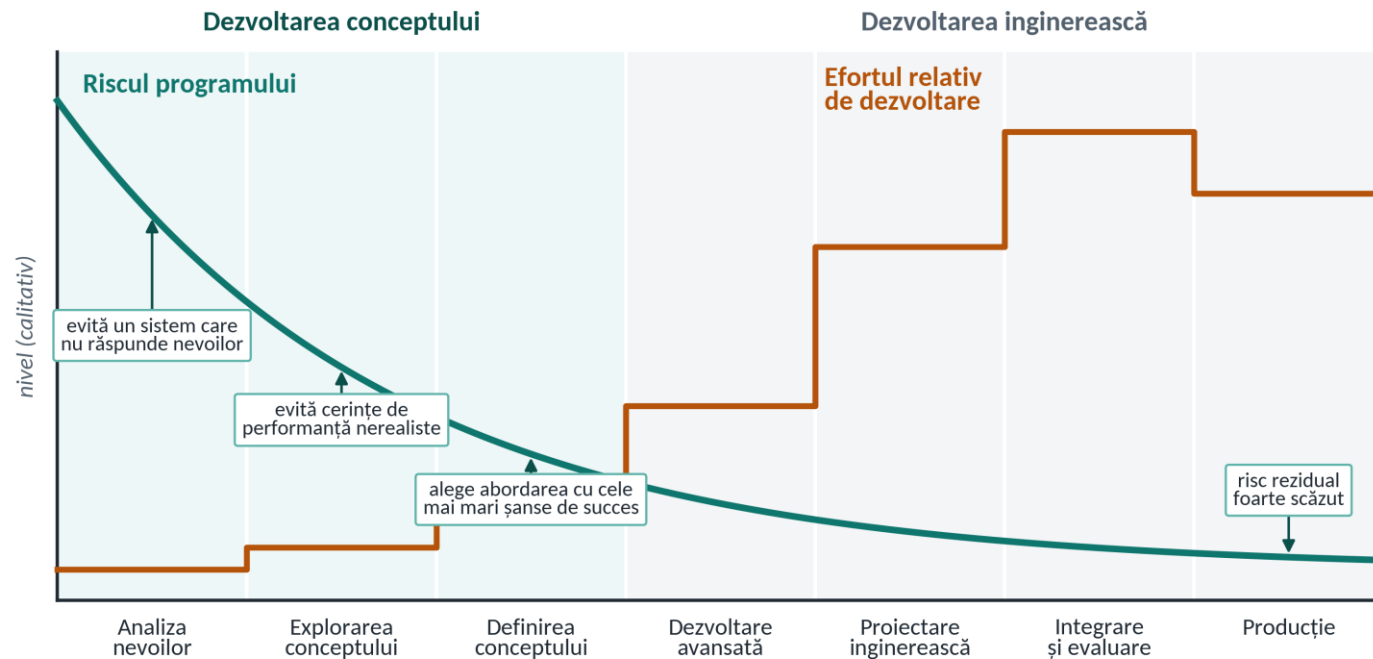
cine „deține” fiecare risc
(proprietarul riscului, *risk owner*)

Acțiuni

cum atenuăm riscurile și cum le monitorizăm

Managementul riscurilor nu este o etapă, ci o activitate care însoțește întregul ciclu de viață. Riscul trebuie redus **pe măsură ce investiția crește**, nu „la final”.

6.5 Evoluția riscului și a efortului



- La început, incertitudinea este ridicată, iar riscul tehnic este maxim. Pe parcurs, cerințele se clarifică, arhitectura se stabilizează, testarea crește, iar riscul scade [1, cap. 5].
- Fiecare fază timpurie reduce un anumit risc: analiza nevoilor, pe cel al unui sistem inutil; explorarea conceptului, pe cel al unor cerințe nerealiste; definirea conceptului alege abordarea cu cele mai mari șanse de succes.
- Rolul ingineriei sistemelor: să identifice riscurile **devreme** și să prevină surprizele târzii.

Fig. 6.2. Variația riscului programului și a efortului de dezvoltare. Sursa: adaptat după [1, cap. 5].

6.6 Când trebuie gestionat riscul?

Decizii timpurii

- impact mare asupra costului final
- ieftin de corectat

Decizii târzii

- impact mic asupra arhitecturii
- foarte costisitoare de corectat

- Paradoxul fazelor timpurii: deciziile au cea mai mare influență asupra costului total tocmai când informația disponibilă este cea mai puțină [3], [4].
- Un management eficient al riscurilor înseamnă investiție timpurie în analiză, testare și prototipare înainte de angajamentele majore, adică „**învățare devreme**”. Este aceeași logică a costului corectării din Capitolul 5.

Este mai ieftin să reduci riscul devreme decât să reperi problemele târziu.

Concluzii

- 1 Riscul este efectul incertitudinii asupra obiectivelor: tehnice, de cost, de termen, de siguranță.
- 2 Amenințarea este externă; vulnerabilitatea este internă și ține de proiectare.
- 3 Managementul riscurilor este continuu, pe tot ciclul de viață.
- 4 Riscul se reduce cel mai ieftin devreme, când investiția este încă mică.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

SEMP-ul face parte din propunerea unui contractor, când sistemul este încă un concept. De unde ați obține informațiile pentru fiecare element al unui SEMP tipic?

Tema 2

Pentru un sistem ales, formulați trei riscuri complete (amenințare, vulnerabilitate, impact) și indicați ce obiectiv afectează fiecare.

CAPITOLUL 7

Identificarea, evaluarea și atenuarea riscurilor

Cuprinsul capitolului

7.1 Procesul de management al riscurilor

7.2 Clasificarea riscurilor

7.3 Identificarea riscurilor

7.4 Evaluarea riscurilor: matricea și cubul de risc

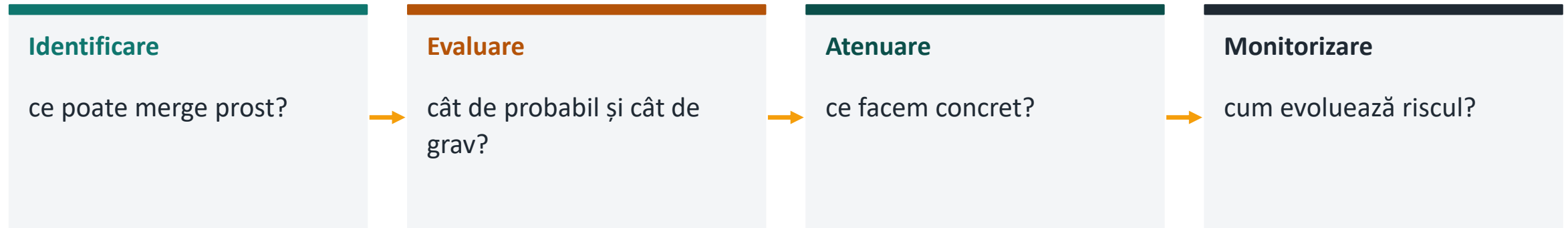
7.5 Atenuarea riscurilor

7.6 Planul de management al riscului și riscul rezidual

7.7 Studiu de caz: controlul traficului aerian

Concluzii · Teme de aplicare

7.1 Procesul de management al riscurilor



- Managementul riscurilor parcurge patru etape, care se reiau la fiecare revizie tehnică și la fiecare decizie majoră [1, cap. 5], [13].
- Monitorizarea se realizează prin **planul de management al riscului**, un document dinamic (secțiunea 7.6).
- Un risc mare, dar cunoscut, poate fi evaluat și atenuat; un risc neidentificat nu intră în niciun plan. De aceea **identificarea condiționează întregul proces**.

7.2 Clasificarea riscurilor

Tehnice

- afectează funcționarea sistemului

ex.: tehnologie nouă, insuficient validată (baterie nouă, algoritm netestat)

De integrare

- afectează interfețele și compatibilitatea

ex.: interfețe neclare între subsisteme (incompatibilitate hardware–software)

Programatice

- afectează timpul și resursele

ex.: subestimarea duratei activităților critice (testare insuficient planificată)

Operaționale

- afectează utilizarea și exploatarea

ex.: sistem greu de utilizat sau de întreținut (interfață om–mașină slab proiectată)

- Clasificarea ajută la alocarea responsabilităților (riscurile programatice revin managerului de proiect, cele de integrare inginerului de sistem), la alegerea măsurilor de atenuare și la prioritizare.

Un risc tehnic ignorat devine adesea un risc de program și de cost.

7.3 Identificarea riscurilor

- Identificarea este prima etapă a managementului riscurilor. Scopul nu este eliminarea tuturor riscurilor, ci **recunoașterea lor timpurie** și înțelegerea impactului posibil asupra sistemului.
- Identificarea se face cât mai devreme în ciclul de viață și se revizuieste periodic.

Cine identifică? O responsabilitate colectivă

- inginerul de sistem și managerul de proiect
- inginerii de subsistem
- specialiștii în testare și integrare

Când? În momentele-cheie ale proiectului

- în faza de concept
- la reviziile tehnice
- înainte de deciziile majore (*milestones*)

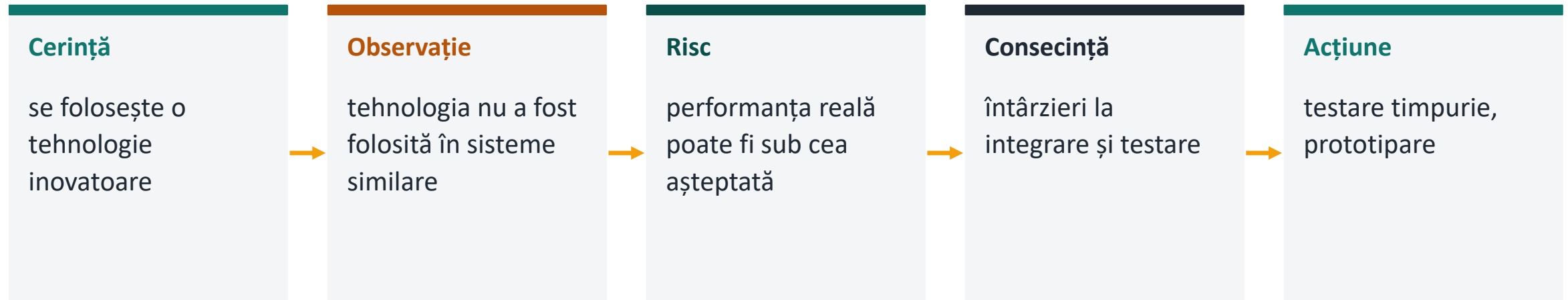
Riscul ignorat devine problemă sigură. Cele mai bune riscuri sunt identificate înainte să existe presiune de timp.

Surse tipice de risc

Cerințe	Tehnologie	Interfețe	Integrare și testare	Factori externi
incomplete, ambigue, contradictorii	maturitate scăzută, lipsa experienței	slab definite, multe dependențe	planificare insuficientă, testare târzie	reglementări, furnizori, mediu

- Cele cinci surse acoperă aproape toate riscurile întâlnite în practică. **Cerințele** sunt sursa cea mai frecventă; formularea lor corectă este tratată în Capitolul 10.
- Majoritatea riscurilor nu apar dintr-o singură cauză, ci din combinarea mai multor factori.

Exemplu: de la observație la risc



- Fiecare risc bine formulat poate fi descris prin acest lanț logic: cerință → observație → risc → consecință → acțiune.

Riscul apare din analiză logică, nu din intuiție.

7.4 Evaluarea riscurilor: matricea de risc

- După identificare, riscurile se evaluează pe două dimensiuni: **probabilitatea** apariției și **impactul** asupra sistemului sau proiectului [1, cap. 5].
- Matricea de risc combină cele două dimensiuni, fiecare pe o scară de la 1 la 5, și produce o ierarhizare a riscurilor.
- Un risc cu impact mare, chiar puțin probabil, poate fi mai important decât unul frecvent, dar minor (de exemplu, căderea completă a sistemului de frânare).
- Evaluarea nu este exactă matematic; este un **sprijin pentru decizie**.

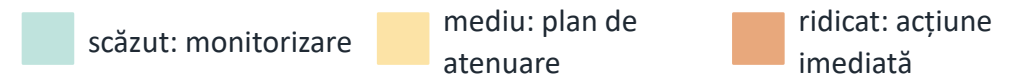
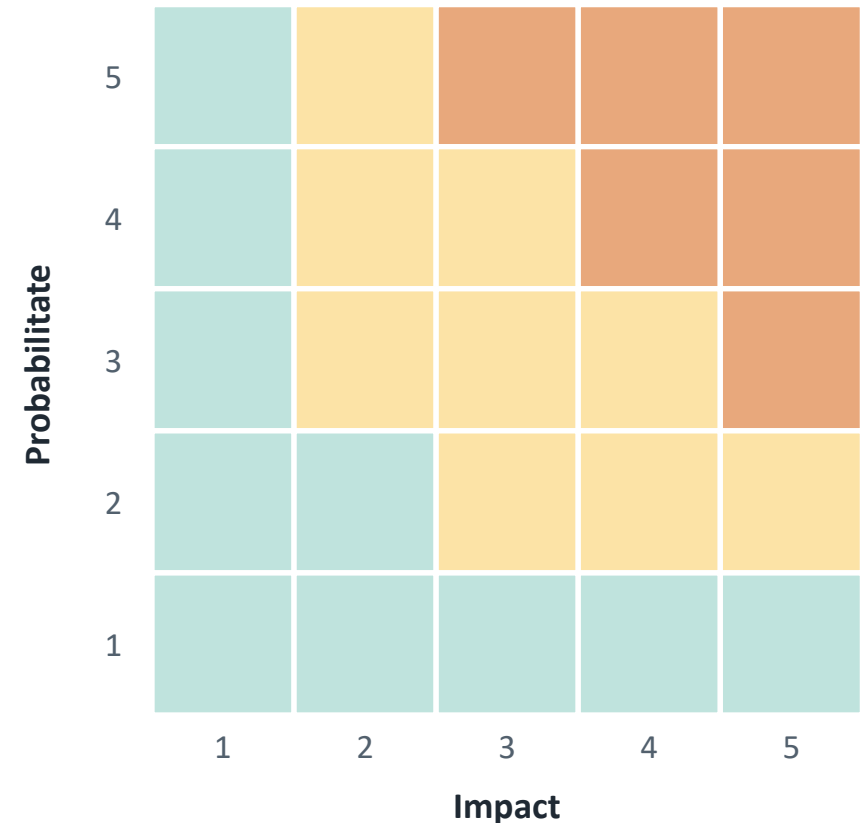


Fig. 7.1. Matricea de risc: probabilitate × impact.

Cubul de risc (1/2): scala probabilității

Fiecare nivel de probabilitate se definește prin încrederea în abordarea tehnică și în procesele proprii: **„Cât de probabil este ca riscul să se producă?”** [1, cap. 5].

Tabelul 7.1. Scala probabilității (1–5). Sursa: adaptat după [1, cap. 5].

Nivel	Probabilitate	Abordarea și procesele dumneavoastră...
1	puțin probabil	...vor evita sau vor atenua eficient acest risc, pe baza practicilor standard
2	probabilitate scăzută	...au atenuat de regulă acest tip de risc, cu supraveghere minimă, în cazuri similare
3	probabil	...pot atenua acest risc, dar sunt necesare soluții alternative
4	foarte probabil	...nu pot atenua acest risc, dar o abordare diferită ar putea
5	aproape sigur	...nu pot atenua acest tip de risc; nu se cunosc alternative

Cubul de risc (2/2): scala consecințelor

Consecințele se evaluează separat pe trei axe: **tehnic** (performanță), **termen** (program) și **cost** (buget): „**Dacă riscul se produce, cât de mare este impactul?**” Rezultatul se plasează în matricea 5 × 5.

Tabelul 7.2. Scala consecințelor (1–5). Sursa: adaptat după [1, cap. 5].

Nivel	Tehnic	Termen	Cost
1	impact minim sau nul	impact minim sau nul	impact minim sau nul
2	deficit minor de performanță; aceeași abordare	activități suplimentare; termenele-cheie se respectă	creștere a bugetului sau a costului unitar < 1%
3	deficit moderat; există soluții alternative	întârziere minoră; unele termene vor fi ratate	creștere < 5%
4	inacceptabil, dar există soluții alternative	drumul critic al proiectului este afectat	creștere < 10%
5	inacceptabil; nu există alternative	reperle-cheie ale proiectului nu pot fi atinse	creștere > 10%

7.5 Atenuarea riscurilor

Atenuarea înseamnă

- reducerea **probabilității**
- reducerea **impactului**
- sau ambele

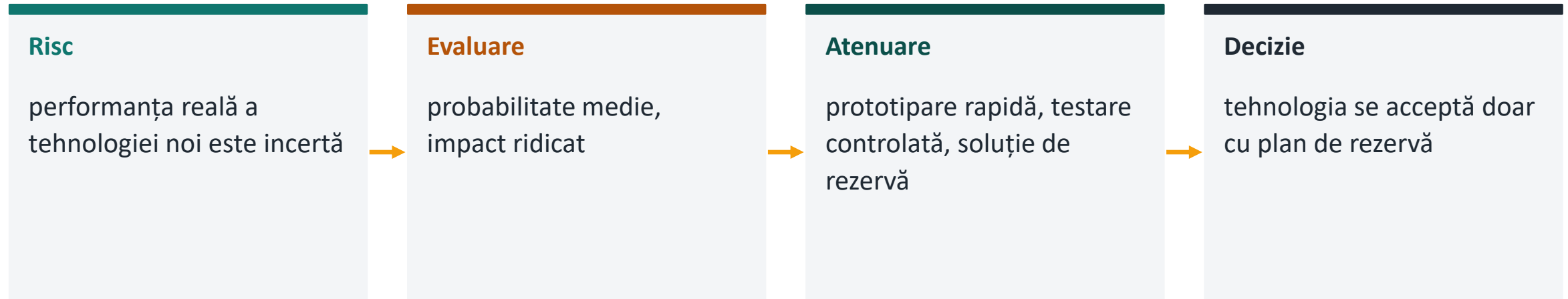
Fiecare risc semnificativ are

- un **responsabil**
- un **plan de acțiune**
- un **criteriu de monitorizare**

Metode uzuale de atenuare [1, cap. 5]

Metodă	Ce face
Revizii intensificate	focalizate pe elementele critice
Supraveghere specială	ingineri seniori, consultanți externi
Analize și teste suplimentare	pentru validarea soluțiilor riscante
Prototipare rapidă	reduce incertitudinea tehnologică
Cerințe conservatoare	și alternative de rezervă

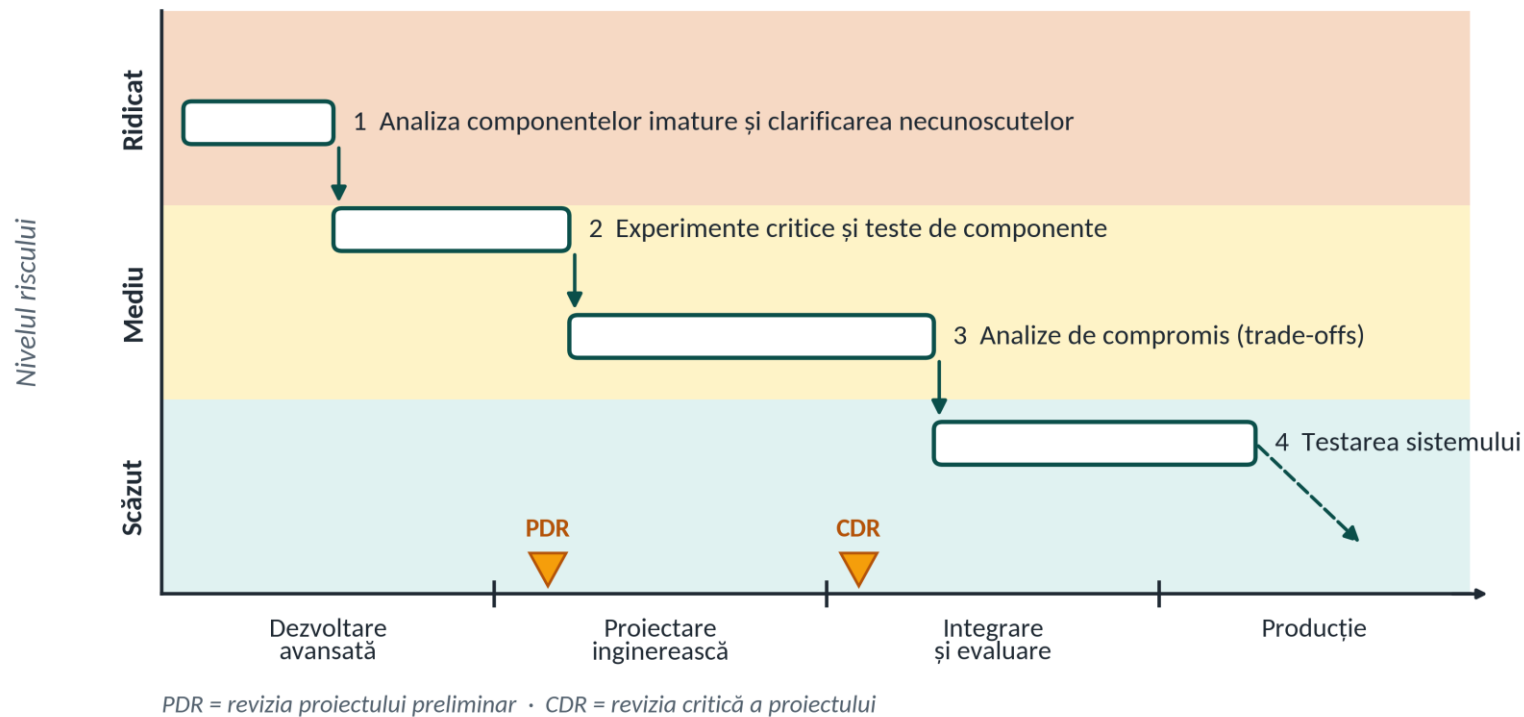
Exemplu: risc tehnologic, de la evaluare la decizie



- Decizia de sistem nu este „acceptăm” sau „respingem” tehnologia, ci o **acceptare condiționată**: planul de rezervă limitează impactul dacă riscul se materializează.
- Metodele de atenuare costă puțin dacă sunt aplicate devreme și foarte mult dacă sunt aplicate târziu.

Un risc fără plan de atenuare este doar o problemă amânată.

Diagrama în cascadă de atenuare a riscului



- Diagrama arată nivelul de risc **planificat** în timp: fiecare activitate finalizată la termen coboară riscul într-o zonă inferioară [1, cap. 5].
- Reviziile PDR (revizia proiectului preliminar) și CDR (revizia critică a proiectului) sunt puncte de control în care se verifică dacă riscul a coborât conform planului.

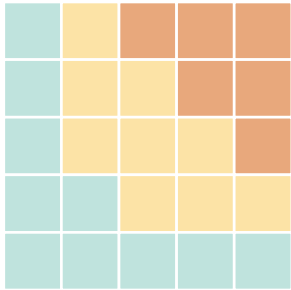
Fig. 7.2. Diagrama în cascadă de atenuare a riscului. Sursa: adaptat după [1, cap. 5].

7.6 Planul de management al riscului

- Planul de management al riscului este un document formal, care cuprinde [1, cap. 5]:
 - lista riscurilor și evaluarea lor;
 - măsurile de atenuare și responsabilitățile;
 - starea curentă a fiecărui risc.
- Este un **document dinamic**, actualizat pe parcursul proiectului, și face parte din SEMP și din planul de management al proiectului (Capitolul 5).
- Forma practică a planului este **fișa de lucru**: un risc = o fișă (pagina următoare).

Scopul nu este eliminarea riscurilor, ci reducerea lor la un nivel acceptabil.

Fișa de lucru pentru planul de risc

Titlul riscului: _____ Proprietar: _____	Proiect: _____ Actualizat: _____				
Descrierea riscului: _____ Cauza de bază: _____ Consecința, dacă riscul se produce: _____ Tip: ☐ tehnic ☐ termen ☐ cost ☐ altul					
 <i>plasați riscul în matrice</i>					
Planul de reducere a riscului					
A acțiuni / reper	Termen	Criteriu de succes	Nivel P	Nivel C	Obs.
1. _____					
2. _____					
3. _____					

- Fiecare risc are propria fișă: titlu și proprietar; descriere, cauză și consecință; tipul riscului; planul de reducere, cu acțiuni, termene și criterii de succes.
- După fiecare acțiune se notează noul nivel al riscului (P = probabilitate, C = consecință), plasat direct în matricea 5 × 5.

Fig. 7.3. Fișa de lucru pentru planul de risc. Sursa: adaptat după [1, cap. 5].

Riscul rezidual și decizia de continuare

- Chiar și după atenuare rămâne un **risc rezidual**: riscul care persistă după aplicarea măsurilor [13]. Conducerea decide dacă riscul rezidual este acceptabil sau dacă proiectul trebuie ajustat.
- Acceptarea riscului rezidual este o decizie **explicită și documentată**, nu o omisiune.

Decizia este

- tehnică · managerială · strategică

Ingineria sistemelor furnizează

- informații · argumente · scenarii

Ingineria sistemelor nu elimină riscul, ci permite decizii asumate.

7.7 Studiu de caz: controlul traficului aerian (ATC)

- Controlul traficului aerian este un sistem critic pentru siguranță, care funcționează în timp real și integrează:
 - senzori (radar, ADS-B, prin care aeronava își transmite singură poziția);
 - software de procesare și decizie;
 - operatori: controlorii de trafic;
 - infrastructură de comunicații.
- Este un **system-of-systems**: sisteme independente a căror performanță globală depinde de integrare. Volumul de date este mare, deciziile sunt rapide, iar toleranța la erori este extrem de redusă.



Fig. 7.4. Centru de control al traficului aerian. Sursa: <https://aerocorner.com/blog/why-is-cockpit-called-cockpit/>

Riscurile sistemului ATC și atenuarea lor

Tehnic

erori de procesare, latență
în transmiterea datelor

De integrare

incompatibilități între
senzori, software și
infrastructură

Operațional

supraîncărcarea
controlorilor la trafic
intens

Interfață om-mașină

afișare neclară sau
incompletă a informațiilor
critice

Atenuare

validare timpurie

redundanță

testare realistă

interfață om-mașină atent
proiectată

Toate cele patru categorii de risc apar simultan; un astfel de sistem trebuie analizat din punctul de vedere al riscurilor încă din faza de concept.

Concluzii

- 1 Riscurile se clasifică în tehnice, de integrare, programatice și operaționale.
- 2 Riscurile se identifică devreme, de întreaga echipă, din surse cunoscute.
- 3 Matricea și cubul de risc prioritizează: probabilitate × impact.
- 4 Atenuarea are responsabil, plan și criteriu de monitorizare; riscul rezidual se acceptă explicit.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Definiți cele două componente ale managementului riscului (evaluarea și atenuarea), cu câte două exemple. Aplicați-le pe un proiect cu tehnologii al căror succes nu este asigurat.

Tema 2

Echipa nu a mai dezvoltat subsistemele necesare. Ce activități de atenuare recomandați? Reduce fiecare probabilitatea, consecințele sau ambele?

Tema 3

Documentați-vă despre Tunelul de sub Canalul Mânecii. Care au fost riscurile principale și ce măsuri au dus la succes?

Recapitularea Părții I

Cap. 1

Gândire de ansamblu, nu optimizarea componentelor individuale.

Cap. 2

Ingineria sistemelor este un rol de maturitate profesională.

Cap. 3

Sistemele se analizează structural și în relație cu mediul.

Cap. 4

Interfețele sunt principalele surse de risc.

Cap. 5

Managementul ingineriei sistemelor completează managementul de proiect.

Cap. 6

Riscul = efectul incertitudinii asupra obiectivelor; se reduce cel mai ieftin devreme.

Cap. 7

Identificare, evaluare, atenuare, monitorizare: un proces continuu.

Dacă uitați restul, rețineți: un sistem nu este suma componentelor · problemele apar la interacțiuni · riscurile apar devreme și se gestionează devreme.

PARTEA II

Dezvoltarea conceptului și dezvoltarea inginerescă

Analiza nevoilor, explorarea și definirea conceptului, arhitectura sistemului, dezvoltarea inginerescă, integrarea și testarea

CAPITOLUL 8

Analiza nevoilor și analiza operațională

Cuprinsul capitolului

8.1 Etapa de dezvoltare a conceptului

8.2 Analiza nevoilor

8.3 Originea unui sistem nou

8.4 Locul analizei nevoilor în ciclul de viață

8.5 Starea de materializare: vizualizare, nu proiectare

8.6 Cele patru activități ale analizei nevoilor

8.7 Analiza operațională: scenarii și obiective

8.8 Exemplu: controlul accesului într-o clădire

Concluzii · Teme de aplicare

8.1 Etapa de dezvoltare a conceptului

- După identificarea și evaluarea riscurilor, proiectul intră în **etapa de dezvoltare a conceptului** [1, cap. 4]. Scopul ei este:
 - înțelegerea clară a nevoilor;
 - formularea cerințelor;
 - pregătirea explorării de soluții.
- Este o etapă timpurie, critică și cu impact major asupra deciziilor ulterioare.
- Analiza nevoilor este ea însăși un instrument de reducere a riscului: reduce riscul de a dezvolta un sistem care nu răspunde nevoilor (Capitolul 6, Fig. 6.2).

Un concept bun începe cu o analiză corectă a nevoilor.

8.2 Analiza nevoilor

Analiza nevoilor răspunde la trei întrebări fundamentale [1, cap. 6]:

Ce problemă trebuie rezolvată?

Pentru cine este dezvoltat sistemul?

În ce condiții va fi utilizat?

- Nevoile sunt, inițial, formulate la nivel operațional, în limbajul utilizatorului, nu al inginerului. Ele sunt adesea **incomplete sau ambigue** și devin surse majore de risc dacă nu sunt clarificate.
- Scopul analizei nevoilor este transformarea lor în cerințe **clare, verificabile și realizabile**. Traducerea este graduală și se întinde pe capitolele 8–10.

Un sistem bine proiectat nu începe cu soluția, ci cu nevoia reală.

8.3 Originea unui sistem nou

- Obiectivul principal al fazei este demonstrarea unei **nevoi operaționale reale** sau a unei **oportunități de piață** [1, cap. 6]. O idee nu este suficientă; trebuie demonstrate o abordare fezabilă, un cost acceptabil și un nivel de risc controlabil.
- Rezultatul fazei este o „viziune a succesului” pentru factorii de decizie: ce problemă se rezolvă, cu ce beneficiu, la ce cost și cu ce risc.

Nevoi operaționale (need pull)

- deficiențe grave ale sistemului actual
- risc principal: se poate satisface nevoia?*

Evoluții tehnologice (technology push)

- o tehnologie nouă permite un salt de performanță
- risc principal: maturitatea tehnologiei*

Ambele surse pot conduce la sisteme complet noi sau la modernizări majore; originea nevoii influențează riscurile și strategia de dezvoltare.

Nevoi impuse de reglementări și de tehnologie

Firul roșu: nevoi impuse de reglementări

- reglementările cer reducerea consumului, creșterea siguranței și reducerea poluării
- nu pot fi satisfăcute prin optimizări minore; cer proiectare inovatoare
- nevoia este externă, clar formulată și tehnic dificil de satisfăcut

ex.: ținta UE 2035 din Capitolul 1

Sisteme determinate de evoluția tehnologică

- tehnologii spațiale: sateliți de comunicații, navigație, observație
- informatizare: sisteme bancare, ticketing, logistică
- nevoia provine din potențialul de îmbunătățire, nu dintr-o deficiență

Risc tipic al sistemelor determinate de tehnologie: **supraestimarea maturității tehnologiei**. Maturitatea se evaluează adesea pe scala TRL (*Technology Readiness Level*, 1–9) [4].

8.4 Locul analizei nevoilor în ciclul de viață

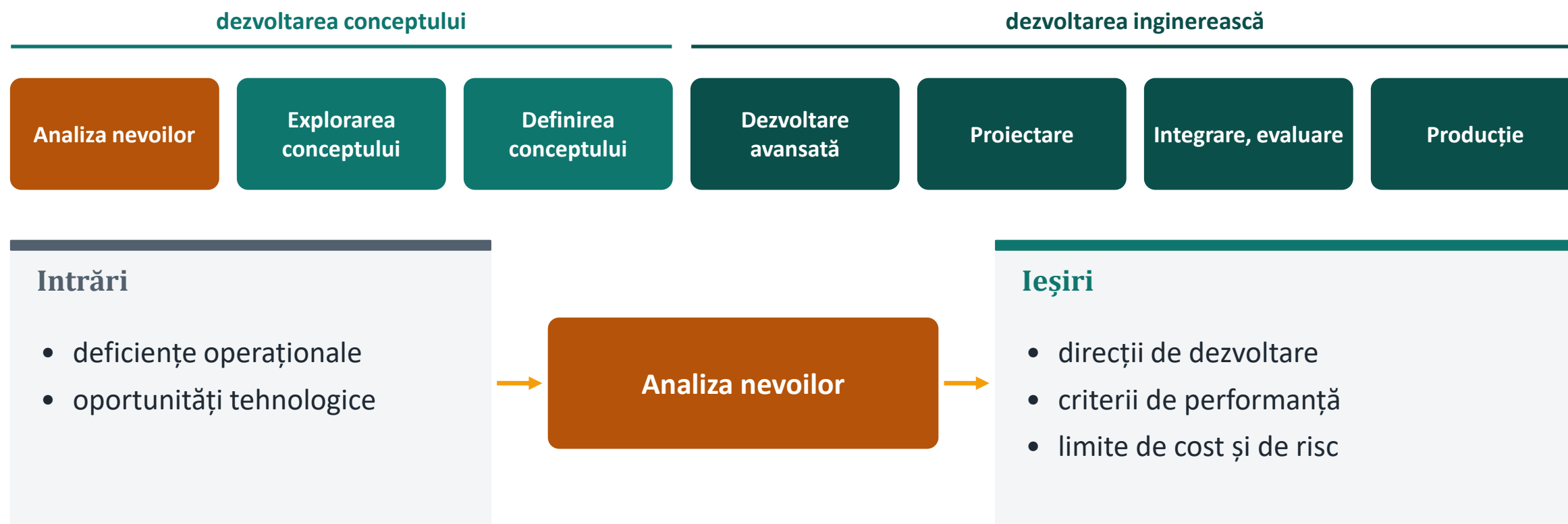


Fig. 8.1. Locul analizei nevoilor în ciclul de viață. Sursa: adaptat după [1, cap. 6].

Analiza nevoilor este prima etapă decizională majoră: la finalul ei se decide dacă proiectul continuă spre explorarea conceptului.

8.5 Starea de materializare: vizualizare, nu proiectare

- „Starea de materializare” arată cât de concret este definit sistemul într-o anumită fază [1, cap. 4]. În analiza nevoilor, accentul este pe obiective operaționale, nu pe soluții tehnice: sistemul este definit la nivelul capabilităților și al eficacității, iar elementele de detaliu sunt doar **vizualizate**, de regulă prin analogie cu sisteme existente.

Tabelul 8.1. Starea de materializare a sistemului pe etape și niveluri. Sursa: adaptat după [1, cap. 4].

Nivel	Analiza nevoilor	Explorarea conceptului	Definirea conceptului	Dezvoltare avansată	Proiectare inginerescă	Integrare și evaluare
Sistem	definește capabilitățile și eficacitatea	identifică, explorează și sintetizează concepte	definește conceptul selectat, cu specificații	validează conceptul		testează și evaluează
Subsistem		definește cerințele și asigură fezabilitatea	definește arhitectura funcțională și fizică	validează subsistemele		integrează și testează
Componentă			alocă funcțiile componentelor	definește specificațiile	proiectează și testează	integrează și testează
Subcomponentă		vizualizează		alocă funcțiile subcomponentelor	proiectează	
Piesă					fabrică sau cumpără	

Detalierea prematură crește riscul și reduce flexibilitatea.

8.6 Cele patru activități ale analizei nevoilor



Analiza operațională

Capitolul 8



Analiza funcțională

Capitolul 9



Definirea cerințelor

Capitolul 10



Validarea cerințelor

Capitolul 10

- Analiza nevoilor cuprinde patru activități succesive, cu bucle de reacție între ele [1, cap. 6]. Diagrama completă este prezentată în secțiunea 10.7.
- Toate patru urmăresc în această lucrare același exemplu: **controlul accesului într-o clădire**.

8.7 Analiza operațională

- Analiza operațională este prima activitate a analizei nevoilor. Scopul ei este înțelegerea modului în care sistemul va fi utilizat și identificarea **obiectivelor operaționale** [1, cap. 6].
- Ea nu descrie soluții tehnice, ci problema operațională: răspunde la „cum trebuie să funcționeze sistemul în realitate”, nu la „cum îl construim”.

Misiuni și scenarii

de utilizare

Utilizatori

și rolurile lor

Obiective operaționale

măsurabile

Fără analiză operațională clară, cerințele vor fi incomplete sau contradictorii, adică prima sursă tipică de risc din Capitolul 7.

Scenariile operaționale

Un scenariu descrie

- **cine** utilizează sistemul
- **în ce condiții** îl utilizează
- **ce rezultat** urmărește

Tipuri de scenarii

- normale
- situații limită
- avarie, urgență

- Scenariile ajută la identificarea nevoilor reale și la descoperirea timpurie a riscurilor [1, cap. 6].
- Scenariile de avarie sunt cele mai des omise și, în același timp, cele care scot la iveală cele mai multe cerințe critice pentru siguranță.

Dacă nu poți descrie un scenariu, încă nu înțelegi nevoia.

Obiectivele operaționale

- Obiectivele operaționale exprimă ce trebuie să realizeze sistemul, din punctul de vedere al utilizatorului. Ele răspund la întrebarea „**de ce este necesar sistemul?**” [1, cap. 6].
- Trebuie să fie clare, verificabile și exprimate în termeni de performanță: timp de răspuns, capacitate, acuratețe, disponibilitate.
- În această fază sunt adesea calitative și încep, de regulă, cu „să asigure...” sau „să ofere...”. Se cuantifică treptat, pe măsură ce devin cerințe (Capitolul 10).
- Analiza operațională fundamentează analiza funcțională și evaluarea fezabilității. Greșelile din această etapă se propagă în toate fazele următoare.

Obiectivele operaționale nu sunt soluții, ci criterii de succes. Erorile de concept sunt cele mai scumpe.

8.8 Exemplu: controlul accesului într-o clădire

- Exemplul comun al capitolelor 8–10 este sistemul de control al accesului într-o clădire de birouri cu acces restricționat.
- Utilizatori: **angajați, vizitatori și personal de securitate**.
- Sistemul funcționează zilnic, în condiții normale și de urgență, cu impact direct asupra siguranței.
- Pe acest sistem facem analiza operațională (aici), analiza funcțională (Capitolul 9), cerințele și validarea (Capitolul 10).

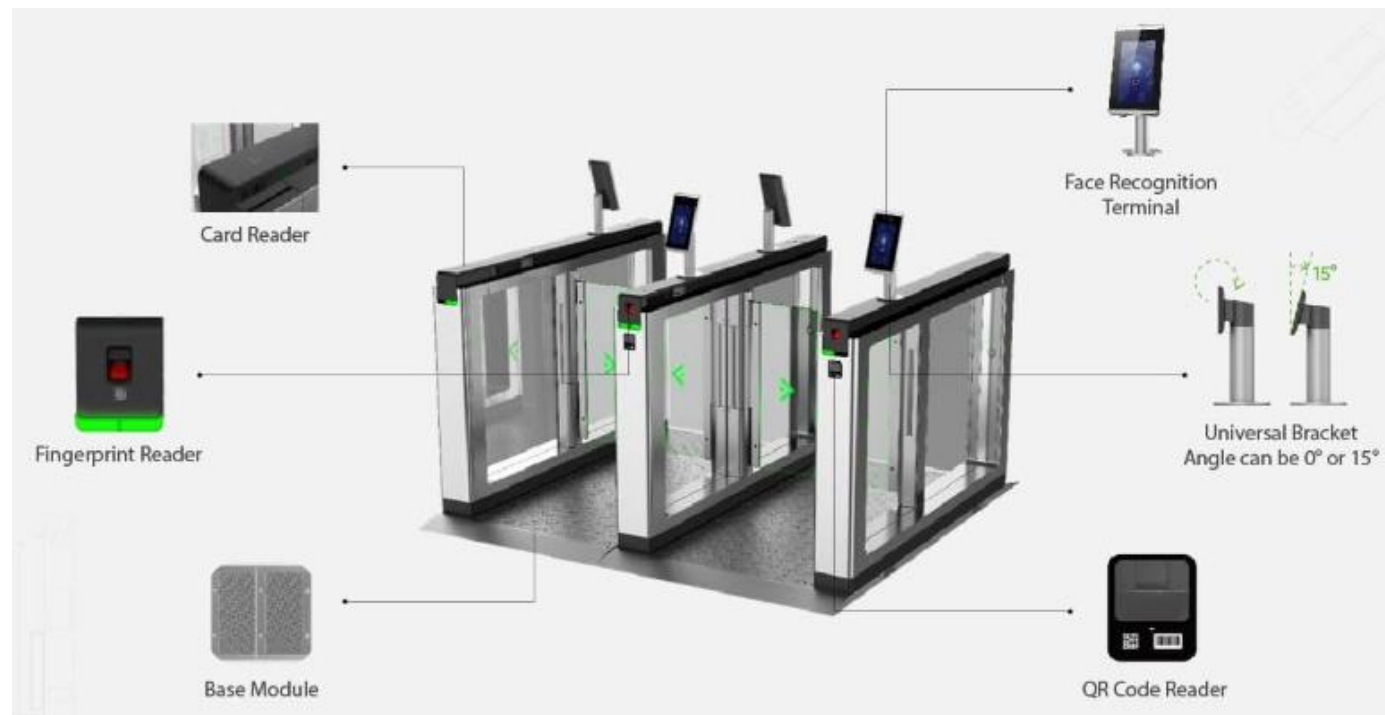


Fig. 8.2. Intrare cu cititoare de acces și turnichete. Sursa: [https://www.hikvision.com/ro/products/Access-Control-Products/Turnstiles/].

Exemplu: trei scenarii operaționale

1 · Acces angajat

Un angajat ajunge la intrare și solicită acces. Dacă este autorizat, accesul trebuie să fie rapid.

2 · Acces vizitator

Vizitatorul se prezintă la recepție. Accesul este permis doar după înregistrare și aprobare, limitat în timp și spațiu.

3 · Situație specială

Urgență sau incident de securitate: accesul trebuie controlat diferit (blocare sau evacuare).

Niciun scenariu nu menționează carduri, cititoare sau turnichete: scenariile descriu utilizarea reală, nu soluția tehnică.

Exemplu: obiectivele operaționale

Pe baza scenariilor, sistemul trebuie să asigure:

- 1 acces rapid pentru utilizatorii autorizați
- 2 restricționarea accesului neautorizat
- 3 evidența și monitorizarea evenimentelor de acces
- 4 posibilitatea de intervenție a personalului de securitate
- 5 funcționare sigură în situații speciale

Acestea sunt obiective operaționale: nu funcții și nu soluții. În Capitolul 9 fiecare va fi descompus în funcții ale sistemului.

Concluzii

1

Un sistem nou pornește de la o nevoie reală, nu de la o idee.

2

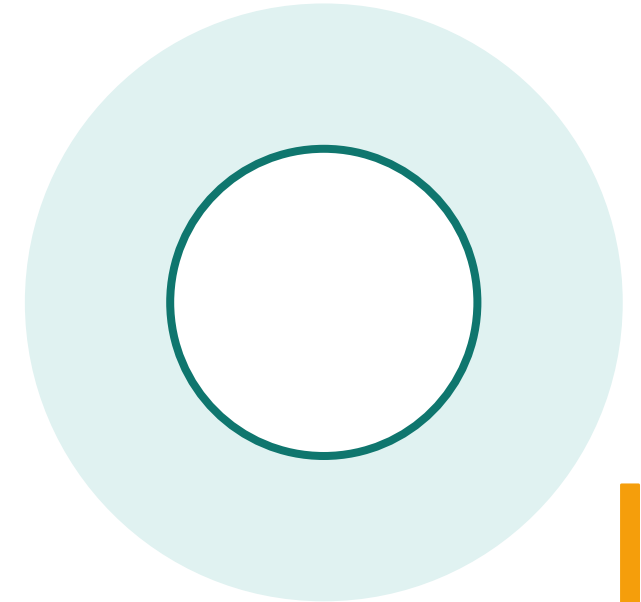
Sursele: nevoi operaționale sau evoluții tehnologice.

3

În analiza nevoilor, sistemul se vizualizează, nu se proiectează.

4

Analiza operațională produce scenarii și obiective operaționale.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Pentru sistemul ales în Capitolul 3: are originea într-o nevoie operațională sau într-o oportunitate tehnologică? Scrieți trei scenarii: normal, limită și de avarie.

Tema 2

Formulați patru obiective operaționale pentru același sistem. Verificați că niciunul nu conține o soluție tehnică.

CAPITOLUL 9

Analiza funcțională

Cuprinsul capitolului

9.1 Ce este analiza funcțională

9.2 De ce este necesară

9.3 Funcția sistemului: verb + obiect

9.4 De la obiective operaționale la funcții

9.5 Verificarea acoperirii

9.6 Analiza funcțională și riscul

Concluzii · Teme de aplicare

9.1 Ce este analiza funcțională

- Analiza funcțională este a doua activitate a analizei nevoilor (B). Este etapa în care trecem de la nevoi și obiective la **ce trebuie să facă sistemul** [1, cap. 6].
- Ea descrie funcțiile sistemului **independent de soluția tehnică** și se situează între problemă (analiza operațională) și soluție (arhitectura).
- Întrebarea la care răspunde: „Ce face sistemul pentru a satisface nevoile identificate?”
- Punctul de plecare sunt scenariile și obiectivele operaționale din Capitolul 8.

Funcțiile nu spun CUM, ci CE face sistemul.

9.2 De ce este necesară analiza funcțională

Fără analiză funcțională

- se trece prea repede la soluții
- apar funcții lipsă sau redundante
- crește riscul de integrare

Cu analiză funcțională

- gândirea este structurată
- se creează baza cerințelor
- se pot compara mai multe soluții

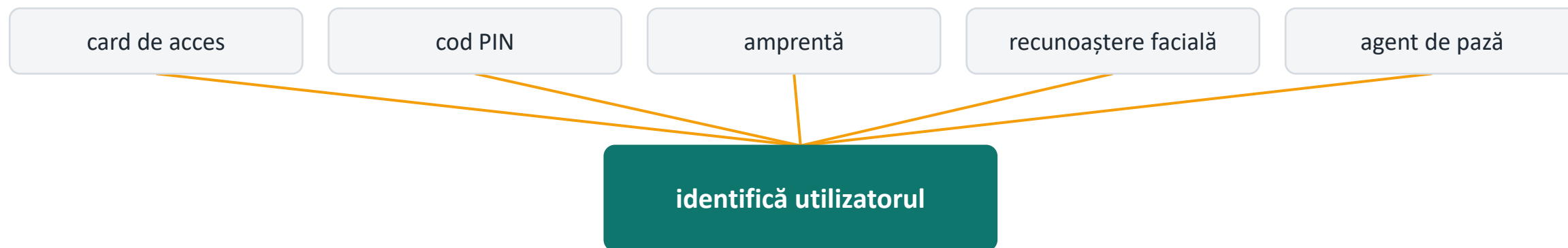


Fig. 9.1. Aceeași funcție, mai multe soluții posibile.

9.3 Funcția sistemului: verb + obiect

O funcție este o acțiune pe care sistemul trebuie să o realizeze, formulată ca **verb + obiect**, independentă de implementare [1, cap. 6], [3].

Corect

- „permite accesul”
- „verifică autorizarea”
- „înregistrează evenimentele”

Greșit

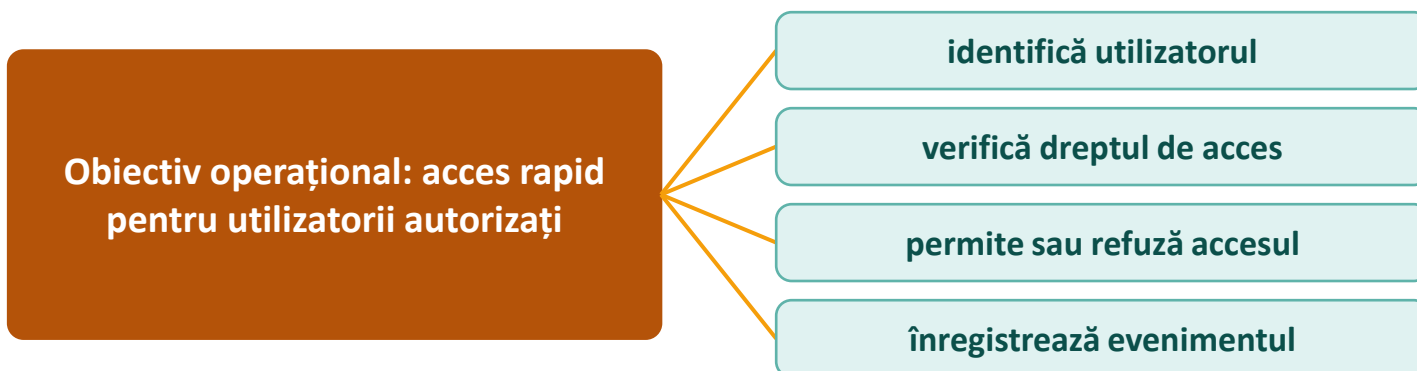
- „folosește card RFID” → identifică utilizatorul
- „afișează pe monitor” → informează operatorul

- „Folosește card RFID” nu este o funcție, ci o tehnologie: o soluție de implementare introdusă prea devreme. Formulată corect, același lucru lasă deschise mai multe soluții.

Regulă practică: dacă în funcție apare o tehnologie, este prea devreme.

9.4 De la obiective operaționale la funcții

Analiza funcțională descompune scenariile, utilizatorii și obiectivele în funcții. **Un obiectiv fără funcții asociate nu va fi realizat în sistem.**



Scenariu: acces vizitator

- înregistrează datele vizitatorului
- solicită aprobarea accesului
- limitează accesul în timp și spațiu
- monitorizează prezența

Scenariu: urgență sau incident de securitate

- detectează situația specială
- modifică regulile de acces
- permite evacuarea controlată
- informează personalul de securitate

- Funcțiile pot fi derivate din obiective și direct din scenarii; cele două căi se verifică reciproc.
- Funcțiile pentru situații speciale sunt adesea uitate dacă lipsește analiza funcțională.

9.5 Verificarea acoperirii: de la obiective la funcții

Tabelul 9.1. Tabelul obiective–funcții pentru controlul accesului.

Obiectiv operațional	Funcții asociate
Acces rapid pentru utilizatorii autorizați	identifică utilizatorul · verifică dreptul de acces · permite sau refuză accesul
Restricționarea accesului neautorizat	verifică dreptul de acces · limitează accesul în timp și spațiu
Evidența evenimentelor de acces	înregistrează evenimentele · monitorizează prezența vizitatorilor
Intervenția personalului de securitate	informează personalul · modifică regulile de acces
Funcționare sigură în situații speciale	detectează situația specială · permite evacuarea controlată

- Tabelul este o formă simplă de **trasabilitate**. Un rând fără funcții semnalează un obiectiv care nu va fi realizat; o funcție care nu apare în niciun rând poate fi redundantă.
- Aceeași funcție („verifică dreptul de acces”) poate servi mai multe obiective.

9.6 Analiza funcțională și riscul

Identifică funcțiile lipsă

din timp, nu la integrare

Clarifică cerințele

ce trebuie să facă sistemul, înainte de proiectare

Reduce riscurile

de integrare, conflictele între cerințe, modificările târzii

- Analiza funcțională acționează asupra primei surse de risc din Capitolul 7: cerințele incomplete, ambigue sau contradictorii.
- Funcțiile identificate devin baza cerințelor funcționale (Capitolul 10), pot fi **alocate subsistemelor** (legătura cu structura funcțională și fizică din Capitolul 3) și vor fi verificate și validate.
- Abia după această etapă se discută soluțiile tehnice și se propune arhitectura sistemului.

Dacă sărim peste analiza funcțională, plătim mai târziu.

Concluzii

1

Funcțiile spun CE face sistemul, nu CUM.

2

O funcție = verb + obiect, fără tehnologie.

3

Fiecare obiectiv operațional trebuie acoperit de funcții.

4

Soluțiile și arhitectura vin după analiza funcțională.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Pentru sistemul ales, descompuneți obiectivele operaționale formulate în Capitolul 8 în funcții. Construiți tabelul obiective–funcții.

Tema 2

Alegeți două dintre funcții. Pentru fiecare, propuneți trei soluții tehnice diferite. Ce criterii ați folosi pentru a alege?

CAPITOLUL 10

Cerințe funcționale și validarea lor

Cuprinsul capitolului

10.1 Ce este o cerință

10.2 Caracteristicile unei cerințe bune

10.3 Formularea cerințelor

10.4 Verificarea cerințelor și trasabilitatea

10.5 Validarea cerințelor

10.6 Decizia de continuare

10.7 Sinteza: diagrama analizei nevoilor

Concluzii · Teme de aplicare

10.1 Ce este o cerință

O **cerință** este o afirmație clară care descrie ce trebuie să facă sistemul, în ce condiții, astfel încât să poată fi verificată [15]. Provine din nevoi, obiective operaționale și funcții.

O cerință NU este

- o soluție tehnică
- o preferință personală
- o descriere vagă

Cerințele

- ghidează proiectarea
- stau la baza testării
- definesc succesul sistemului

Cerințele greșite duc la

- produse care nu rezolvă problema
- costuri mari de modificare
- conflicte între părțile interesate

Funcțiile spun CE face sistemul; cerințele spun **CÂT DE BINE** și **ÎN CE CONDIȚII**. Majoritatea riscurilor majore apar din cerințe incomplete, ambigue sau contradictorii.

10.2 Caracteristicile unei cerințe bune



- Caracteristicile corespund criteriilor de calitate a cerințelor din standardul ISO/IEC/IEEE 29148 [15].
- Verificabilitatea este cea mai des încălcată și cea mai ușor de testat: întrebați „cum demonstrez că sistemul îndeplinește cerința?”.

Regulă simplă: dacă nu poți spune cum o verifici, nu este cerință.

10.3 Formularea cerințelor: „Sistemul trebuie să...”

- Se evită termenii vagi („rapid”, „ușor”, „eficient”), referirile directe la tehnologii și exprimările condiționale neclare. Se preferă valori măsurabile și condiții precise de operare.
- În practica internațională, „shall” (trebuie) marchează o cerință obligatorie, iar „should” (ar trebui) o preferință [15]; în română, „trebuie să” are același rol.

Tabelul 10.1. De la formulări slabe la cerințe corecte (controlul accesului).

Formulare slabă	Cerință corectă	Greșeala
Sistemul trebuie să fie rapid.	Sistemul trebuie să permită accesul în maximum 2 s de la solicitare.	termen vag
Sistemul folosește carduri RFID.	Sistemul trebuie să identifice utilizatorul înainte de acordarea accesului.	tehnologie în cerință
Sistemul ar trebui, dacă se poate, să rețină intrările.	Sistemul trebuie să înregistreze toate evenimentele de acces.	formulare condițională

De la funcție la cerință

Funcție: „permite accesul utilizatorilor autorizați”

Sistemul trebuie să permită accesul utilizatorilor autorizați **în timpul programului de lucru**, **în maximum 2 secunde de la solicitare**.

↑ condiție de operare

↑ măsură verificabilă

Cerințe pentru accesul vizitatorilor

- să permită înregistrarea vizitatorilor înainte de acordarea accesului
- să restricționeze accesul la zonele aprobate, pe durata autorizată
- să înregistreze toate evenimentele de acces ale vizitatorilor

Exemplu de corectare: situații speciale

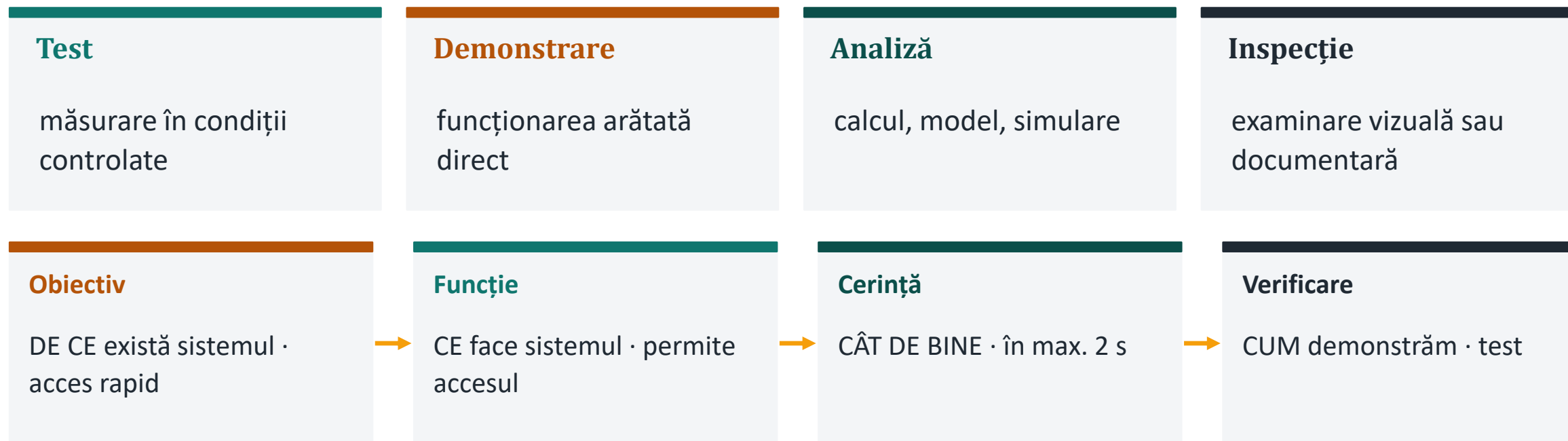
Slab: „să permită evacuarea **rapidă** în caz de urgență”

Corect: „să deblocheze toate ieșirile de urgență în maximum 1 s de la activarea alarmei de incendiu, inclusiv la căderea alimentării principale”

Cerințele reflectă și scenariile critice, nu doar pe cele normale; toate pot fi testate.

10.4 Verificarea cerințelor și trasabilitatea

Fiecare cerință are o metodă de verificare, stabilită odată cu cerința, nu la final [4], [3]:



Trasabilitate: orice cerință se urmărește înapoi până la un obiectiv și înainte până la o metodă de verificare. Cerință fără verificare = risc major.

10.5 Validarea cerințelor

Validarea (activitatea D) verifică dacă cerințele reflectă corect nevoile reale și dacă sistemul propus va rezolva problema [2], [3].

Validare: „Construim sistemul potrivit?”

- raportată la nevoie și la utilizator
building the right system

Verificare: „Construim corect sistemul?”

- raportată la cerințe și la implementare
building the system right

- Un sistem poate trece toate testele și totuși să fie inutil: a fost construit corect, dar nu era sistemul potrivit.
- Fără validare crește riscul de a livra un sistem inutil, iar modificările apar târziu și sunt costisitoare. Validarea reduce **riscurile strategice**, adică riscul ca întregul proiect să fie orientat greșit.

Cum se realizează validarea și ce se validează

Cum se realizează

- scenarii operaționale realiste
- modele analitice sau de simulare
- evaluări cost–beneficiu
- revizii cu părțile interesate
- demonstrarea fezabilității tehnice și economice

Ce se validează

- corectitudinea obiectivelor operaționale (A)
- completitudinea funcțiilor (B)
- realismul cerințelor (C)
- fezabilitatea conceptului de sistem
- raportul cost – performanță – risc

Nu se caută perfecțiunea, ci încrederea rezonabilă în concept. Dacă una dintre componente eșuează, conceptul trebuie revizuit.

Exemplu: validarea în controlul accesului

Permit cerințele...

- acces rapid pentru angajați?
- control eficient al vizitatorilor?
- intervenție în situații speciale?

Sunt cerințele...

- realizabile la costul estimat?
- compatibile cu utilizarea reală?
- acceptabile pentru utilizatori și pentru securitate?

- Întrebările de validare se pun beneficiarului și utilizatorilor, nu echipei de proiectare.

Un singur „nu” înseamnă revizuirea cerințelor.

10.6 Decizia de continuare a dezvoltării

Continuare a dezvoltării

Ajustare a conceptului

Oprire a proiectului

- Validarea susține una dintre cele trei decizii. Decizia este tehnică, economică și strategică; ingineria sistemelor oferă argumente, scenarii și evaluări obiective.
- Etapa de analiză a nevoilor este completă atunci când nevoile sunt înțelese, funcțiile sunt definite, iar cerințele sunt formulate și validate. Rezultatul: un concept de sistem credibil, un nivel de risc acceptabil și baza pentru explorarea conceptului.
- Oprirea unui proiect în această fază nu este un eșec, ci un rezultat valid: costul pierdut este minim față de un sistem livrat și nefolosit.

Este mai bine să oprești un proiect devreme decât să finalizezi un sistem inutil.

10.7 Sinteza: diagrama analizei nevoilor

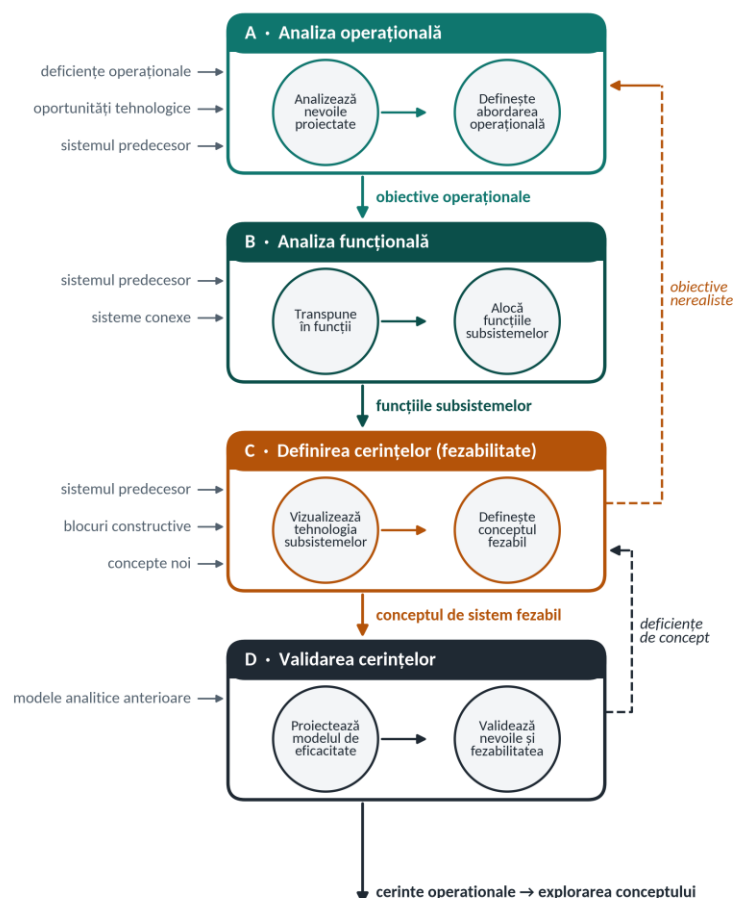


Fig. 10.1. Diagrama analizei nevoilor. Sursa: adaptat după [1, cap. 6].

- Diagrama sintetizează întreaga fază [1, cap. 6]: intrările (deficiențe operaționale, oportunități tehnologice, sistemul predecesor), cele patru activități și ieșirea, care trece în explorarea conceptului.
- Buclele de reacție arată că procesul nu este liniar: obiectivele nerealiste trimit înapoi la analiza operațională, iar deficiențele de concept, la definirea cerințelor.
- Fiecare activitate produce intrarea celei următoare: obiective operaționale → funcții → concept fezabil → cerințe operaționale.

Concluzii

- 1 Cerința spune cât de bine și în ce condiții: „Sistemul trebuie să...”.
- 2 O cerință bună este clară, verificabilă, realizabilă, necesară și independentă de soluție.
- 3 Trasabilitate: obiectiv → funcție → cerință → verificare.
- 4 Validarea întreabă „construim sistemul potrivit?”; verificarea, „îl construim corect?”.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Pentru o nouă aeronavă de pasageri, enumerați trei obiective operaționale și cerințele funcționale necesare pentru fiecare (măsuri calitative).

Tema 2

Pentru sistemul ales, transformați funcțiile din Capitolul 9 în cinci cerințe, fiecare cu metoda de verificare. Formulați două întrebări de validare.

CAPITOLUL 11

Explorarea conceptului de sistem

Cuprinsul capitolului

11.1 Ce este explorarea conceptului

11.2 Locul explorării în ciclul de viață

11.3 De ce este necesară și ce NU este

11.4 Intrări, ieșiri și starea de materializare

11.5 Evaluarea conceptelor și îngustarea spațiului decizional

11.6 Exemple: ATC și automobilul după 2035

11.7 Diagrama explorării conceptului

Concluzii · Teme de aplicare

11.1 Ce este explorarea conceptului

- Explorarea conceptului (*concept exploration*) este etapa în care analizăm **mai multe concepte posibile** de sistem și evaluăm avantajele și limitările fiecăruia [1, cap. 7].
- Un **concept** este o idee coerentă despre organizarea sistemului, fără detalii constructive sau tehnologice. El se situează între funcții (ce face sistemul) și arhitectura detaliată (din ce este făcut).
- Scopul etapei: identificarea conceptelor fezabile și eliminarea din timp a celor cu risc sau cost inacceptabil.
- Tentația naturală a inginerului este să treacă direct la soluția „evidentă”. Cu cerințele validate în mână, răspunsul corect la întrebarea „care este soluția?” este: **încă niciuna**; mai întâi, mai multe soluții candidate.

Nu căutăm soluția perfectă, ci soluții candidate.

11.2 Locul explorării conceptului în ciclul de viață

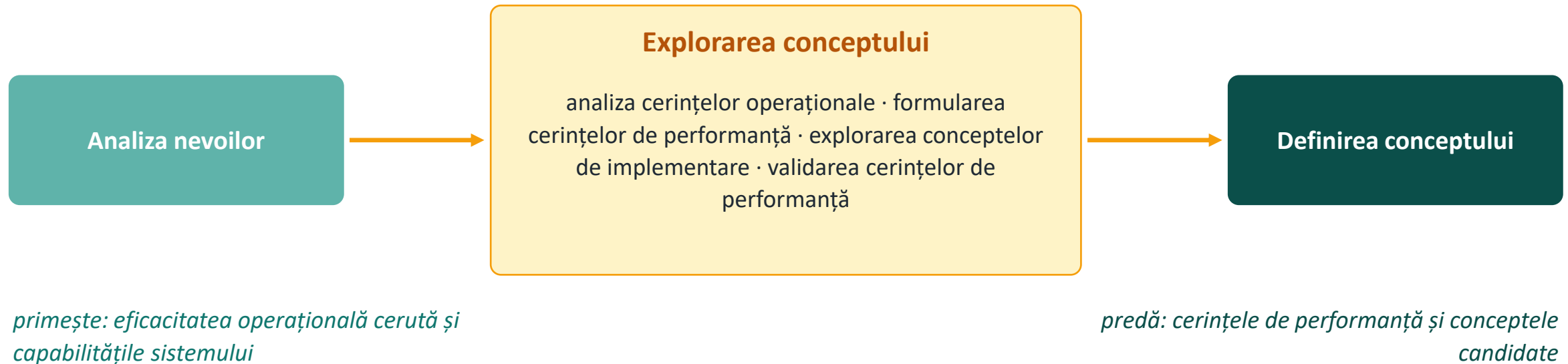


Fig. 11.1. Locul explorării conceptului între analiza nevoilor și definirea conceptului. Sursa: adaptat după [1, cap. 7].

Fiecare fază primește ceva concret de la cea anterioară și predă ceva concret celei următoare.

11.3 De ce este necesară și ce NU este explorarea

De ce este necesară

- aceleași cerințe pot fi satisfăcute prin arhitecturi, niveluri de automatizare și tehnologii diferite
- alegerea directă a unei soluții crește riscul de eșec și limitează optimizarea
- explorarea reduce riscurile timpurii și permite comparația obiectivă

Ce NU este

- proiectare detaliată
- dimensionare de componente
- alegere de furnizori sau de tehnologii concrete

- Accentul este pe **structură**, pe **principii de funcționare** și pe **compromisurile majore**: în această etapă nu desenăm piese, nu scriem cod și nu optimizăm performanțe fine. Este explorarea timpurie, când o decizie greșită se corectează aproape gratuit (Capitolele 5 și 6).

„STOP cognitiv”: dacă în discuție apare prea mult „cum anume”, este prea devreme. Deciziile timpurii greșite sunt cele mai scumpe.

11.4 Intrări, ieșiri și starea de materializare



Tabelul 11.1. Starea de materializare în explorarea conceptului. Sursa: adaptat după [1, cap. 4].

Nivel	Analiza nevoilor	Explorarea conceptului	Definirea conceptului	Dezvoltare avansată	Proiectare inginerescă	Integrare și evaluare
Sistem	definește capabilitățile și eficacitatea	identifică, explorează și sintetizează concepte	definește conceptul selectat, cu specificații	validează conceptul		testează și evaluează
Subsistem		definește cerințele și asigură fezabilitatea	definește arhitectura funcțională și fizică	validează subsistemele		integrează și testează
Componentă			alocă funcțiile componentelor	definește specificațiile	proiectează și testează	integrează și testează
Subcomponentă		vizualizează		alocă funcțiile subcomponentelor	proiectează	
Piesă					fabrică sau cumpără	

11.5 Evaluarea conceptelor

Conceptele se evaluează **comparativ**, nu detaliat, din cinci perspective [1, cap. 7], [1, cap. 9]:

performanță

cost

risc

fezabilitate tehnică

impact operațional

- Un concept bun satisface cerințele principale, are riscuri controlabile și este realist de implementat. Nu trebuie să fie cel mai performant, ci **echilibrat**.
- În practică se folosesc scale calitative (slab, mediu, bun) sau o **matrice de decizie ponderată**, în care fiecare criteriu are o pondere; contează ordinea relativă a conceptelor, nu valorile exacte.
- Un concept slab identificat devreme economisește bani: eliminarea lui este un rezultat, nu un eșec.

Spațiul decizional se îngustează

Idei inițiale · multe, neevaluate

Concepte candidate · coerente, comparabile

Concepte fezabile · după evaluare

1–2 concepte promițătoare · spre definire

Fig. 11.2. Explorarea conceptului îngustează spațiul decizional.

La fiecare pas se elimină opțiunile nefezabile sau cu risc ori cost inacceptabil. Explorarea nu alege soluția finală, ci **reduce spațiul decizional**.

11.6 Exemplu: explorarea conceptului pentru ATC

Sistem automatizat de control al traficului aerian, cu cerințe-cheie: siguranță foarte ridicată, operare în timp real, integrare cu infrastructura existentă (Capitolul 7). Trei concepte, diferite prin **principiul de organizare** (unde se ia decizia), nu prin tehnologii:

Tabelul 11.2. Compararea conceptelor ATC.

Concept	Avantaj principal	Risc principal
A · Control centralizat	simplu de controlat	punct unic de eșec (SPOF, Single Point of Failure)
B · Control distribuit	mai robust	complexitate crescută de integrare
C · Control asistat de AI	potențial mare de performanță	risc ridicat de validare și certificare

Toate conceptele pot satisface cerințele, dar cu profiluri de risc diferite. Explorarea nu alege încă, ci pregătește alegerea.

Firul roșu: automobilul după 2035

Electric pe baterie (BEV)

- + eficiență energetică ridicată
- autonomie, infrastructura de încărcare

Pile de combustie cu hidrogen

- + alimentare rapidă
- infrastructură rară, eficiență globală mai mică

Hibrid reîncărcabil

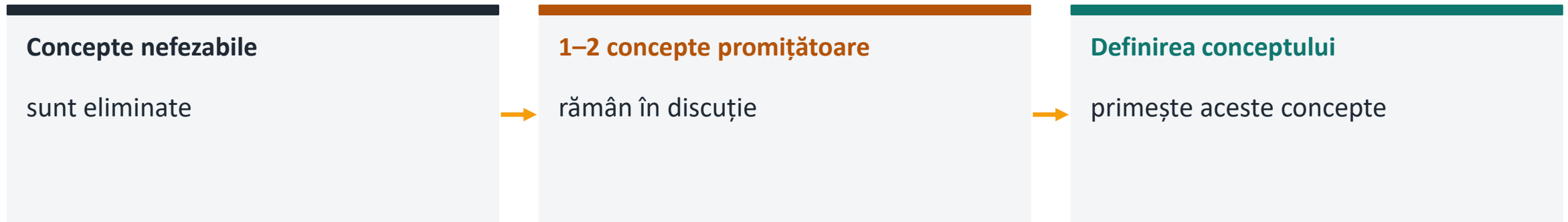
- + flexibilitate în utilizare
- două sisteme de propulsie, emisii nenule

Carburanți sintetici

- + compatibili cu flota existentă
- eficiență globală redusă, cost ridicat

- Aceleași cerințe (emisii reduse la utilizare, autonomie, cost), patru concepte diferite; legătura cu ținta UE 2035 din Capitolul 1 [9], [10] și cu infograficul din Fig. 1.4 [11].
- Evaluările sunt calitative și orientative; o analiză reală ar cere date actualizate de cost și infrastructură. Unele concepte pot fi eliminate pe criteriu **reglementar**, nu doar tehnic.

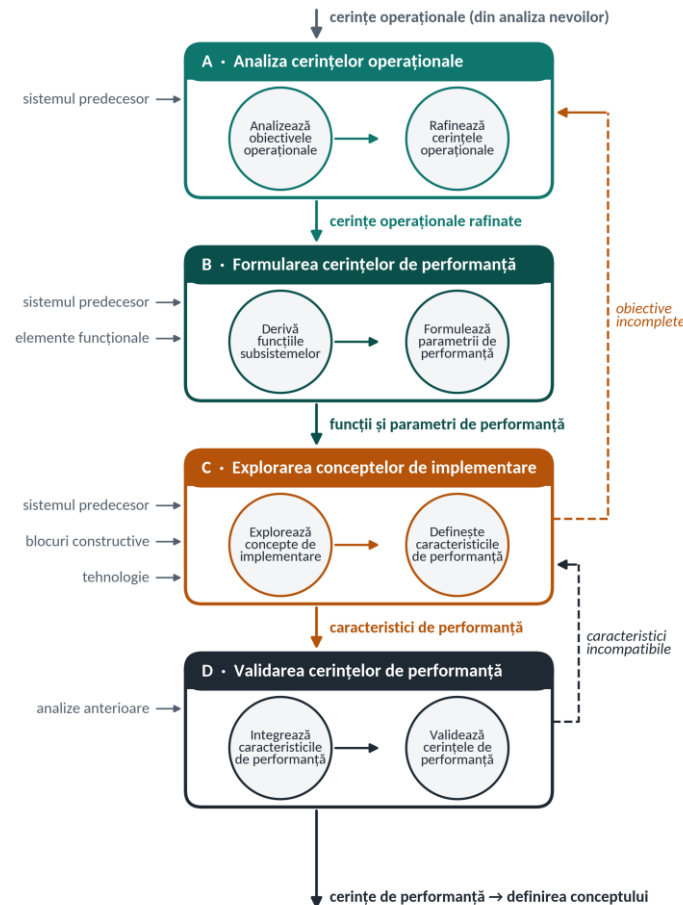
Rezultatul explorării conceptului



- Decizia de a continua nu este doar tehnică: este și **economică** și **strategică**. Bugetul contează, dar și direcția pe care o urmărește organizația.

Explorarea conceptului pregătește decizia, nu o înlocuiește.

11.7 Diagrama explorării conceptului



- Structura A–D repetă logica analizei nevoilor (Fig. 10.1), cu un nivel mai jos [1, cap. 7].
- **A. Analiza cerințelor operaționale** are nevoie de: un set inițial de cerințe operaționale, conceptul operațional al sistemului și contextul operațional (scenariile din Capitolul 8).
- Dezvoltarea cerințelor parcurge patru pași: colectare, analiză, validare, documentare.
- Ieșirea diagramei: **cerințele de performanță**, care intră în definirea conceptului.

Fig. 11.3. Diagrama explorării conceptului. Sursa: adaptat după [1, cap. 7].

Concluzii

- 1 Explorarea analizează mai multe concepte candidate, nu una singură.
- 2 Un concept înseamnă organizare, fără detalii constructive.
- 3 Evaluarea este comparativă: performanță, cost, risc, fezabilitate, impact operațional.
- 4 Rezultatul: 1–2 concepte promițătoare, nu soluția finală.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Mașina electrică: ce componente majore ale automobilului pe benzină s-ar păstra cu modificări minore? Care s-ar modifica substanțial? Care ar fi noi? (Ignorați sistemul audio, cruise control-ul, scaunele, geamurile și airbagurile.)

Tema 2

Pentru o mașină de tuns gazonul, elaborați: (a) un set de cerințe operaționale; (b) un set de cerințe de performanță.

CAPITOLUL 12

Definirea conceptului de sistem

Cuprinsul capitolului

12.1 De la explorare la definire

12.2 Ce înseamnă „concept selectat”

12.3 Ce se definește și ce NU se definește încă

12.4 Exemplu: conceptul ATC selectat

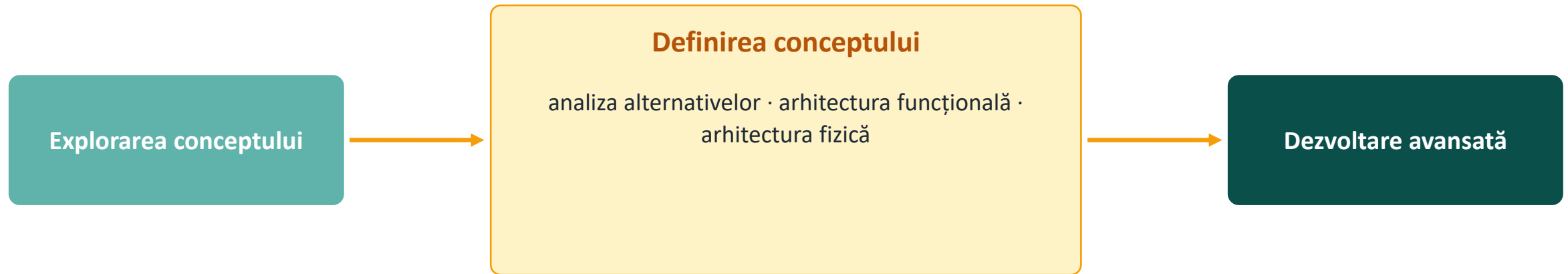
12.5 Firul roșu: automobilul electric

12.6 Starea de materializare și diagrama definirii

12.7 Explorarea vs. definirea conceptului

Concluzii · Teme de aplicare

12.1 De la explorare la definirea conceptului



primește: cerințele de performanță și conceptele candidate

predă: specificațiile funcționale și conceptul definit

Fig. 12.1. Definirea conceptului: ultima fază a dezvoltării conceptului. Sursa: adaptat după [1, cap. 8].

- Definirea conceptului deschide dezvoltarea inginerească [1, cap. 8]. Explorarea v-a lăsat cu două concepte promițătoare; acum **alegeți unul** și îl structurați suficient cât să poată deveni arhitectură.

Explorarea compară. Definirea structurează.

12.2 Ce înseamnă „concept selectat”

Este

- o abordare clară de organizare a sistemului
- capabilă să satisfacă cerințele esențiale
- cu riscuri acceptabile
- realistă de implementat

NU este

- un desen detaliat
- o listă de componente
- o soluție optimizată

- Atributele conceptului selectat corespund criteriilor de evaluare din Capitolul 11: performanță (cerințele esențiale), risc și fezabilitate.

Conceptul trebuie să fie suficient de clar, nu complet.

12.3 Ce se definește și ce NU se definește încă

Se definesc

- principiile de funcționare
- structura de nivel înalt: subsistemele majore
- rolurile principale ale subsistemelor
- interfețele critice
- ipotezele de bază

NU se definesc încă

- tehnologii concrete
- dimensionarea componentelor
- optimizarea performanțelor fine
- decizii ireversibile (contracte ferme, echipamente dedicate)

Ipotezele de bază se documentează explicit (infrastructură disponibilă, nivel de trafic, mediu de operare): o ipoteză nedocumentată care se dovedește falsă devine un risc major. Tot ce se definește aici influențează arhitectura.

12.4 Exemplu: conceptul ATC selectat

- În acest exemplu se selectează **controlul distribuit**, pentru robustețe: elimină punctul unic de eșec al conceptului centralizat, acceptând o integrare mai complexă, tratată prin interfețe critice bine definite.
- Principii: decizie distribuită pe zone, coordonare între centre, suport automat pentru operatorii umani. Nu se definesc încă algoritmi, hardware-ul sau software-ul specific.

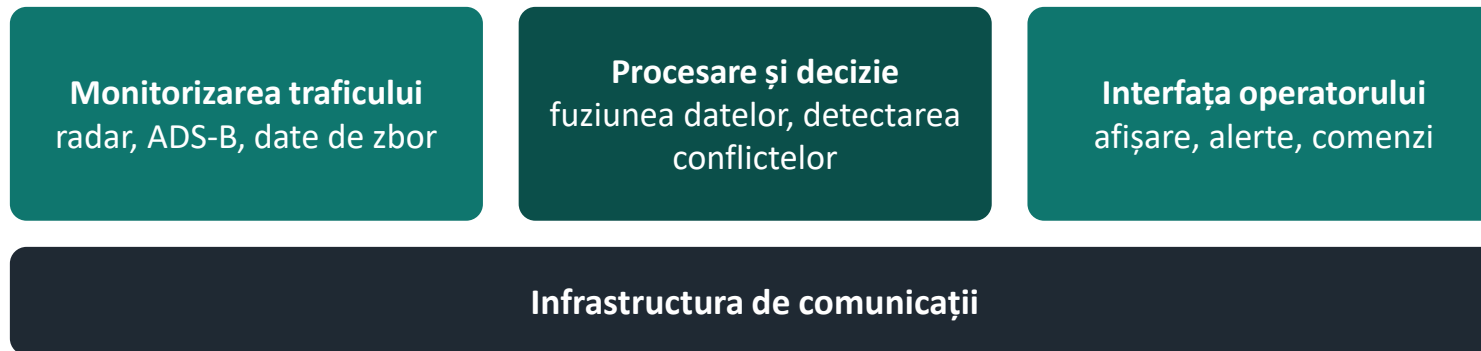


Fig. 12.2. Structura de nivel înalt a conceptului ATC distribuit.

Interfețe critice

- între centrele de control
- om–mașină: controlorul și sistemul de suport
- cu infrastructura externă: aeronave, aeroporturi, rețele de date

12.5 Firul roșu: conceptul automobilului electric

Concept selectat (ilustrativ): **propulsie electrică pe baterie**. Nu se aleg încă chimia bateriei, tipul motorului sau puterea încărcătorului.

Subsisteme majore

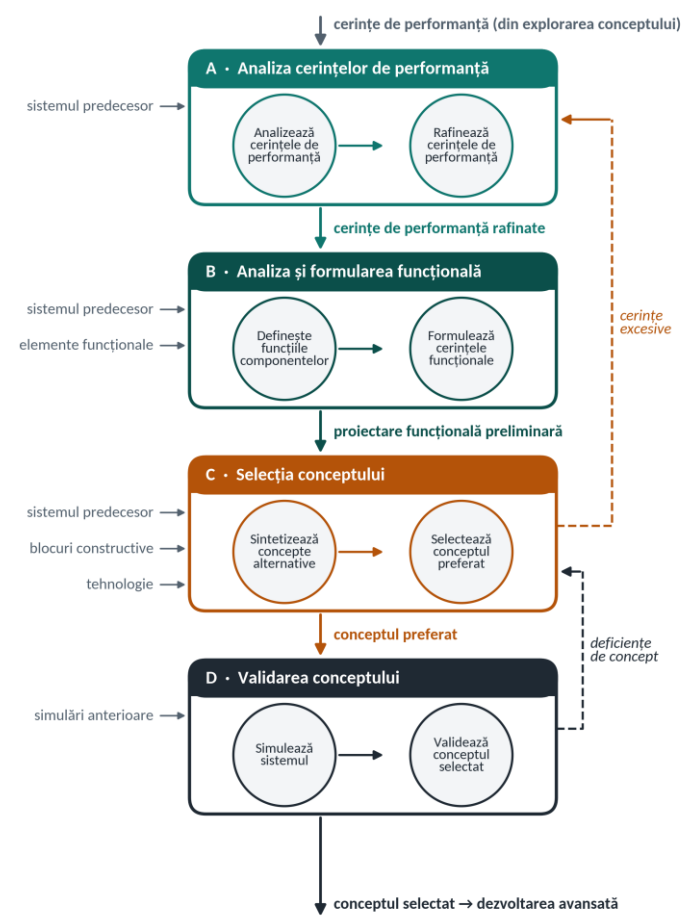
- stocarea energiei
- propulsia electrică
- încărcarea
- managementul energiei și termic

Interfețe critice

- vehicul – rețeaua de încărcare
- baterie – managementul termic
- șofer – vehicul (autonomia afișată)

Ipoteză de bază de documentat: **infrastructura de încărcare este disponibilă pe rutele țintă**. Dacă se dovedește falsă, întregul concept devine riscant.

12.6 Starea de materializare și diagrama definirii



Tabelul 12.1. Materializarea în definirea conceptului [1, cap. 4].

Nivel	Definirea conceptului
Sistem	definește conceptul selectat, cu specificații
Subsistem	definește arhitectura funcțională și fizică
Componentă	alocă funcțiile componentelor

- Pentru prima dată, conceptul ajunge la nivel de componentă: sistemul se materializează de sus în jos, câte un nivel pe fază.
- Activitățile A–D: analiza cerințelor de performanță, analiza și formularea funcțională, **selecția conceptului**, validarea conceptului [1, cap. 8]. Ieșirea: conceptul selectat, care intră în dezvoltarea avansată.

Fig. 12.3. Diagrama definirii conceptului. Sursa: adaptat după [1, cap. 8].

12.7 Explorarea vs. definirea conceptului

Tabelul 12.2. Comparație între explorarea și definirea conceptului.

	Explorarea conceptului	Definirea conceptului
Întrebarea	„Ce opțiuni avem?”	„Ce concept alegem?”
Analizează / stabilește	concepte alternative, compromisuri majore	conceptul selectat, principiile de funcționare
Structura	nu se fixează	structura de nivel înalt, interfețele critice
Rolul	elimină opțiunile nefezabile, reduce spațiul decizional	pregătește sistemul pentru arhitectură

Explorarea compară alternative. Definirea fixează o direcție clară. Un concept bine definit reduce riscul arhitecturii.

Concluzii

1

Conceptul selectat: suficient de clar, nu complet.

2

Se definesc principii, subsisteme, interfețe critice și ipoteze de bază.

3

Nu se aleg încă tehnologii și nu se dimensionează nimic.

4

Explorarea compară; definirea fixează direcția.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Pentru sistemul ales în capitolele anterioare, definiți conceptul selectat: principii de funcționare, structura de nivel înalt, interfețe critice, ipoteze de bază și riscuri cunoscute.

Tema 2

Pentru mașina de tuns gazonul din Capitolul 11, propuneți trei concepte, comparați-le și definiți-l pe cel ales.

CAPITOLUL 13

Arhitectura sistemului

Cuprinsul capitolului

13.1 De la concept la arhitectură

13.2 Arhitectura în ciclul de viață

13.3 Ce este și ce face arhitectura

13.4 Alocarea funcțiilor: exemplul controlului accesului

13.5 Ce NU face arhitectura

13.6 Exemple: ATC și automobilul electric

13.7 Unde se oprește arhitectura

Concluzii · Teme de aplicare

13.1 De la concept la arhitectura sistemului

Am realizat (capitolele 11–12)

- explorarea alternativelor de concept
- definirea conceptului selectat
- structura conceptuală și interfețele majore

Următorul pas

- organizarea sistemului în arhitectură
- alocarea funcțiilor către subsisteme
- definirea sistematică a interfețelor

- Conceptul ales este acum clar, coerent și suficient de stabil pentru detaliere. Putem începe să proiectăm piese? **Încă nu:** mai întâi trebuie stabilit cine face ce și cum comunică. Altfel, echipe excelente pot proiecta subsisteme care nu se potrivesc la integrare.

Arhitectura transformă conceptul într-un sistem realizabil.

13.2 Arhitectura în ciclul de viață

După definirea conceptului, sistemul intră în **dezvoltarea ingineriască**; arhitectura este prima ei etapă, la începutul dezvoltării avansate [1, cap. 4]. Rolul ei: organizarea sistemului, reducerea complexității, pregătirea proiectării detaliate.



Fig. 13.1. Etapa de dezvoltare ingineriască. Sursa: adaptat după [1, cap. 4].

Dezvoltarea ingineriască începe cu arhitectura, nu cu proiectarea.

13.3 Ce este și ce face arhitectura

Arhitectura descrie subsistemele majore, rolul fiecăruia și interfețele dintre ele [1, cap. 8], [3]. Definiția de lucru folosită în continuare: „**cine face ce și cum comunică**”, nu „cum se construiește”.

1

Descompune

sistemul în subsisteme

2

Alocă

funcțiile către subsisteme

3

Identifică

interfețele critice

4

Evaluează

riscurile de integrare

Arhitectura este primul loc în care apar problemele reale: pentru prima dată se vede cum se împart responsabilitățile și unde se pot ciocni.

13.4 Alocarea funcțiilor: exemplul controlului accesului

Tabelul 13.1. Alocarea funcțiilor din Capitolul 9 către subsistemele conceptului distribuit.

Funcție (Capitolul 9)	Subsistem căruia îi este alocată
identifică utilizatorul	Identificare
verifică dreptul de acces	Decizie de acces
permite sau refuză accesul	Acționarea punctelor de acces
înregistrează evenimentele	Monitorizare și jurnalizare
informează personalul de securitate	Monitorizare și jurnalizare
permite evacuarea controlată	Decizie de acces + Acționare

O funcție împărțită între două subsisteme creează o **interfață critică**. Cerința din Capitolul 10 (deblocarea ieșirilor în maximum 1 s, inclusiv la căderea alimentării) se regăsește acum în interfața dintre decizie și acționare.

13.5 Ce NU face arhitectura

nu alege componente hardware

nu definește algoritmi concreți

nu stabilește furnizori

nu implementează soluții

- Este aceeași disciplină a „STOP-ului cognitiv” din capitolele 11–12, aplicată arhitecturii. Toate aceste decizii aparțin proiectării (Capitolul 15).

Dacă intrăm în aceste lucruri, nu mai suntem în arhitectură.

13.6 Exemplu: arhitectura sistemului ATC

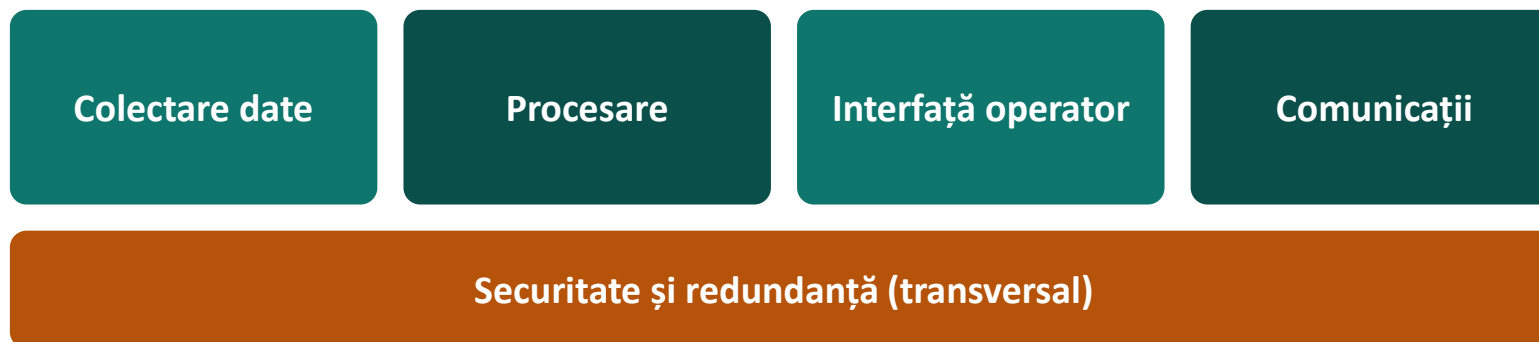


Fig. 13.2. Arhitectura sistemului ATC: cinci subsisteme.

Interfețe critice

- senzori – procesare
- procesare – operator
- sistem – infrastructuri externe

- Față de structura conceptuală din Capitolul 12 apare un subsistem nou, **securitate și redundanță**, care traversează celelalte: o preocupare transversală care apare abia la nivel de arhitectură.
- Sistemul este împărțit pe mari funcții, nu pe tehnologii. Erorile apar cel mai frecvent la schimbul de date între subsisteme (Capitolul 4): interfețele sunt principala sursă de risc.

Firul roșu: arhitectura automobilului electric

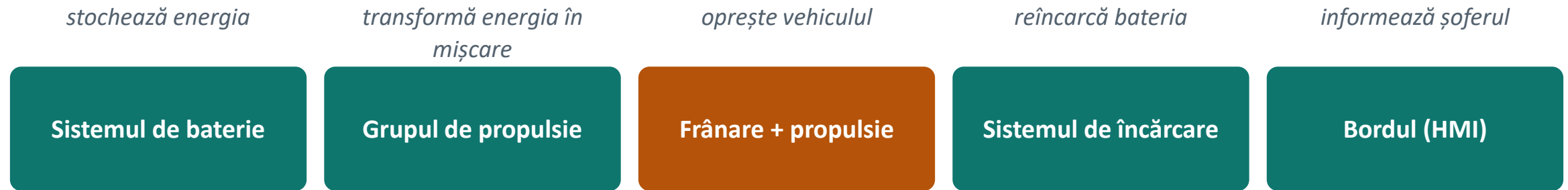


Fig. 13.3. Funcțiile automobilului electric alocate subsistemelor.

- La frânare, motorul electric funcționează ca generator și recuperează energie; frânele mecanice intervin când decelerația cerută depășește ce poate face motorul (**frânare recuperativă**).
- Coordonarea celor două subsisteme este critică pentru siguranță: un exemplu clasic de interfață apărută prin alocarea unei funcții la două subsisteme.

Regula: o funcție alocată la două subsisteme creează o interfață critică.

13.7 Unde se oprește arhitectura

La final AVEM

- subsisteme definite
- funcții alocate
- interfețe identificate
- riscuri cunoscute

NU avem

- componente
- cod
- soluții constructive

- Arhitectura oferă structura pe care se construiesc fazele următoare: dezvoltarea avansată, proiectarea tehnică, integrarea și evaluarea (Capitolele 15–16). Înainte de ele, Capitolul 14 arată cum se **modelează** și se comunică arhitectura.

Arhitectura stabilește structura, nu implementarea.

Concluzii

1

Arhitectura este prima etapă a dezvoltării ingineresti.

2

„Cine face ce și cum comunică”, nu cum se construiește.

3

Arhitectura descompune, alocă funcții, identifică interfețe, evaluează riscuri.

4

Interfețele sunt principala sursă de risc.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Pentru sistemul ales: definiți subsistemele, construiți tabelul funcții–subsisteme și identificați interfețele critice și riscurile de integrare.

Tema 2

Automobilul electric: ce s-ar schimba în arhitectură dacă frânarea ar fi alocată doar sistemului de frânare? Ce câștigați și ce pierdeți?

CAPITOLUL 14

Modelarea arhitecturii: SysML și diagrame

Cuprinsul capitolului

14.1 Rolul SysML în arhitectura sistemului

14.2 Diagramele folosite în arhitectură

14.3 Diagrama de cerințe: trasabilitatea completă

14.4 Diagrama de context: granița sistemului

14.5 Exemplu rezolvat: arhitectură sau proiectare?

14.6 SysML v2: noua generație a limbajului

Concluzii · Teme de aplicare

14.1 Rolul SysML în arhitectura sistemului

- Cum explicați arhitectura unui avion unor sute de ingineri, din zeci de echipe și companii? Nu cu zeci de documente diferite, ci cu un **limbaj comun de modelare**.
- SysML (*Systems Modeling Language*) este un standard al Object Management Group (OMG), derivat din UML și adaptat ingineriei sistemelor [16].

SysML ESTE

- un instrument de modelare
- un limbaj comun între discipline

SysML NU ESTE

- arhitectura în sine
- un limbaj de implementare

În arhitectură, SysML reprezintă cerințele, funcțiile și relațiile dintre subsisteme. **SysML ajută arhitectura, nu o definește**: un instrument excelent nu salvează o arhitectură proastă.

14.2 Diagramele folosite în arhitectură

Diagrama de cerințe

cerințele și relațiile dintre ele (SysML)

Diagrama de context

sistemul și mediul său extern

Diagrama funcțională (FBD)

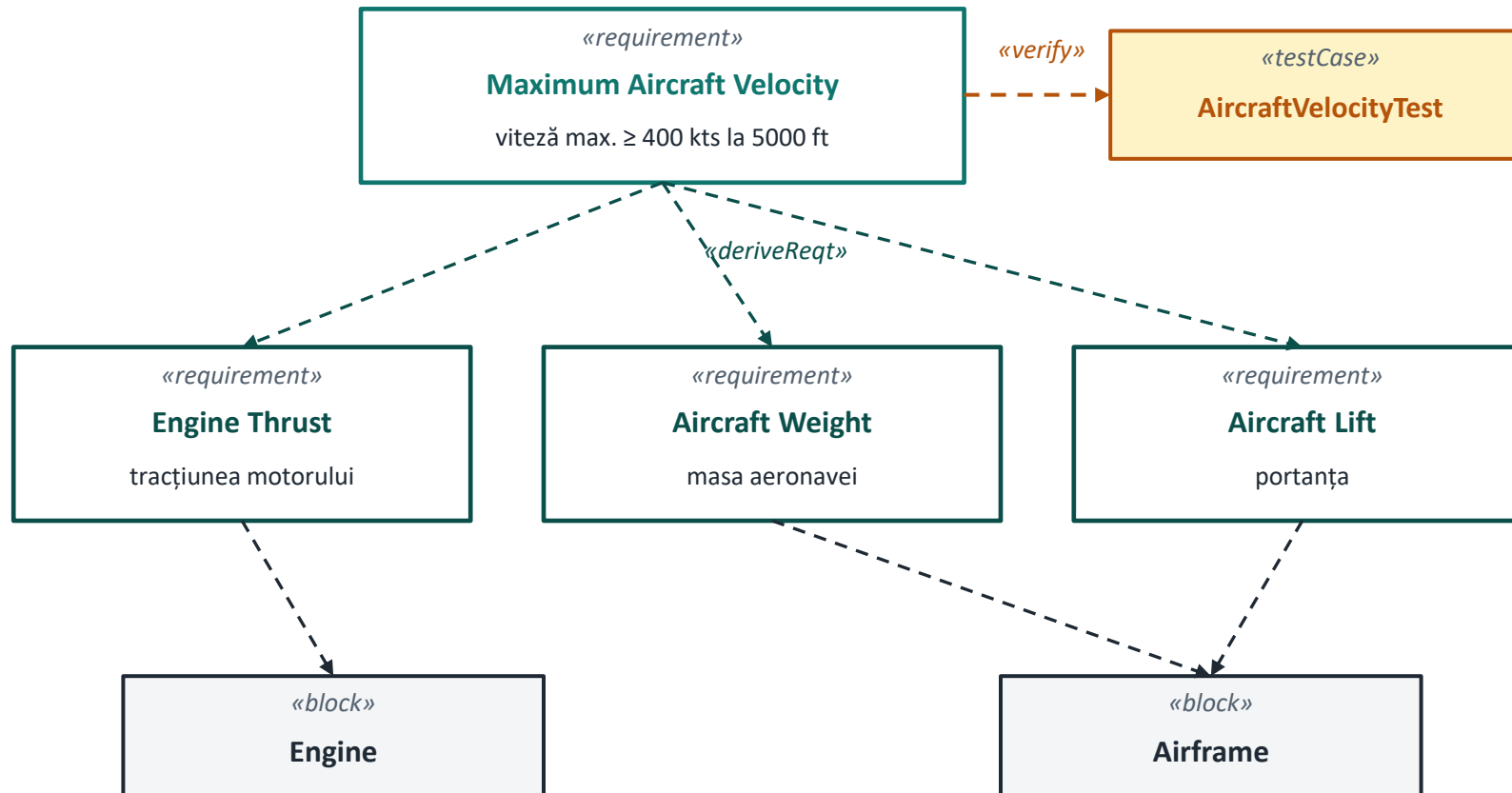
ce face sistemul, prin blocuri funcționale și legăturile dintre ele

Diagrama de activitate

sucesiunea acțiunilor și a fluxurilor

- Capitolul detaliază primele două. Diagrama funcțională (*Functional Block Diagram*) formalizează analiza funcțională din Capitolul 9; detalierea completă a diagramelor aparține etapelor următoare [16].

14.3 Diagrama de cerințe: exemplul avionului



- Textul cerinței principale:
„The aircraft **shall** have a maximum velocity of no less than 400 kts at 5000 ft altitude under standard day conditions”; „shall” = „trebuie să” (Capitolul 10).

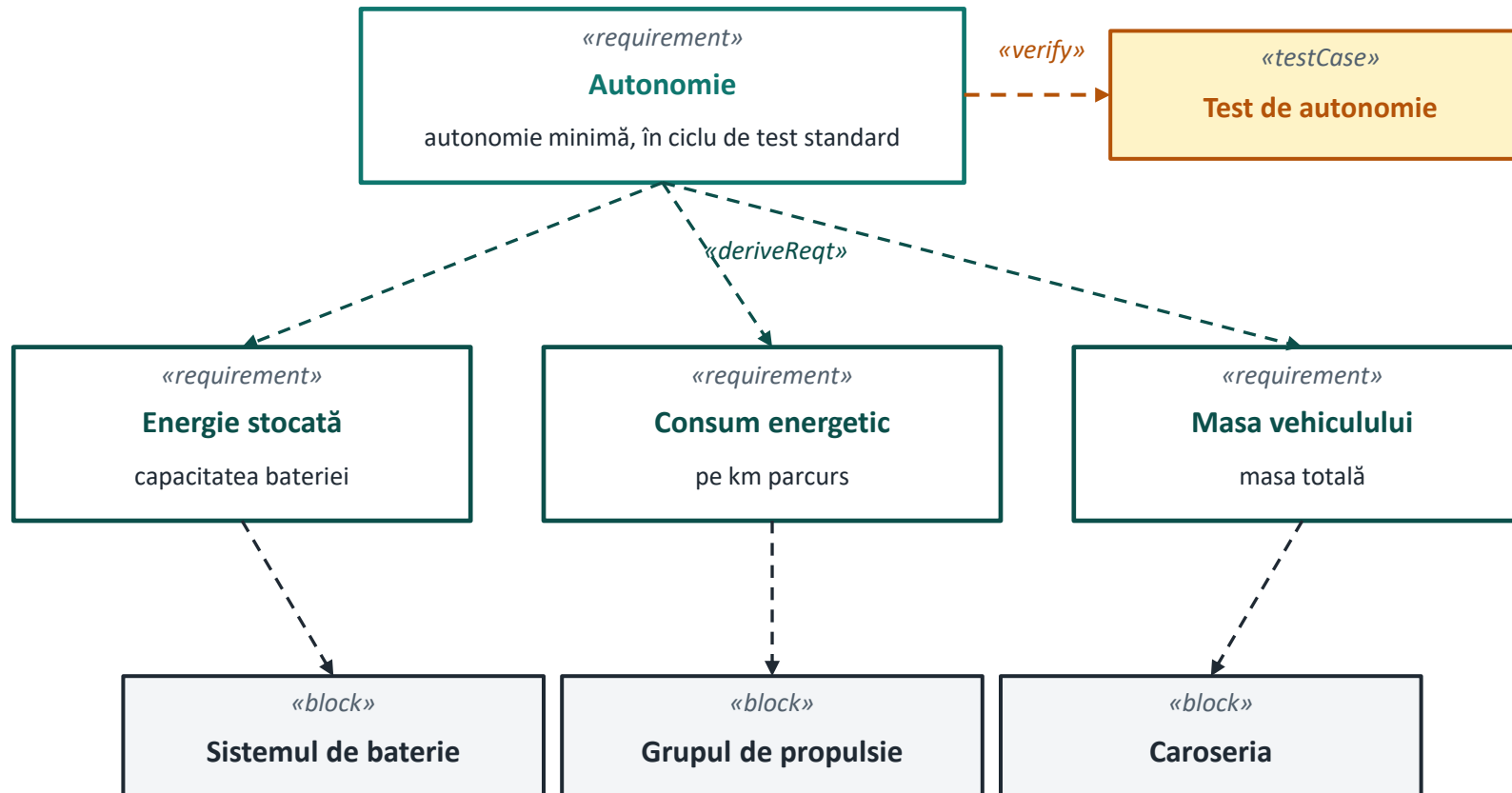
Fig. 14.1. Diagramă de cerințe SysML pentru viteza maximă a unui avion (exemplu ilustrativ, notația [16]).

Cum se citește diagrama de cerințe

- 1 Cerința de nivel sistem** Maximum Aircraft Velocity definește performanța globală cerută.
- 2 Cerințe derivate «deriveReq»** viteza depinde de tracțiune, masă și portanță.
- 3 Elemente fizice «satisfy»** Engine satisface tracțiunea; Airframe, masa și portanța.
- 4 Verificare «verify»** cerința principală este verificată prin AircraftVelocityTest.

Este lanțul obiectiv → funcție → cerință → verificare din Capitolul 10, acum formalizat într-un model: instrumentele SysML pot verifica automat că fiecare cerință are un element care o satisface și un test care o verifică.

Firul roșu: cerințele automobilului electric



- Autonomia depinde de energia stocată, de consum și de masă; dar masa crește odată cu bateria: un **compromis clasic**, pe care diagrama îl face vizibil.
- Fiecare cerință derivată este satisfăcută de un subsistem din arhitectura din Capitolul 13. Valorile concrete (km, kWh) nu se fixează aici.

Fig. 14.2. Diagramă de cerințe pentru autonomia automobilului electric (exemplu ilustrativ).

14.4 Diagrama de context: avionul de pasageri

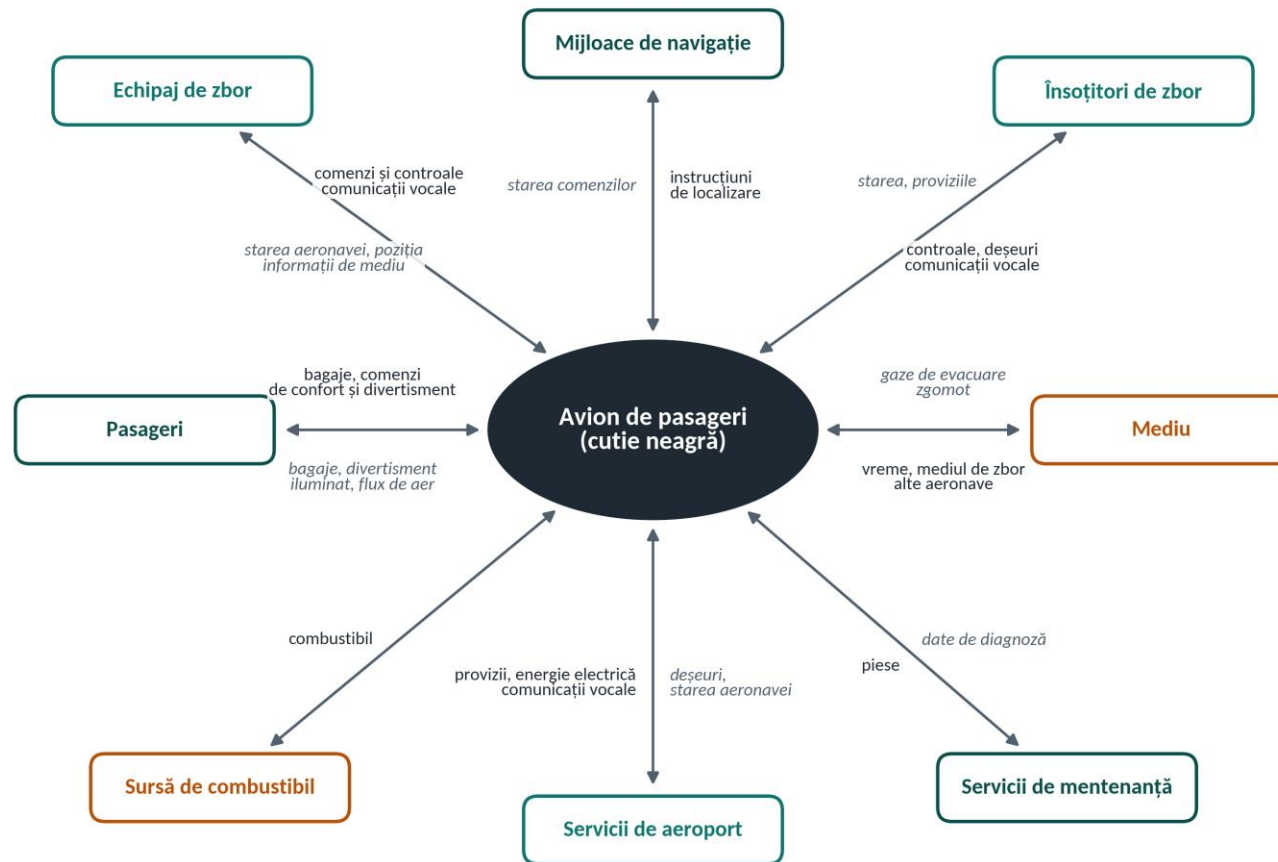


Fig. 14.3. Diagrama de context a unui avion de pasageri. Sursa: adaptat după [1, cap. 3].

- Este aceeași diagramă de context din Capitolul 3, aplicată unui sistem complex. Sistemul analizat este tratat ca **cutie neagră** (*black-box*).
- Fluxurile sunt de patru tipuri: **informații** (stare, comenzi, comunicații), **energie** (combustibil, electricitate), **materiale** (provizii, bagaje) și **efecte asupra mediului** (zgomot, emisii).
- Fiecare săgeată devine, în arhitectură, o interfață externă de definit.

14.5 Exemplu rezolvat: arhitectură sau proiectare?

Pentru sistemul ATC aflat în etapa de arhitectură, care dintre afirmații aparțin **arhitecturii**? Testul: afirmația spune „cine face ce și cum comunică” sau „cum se construiește”?

Afirmația	Verdict	Explicație
Sistemul este organizat în subsisteme de colectare date, procesare, interfață operator, comunicații.	DA	descompunerea în subsisteme
Sistemul utilizează radar primar și secundar.	NU	tehnologie concretă: proiectare
Există o interfață om–mașină pentru controlorii de trafic.	DA	interfață identificată; forma ei ține de proiectare
Datele sunt procesate cu un algoritm bazat pe inteligență artificială.	NU	soluție de implementare
Subsistemele comunică prin interfețe bine definite.	DA	relațiile dintre subsisteme
Serverele sunt configurate în redundanță activ–activ.	NU	redundanța ca principiu poate fi arhitectură; configurația concretă, nu
Sistemul trebuie să respecte cerințe stricte de siguranță și timp real.	INTRARE	este o cerință care constrânge arhitectura

Arhitectura spune **CINE** face ce; proiectarea spune **CUM**.

14.6 SysML v2: noua generație a limbajului

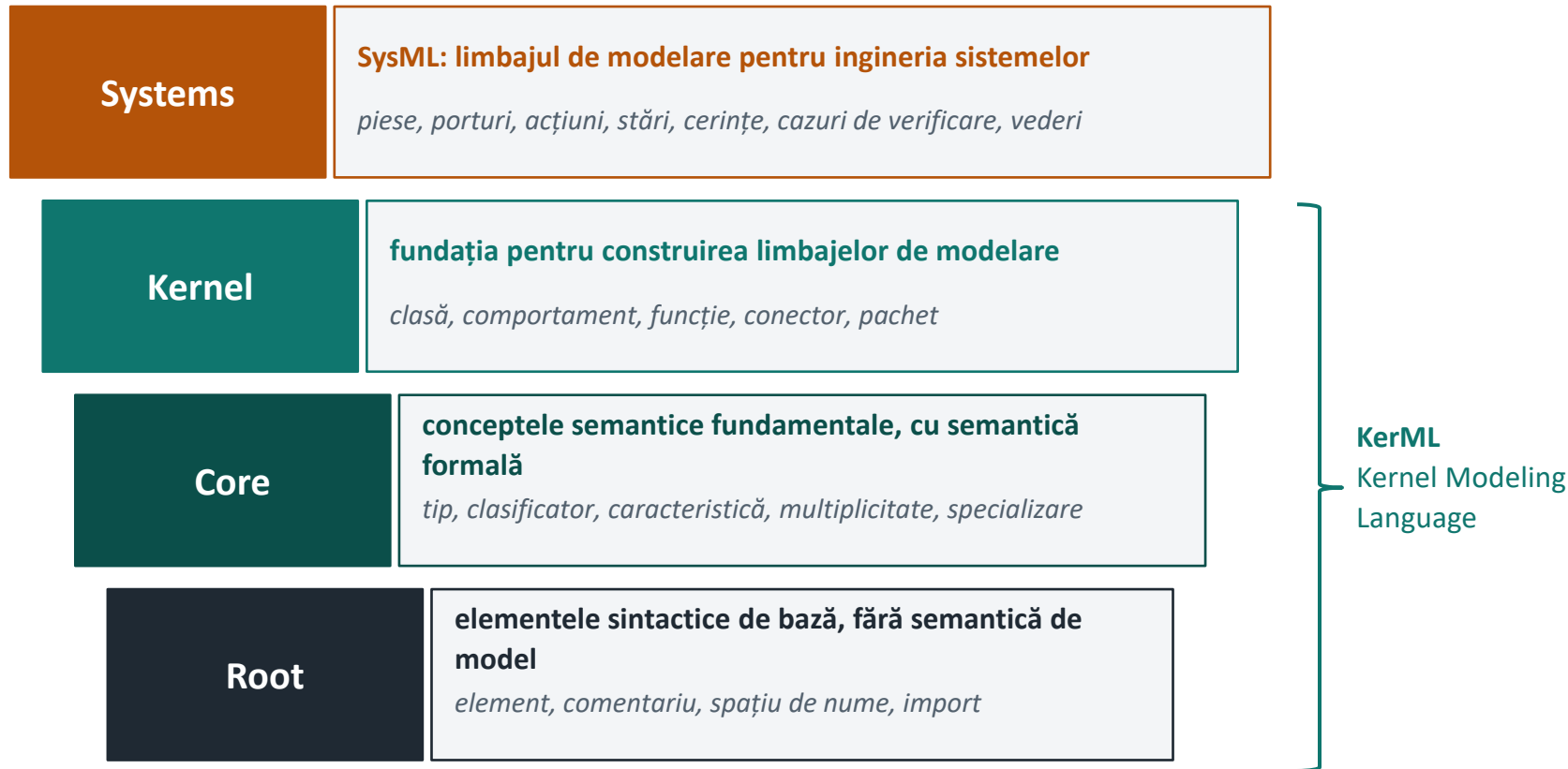
- În iulie 2025, Object Management Group a adoptat oficial **SysML v2** [18]. Nu este o simplă revizie: limbajul a fost reconstruit, pe baza experienței acumulate în cei peste 15 ani de utilizare a SysML v1.

Tabelul 14.1. SysML v1 și SysML v2: principalele diferențe.

Aspect	SysML v1.x	SysML v2
Fundamentul limbajului	profil UML: adaptarea unui limbaj din software	limbaj nou, cu bază semantică formală proprie (KerML)
Notăția	numai grafică (diagrame)	textuală și grafică, echivalente
Elementele de structură	blocuri «block»	definiții și utilizări: part def / part
Interfețele	porturi proxy, blocuri de interfață	port def, interface def, flow
Schimbul de modele	XML, dependent de instrument	interfață de programare (API) standard

Conceptele din acest capitol rămân valabile: cerință, derivare, satisfacere, verificare, context. Se schimbă **modul de exprimare**, care devine precis și verificabil automat.

Arhitectura limbajului SysML v2



- Primele trei niveluri formează **KerML**, un limbaj-nucleu cu semantică formală, traductibilă în logică matematică.
- SysML se construiește peste KerML: fiecare element de model are un înțeles precis, nu doar o reprezentare grafică.
- **Consecința practică:** un model SysML v2 poate fi verificat automat de instrument, nu doar desenat.

Fig. 14.4. Arhitectura pe patru niveluri a limbajului SysML v2. Sursa: adaptat după [19].

Definiție și utilizare: tiparul de bază al limbajului

notație textuală SysML v2

```
// Definiții
part def Vehicle {
    attribute mass : ISQ::MassValue;
    part eng : Engine;
}
part def Engine {
    part cyl : Cylinder[4..6];
}
part def Cylinder;

// Utilizări
part smallVehicle : Vehicle {
    part redefines eng {
        part redefines cyl[4];
    }
}
part bigVehicle : Vehicle {
    part redefines eng {
        part redefines cyl[6];
    }
}
```

Fig. 14.5. Două variante de vehicul obținute din aceeași definiție.
Sursa: adaptat după [19].

Definiția (def)

descrie un tip: ce caracteristici au toate elementele de acest fel.
Corespunde blocului din SysML v1.

Utilizarea

folosește definiția într-un context anume: motorul acestui vehicul, cu exact 4 cilindri.

Același tipar se aplică tuturor elementelor: **part def / part**, **port def / port**, **action def / action**, **requirement def / requirement**. Odată înțeles, se poate citi întregul limbaj.

Structura: porturi, interfețe și fluxuri

```

port def FuelOutPort {
  out item fuelSupply : Fuel;
  in item fuelReturn : Fuel;
}
port def FuelInPort {
  in item fuelSupply : Fuel;
  out item fuelReturn : Fuel;
}
interface def FuelInterface {
  end port supplierPort : FuelOutPort;
  end port consumerPort : FuelInPort;
}
part vehicle : Vehicle {
  part tankAssy : FuelTankAssembly;
  part eng : Engine;
  interface : FuelInterface connect
    supplierPort ::> tankAssy.fuelTankPort to
    consumerPort ::> eng.engineFuelPort;
}

```

notație textuală SysML v2

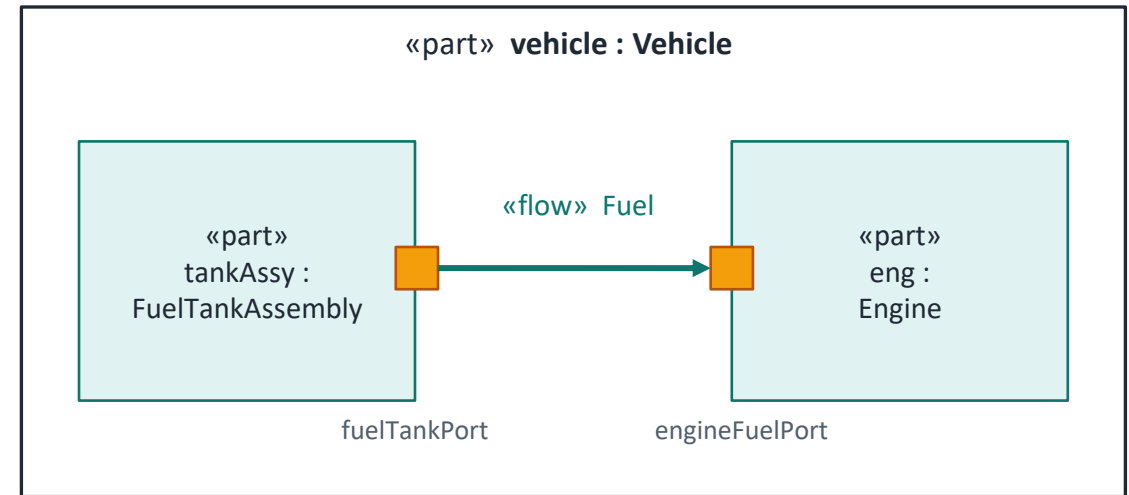


Fig. 14.6. Rezervorul și motorul, legate printr-o interfață cu flux de combustibil. Sursa: adaptat după [19].

- **Portul** este punctul prin care o piesă își expune, controlat, caracteristicile. Două porturi sunt compatibile dacă direcțiile se potrivesc invers (out ↔ in).
- **Interfața** se definește o singură dată și se reutilizează: este interfața din Capitolul 4, acum verificabilă de instrument.

Comportamentul (1): acțiuni și succesiuni

notație textuală SysML v2

```

action def Focus { in scene : Scene; out image : Image; }
action def Shoot { in image : Image; out picture : Picture; }

action def TakePicture {
  in scene : Scene;
  out picture : Picture;

  bind focus.scene = scene;
  action focus : Focus { in scene; out image; }

  flow focus.image to shoot.image;
  first focus then shoot;

  action shoot : Shoot { in image; out picture; }
  bind shoot.picture = picture;
}
  
```

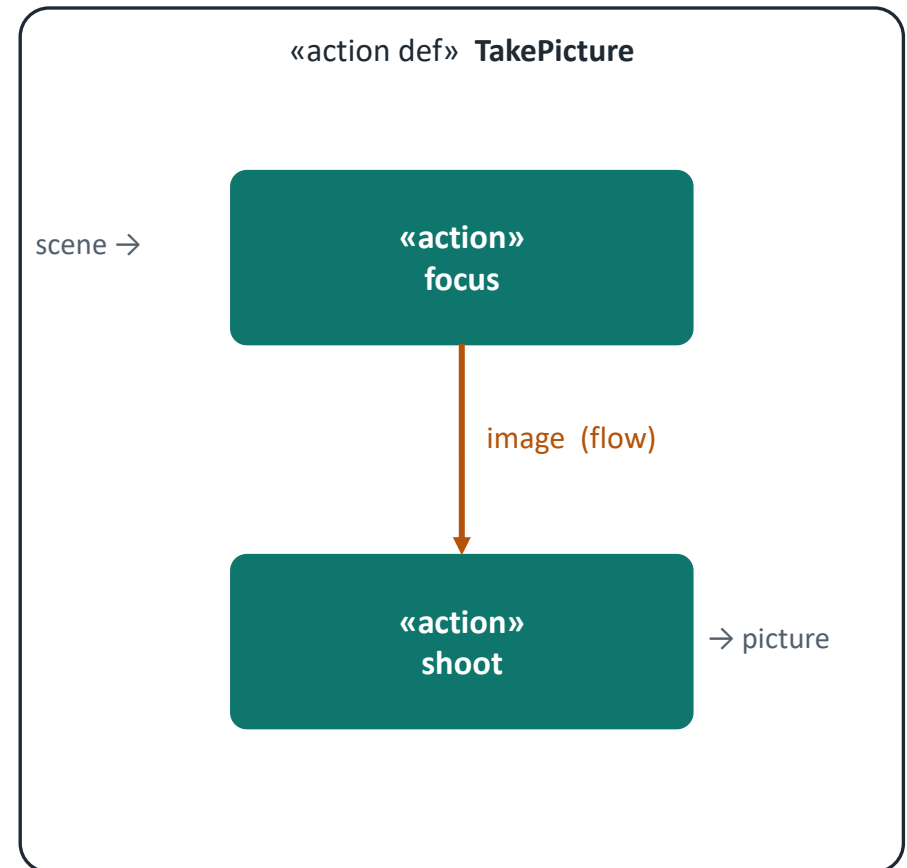


Fig. 14.7. Acțiunea „fă o fotografie”: focalizare, apoi declanșare. Sursa: adaptat după [19].

Comportamentul (2): stări și tranziții

```
state def VehicleStates {
  first start then off;

  state off;
  transition off_to_starting
    first off
    accept VehicleStartSignal
    then starting;

  state starting;
  transition starting_to_on
    first starting
    accept VehicleOnSignal
    then on;

  state on;
  transition on_to_off
    first on
    accept VehicleOffSignal
    then off;
}
```

notație textuală SysML v2

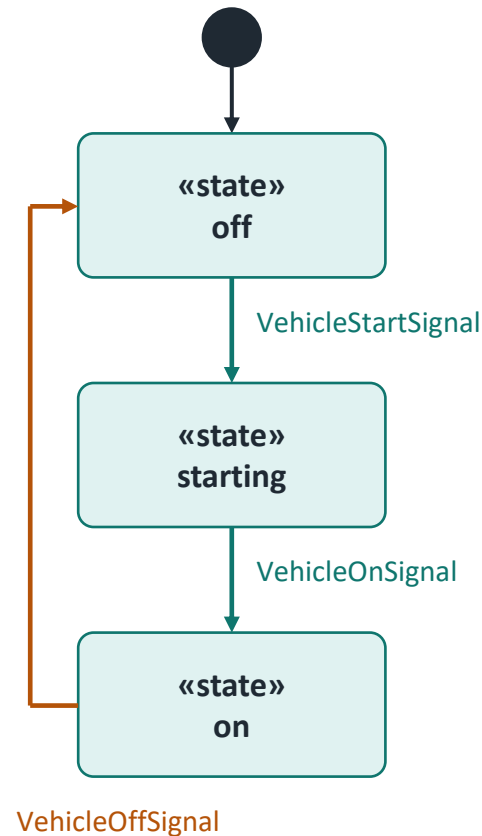


Fig. 14.8. Stările unui vehicul: oprit, pornire, pornit. Sursa: adaptat după [19].

- O **definiție de stare** corespunde mașinii de stări din SysML v1.
- Tranzițiile se declanșează la primirea unui semnal (**accept**).
- O stare poate avea acțiuni la intrare, pe durată și la ieșire (**entry**, **do**, **exit**).
- Piesa care are aceste stări le declară prin **exhibit**: comportamentul este legat direct de structură.

Cerințe, satisfacere și verificare în SysML v2

```

requirement def MassLimitationRequirement {
  doc /* The total mass of a vehicle shall be less
       * than or equal to the required mass. */
  subject vehicle : Vehicle;
  attribute massReqd : ISQ::MassValue;
  require constraint { vehicle.mass <= massReqd }
}
requirement <'1.1'> vehicleMassLimit
  : MassLimitationRequirement {
  attribute :>> massReqd = 2000[SI::kg];
}
satisfy vehicleMassLimit by vehicle_c1;

verification def VehicleMassTest {
  subject testVehicle : Vehicle;
  objective { verify vehicleMassLimit; }
  return verdict : VerdictKind;
}

```

notație textuală SysML v2

Tabelul 14.2. Relațiile din diagrama de cerințe, în SysML v2.

SysML v1 (Fig. 14.1)	SysML v2
«requirement»	requirement def / requirement
«deriveReq»	conexiune de derivare
«satisfy»	satisfy ... by ...
«testCase» + «verify»	verification def + verify

Cerința devine o **constrângere evaluabilă**:
instrumentul poate verifica dacă masa
vehiculului respectă limita de 2000 kg.
Verdictul unui caz de verificare este *pass*, *fail*,
inconclusive sau *error*.

Fig. 14.9. Cerință, satisfacere și caz de verificare pentru masa vehiculului. Sursa:
adaptat după [19].

Exemplu rezolvat: controlul accesului în SysML v2

notație textuală SysML v2

```
package ControlAcces {
  private import ISQ::*;
  private import SI::*;
  action def VerificaDreptAcces;
  part def Identificare;
  part def DecizieAcces {
    perform action verificare : VerificaDreptAcces;
  }
  part def ActionarePuncteAcces;
  part def MonitorizareJurnalizare;
  part def SistemControlAcces {
    attribute timpRaspuns : DurationValue;
    part identificare : Identificare;
    part decizie : DecizieAcces;
    part actionare : ActionarePuncteAcces;
    part monitorizare : MonitorizareJurnalizare;
  }
  requirement def <'R1'> AccesRapid {
    doc /* Sistemul trebuie să permită accesul utilizatorilor
        * autorizați în maximum 2 s de la solicitare. */
    subject sistem : SistemControlAcces;
    require constraint { sistem.timpRaspuns <= 2[s] }
  }
  part sistemCladire : SistemControlAcces;
  requirement accesRapid : AccesRapid;
  satisfy accesRapid by sistemCladire;
}
```

1

Structura

cele patru subsisteme din arhitectura de la 13.4 devin piese ale sistemului.

2

Alocarea funcției

„verifică dreptul de acces” (Capitolul 9) este executată (perform) de subsistemul de decizie.

3

Cerința

cerința de 2 s din Capitolul 10 devine o constrângere asupra timpului de răspuns.

4

Satisfacerea

sistemul clădirii este declarat ca satisfăcând cerința; un caz de verificare o poate testa.

Concluzii

- 1 SysML este un instrument de modelare, nu arhitectura.
- 2 Diagrama de cerințe: relațiile deriveReq, satisfy, verify.
- 3 Diagrama de context: granița sistemului și fluxurile care o traversează.
- 4 Arhitectura organizează; tehnologiile vin la proiectare.
- 5 SysML v2 exprimă aceleași concepte textual și grafic, cu semantică formală, verificabilă automat.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Pentru sistemul ales, construiți o diagramă de cerințe cu o cerință de nivel sistem, trei cerințe derivate, subsistemele care le satisfac și testul de verificare.

Tema 2

Realizați diagrama de context a aceluiași sistem, la nivel de arhitectură. Clasificați fluxurile: informații, energie, materiale, efecte asupra mediului.

Tema 3

Scrieți în notația textuală SysML v2 structura aceluiași sistem (trei–patru piese) și una dintre cerințele lui, cu constrângerea și relația satisfy. Folosiți ca model exemplul rezolvat din secțiunea 14.6.

CAPITOLUL 15

Dezvoltarea inginerească: de la arhitectură la soluții concrete

Cuprinsul capitolului

15.1 Drumul parcurs și poziționarea în ciclul de viață

15.2 Ce este dezvoltarea inginerească

15.3 Pașii și activitățile principale

15.4 Legătura dintre arhitectură și dezvoltare

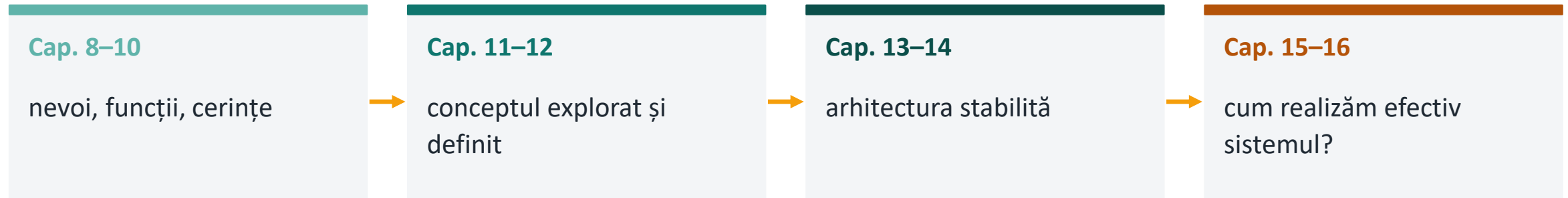
15.5 Ce se decide abia acum: ATC și automobilul

15.6 Riscuri specifice

15.7 Abordări ale dezvoltării: predictivă și adaptivă

Concluzii · Teme de aplicare

15.1 Drumul parcurs și poziționarea în ciclul de viață



- Dezvoltarea inginerescă începe după finalizarea arhitecturii și include proiectarea detaliată, implementarea, integrarea și testarea [1, cap. 10], [1, cap. 12].
- Este etapa cu **cel mai mare consum de resurse** și cu **cele mai multe decizii ireversibile** (comenzi ferme de componente, scule de producție, contracte cu furnizori). De aceea greșelile din etapele anterioare costă aici cel mai mult (Fig. 6.2).

Dezvoltarea inginerescă transformă arhitectura în soluții concrete.

15.2 Ce este dezvoltarea inginerescă

Transformarea subsistemelor arhitecturale în soluții tehnice concrete. Apar pentru prima dată tehnologiile concrete, dimensionarea și furnizorii: exact elementele excluse deliberat în capitolele 11–14.

proiectare hardware și
software

alegerea tehnologiilor

dimensionarea componentelor

implementare și integrare

Arhitectura

„ce și cum este organizat”

Dezvoltarea inginerescă

„cum se construiește efectiv”

Este etapa „hands-on” a ciclului de viață; fazele ei sunt cele din Fig. 13.1.

15.3 Pași și activitățile principale

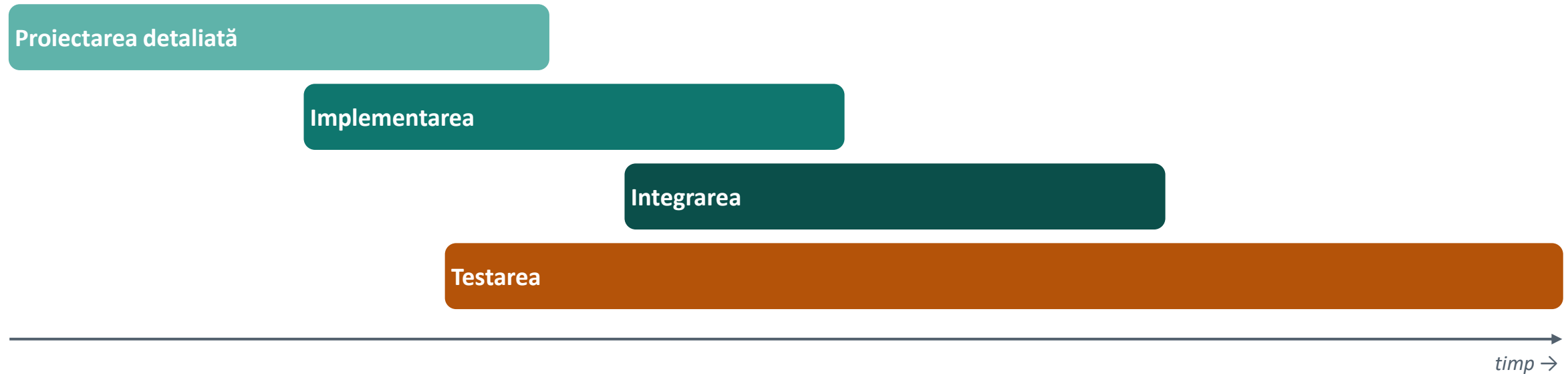
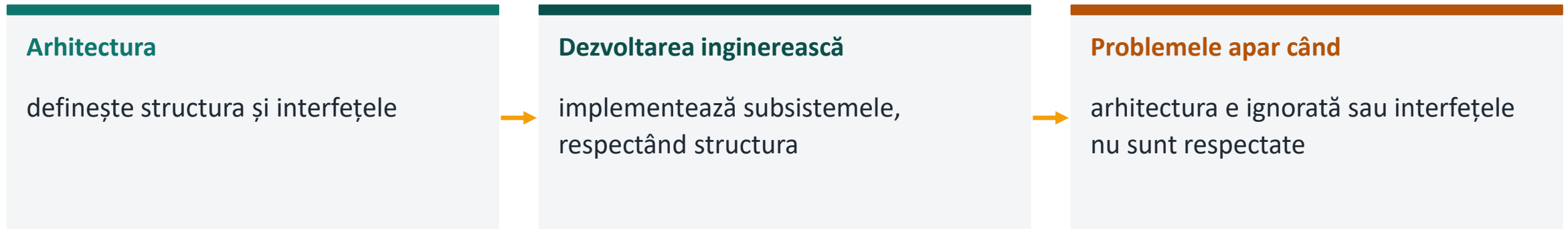


Fig. 15.1. Pașii dezvoltării ingineresti se suprapun (reprezentare schematică).

- Logic, pașii urmează această ordine; în practică se suprapun: testarea începe încă din implementare (teste de componente) și continuă pe toată integrarea (Capitolul 16).
- Activități principale: proiectarea detaliată a subsistemelor; dezvoltarea componentelor hardware și software; integrarea progresivă; testarea și validarea tehnică; gestionarea modificărilor. Toate sunt iterative și interdependente.

15.4 Legătura dintre arhitectură și dezvoltare



- Fiecare subsistem este proiectat și dezvoltat separat, de echipa lui, dar respectă rolul și interfețele stabilite. O echipă care schimbă „local” formatul datelor pe care le trimite poate bloca întreaga integrare.
- Dacă în implementare apare nevoia de a schimba arhitectura, aceasta este o **modificare formală**, cu analiză de impact (Capitolul 16), nu o decizie locală.

Dezvoltarea inginerescă implementează deciziile, nu le reinventează. O arhitectură bună face dezvoltarea mai simplă și mai sigură.

15.5 Ce se decide abia acum: ATC și automobilul

Tabelul 15.1. Decizii de proiectare care devin legitime în dezvoltarea inginerescă.

Subsistem	Decizie de proiectare (abia acum)
ATC · Colectare date	radar primar și secundar
ATC · Procesare	algoritm bazat pe inteligență artificială
ATC · Securitate și redundanță	servere configurate activ–activ
Auto · Sistemul de baterie	chimia celulelor, capacitatea, răcirea
Auto · Grupul de propulsie	tipul motorului electric, inverterul
Auto · Sistemul de încărcare	puterea maximă de încărcare, standardul de conector
Auto · Managementul energiei	software-ul de gestiune a bateriei (BMS)

Afirmațiile marcate „NU” în exemplul din secțiunea 14.5 devin acum decizii legitime; fiecare trebuie să respecte rolul și interfețele din arhitectură (de exemplu, motorul respectă interfața frânare–propulsie).

15.6 Riscuri specifice dezvoltării ingineresti

de integrare

tehnologice

depășiri de cost și timp

probleme de
compatibilitate

modificări târzii ale
cerințelor

- Sunt aceleași categorii din Capitolul 7, acum cu impact financiar maxim. În această etapă, riscurile tehnice se transformă rapid în **riscuri programatice**: o problemă tehnică nerezolvată devine, în câteva săptămâni, depășire de cost și de termen.

Majoritatea acestor riscuri pot fi reduse printr-o arhitectură clară și stabilă: tot ce s-a investit în capitolele 13–14 se recuperează acum.

15.7 Abordări ale dezvoltării: predictivă și adaptivă

- Pașii dezvoltării ingineresti pot fi organizați în moduri diferite. Ciclul de viață al dezvoltării se poate parcurge **predictiv**, planificat de la început și executat conform planului, sau **adaptiv**, ajustat pe parcurs, pe măsură ce echipa și beneficiarul învață [17, cap. 10].

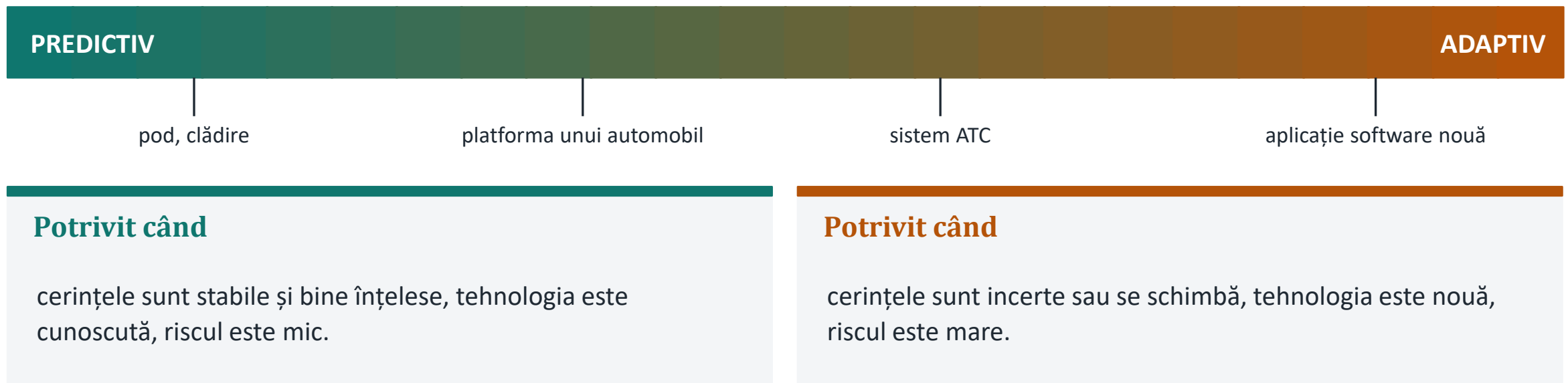
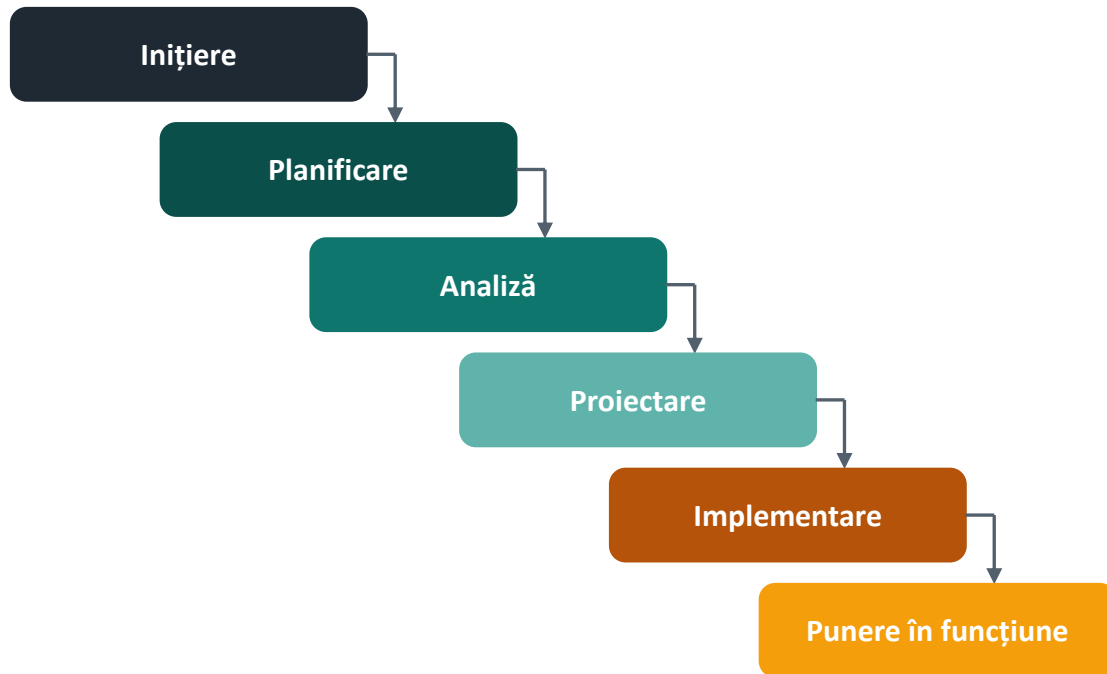


Fig. 15.2. Continuumul abordărilor, de la complet predictiv la complet adaptiv. Sursa: adaptat după [17, cap. 10].

Modelul în cascadă și cascada modificată

a) Cascada clasică



b) Cascada modificată

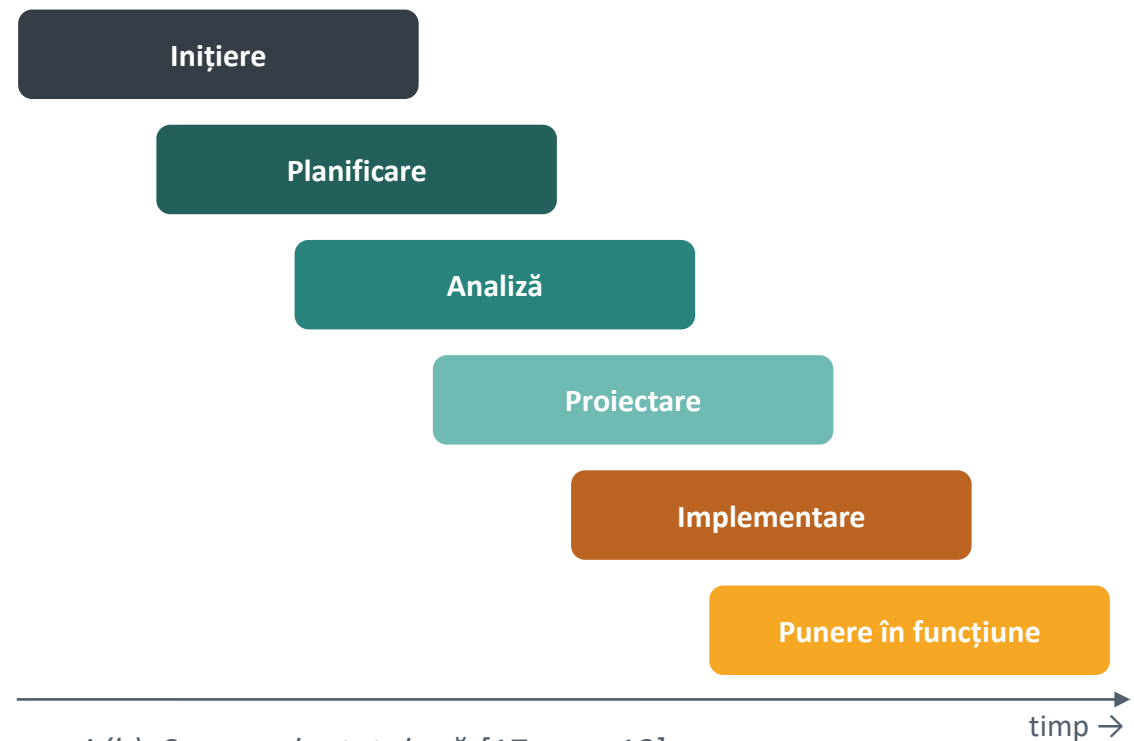
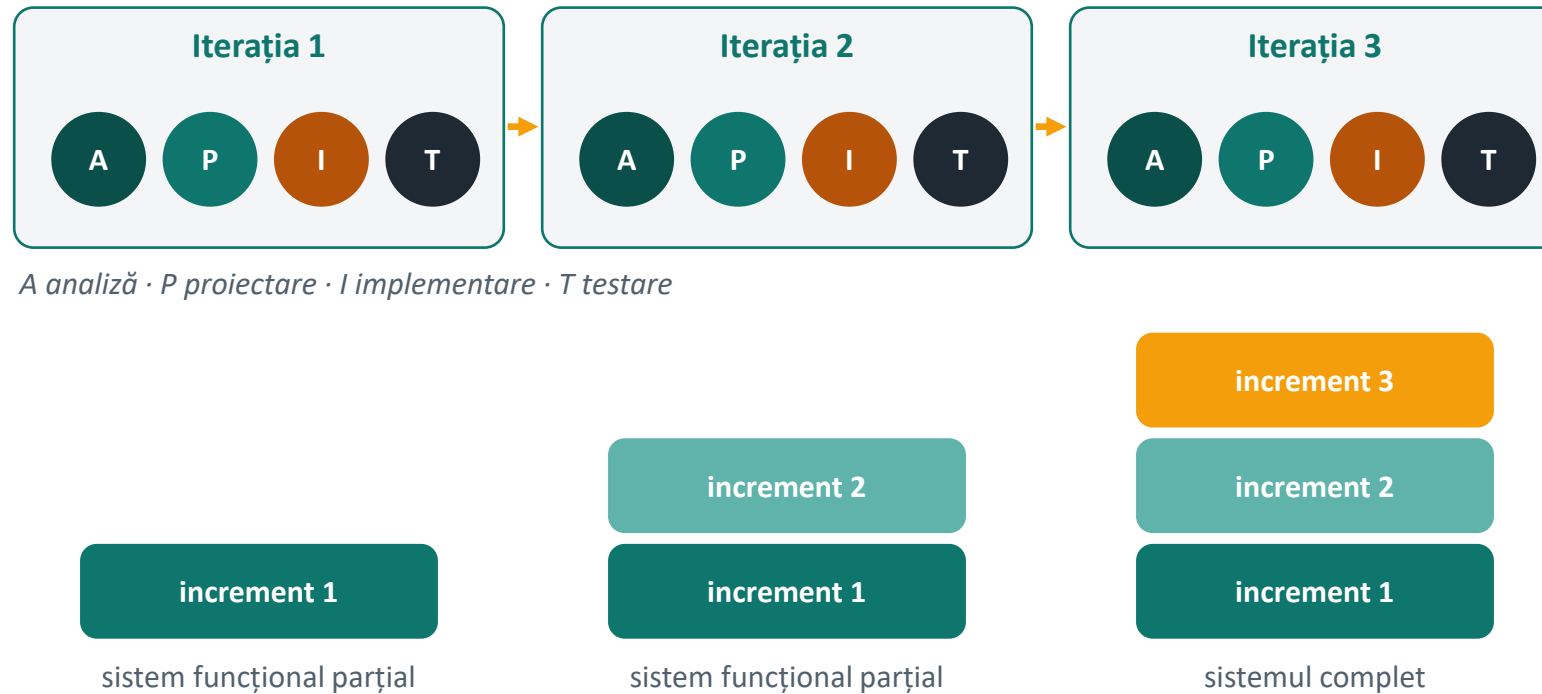


Fig. 15.3. Fazele parcurse succesiv (a) și cu suprapuneri (b). Sursa: adaptat după [17, cap. 10].

- Cascada clasică presupune că o fază se încheie înainte de următoarea, fără întoarcere; în practică a funcționat rar. Cascada modificată permite suprapunerea fazelor, dar proiectul rămâne unul singur, parcurs o dată. După punerea în funcțiune urmează **suportul** (mentenanța).

Dezvoltarea iterativă și incrementală



- **Iterația** este un mini-proiect: o parte mică a sistemului este analizată, proiectată, implementată și testată.
- **Incrementul** este partea de sistem finalizată și integrată; sistemul „crește” de la o iterație la alta.
- **Scheletul funcțional** (*walking skeleton*): întâi se construiește structura completă, cu funcționalitate minimă, apoi se completează.
- Este integrarea progresivă din Capitolul 16, extinsă la întregul proces.

Fig. 15.4. Iterații succesive; fiecare adaugă un increment integrat sistemului. Sursa: adaptat după [17, cap. 10].

Metodologie, model, instrument, tehnică

Metodologie

- ghid pentru toate fazele ciclului de viață: ce activități, ce modele, ce tehnici.

Exemplu: Procesul unificat (UP), Scrum

Model

- reprezentare a unui aspect important al realității (o abstracție), de obicei grafică.

Exemplu: diagrama de context, diagrama de cerințe (Capitolul 14)

Instrument

- aplicație software care ajută la crearea modelelor sau a altor componente ale proiectului.

Exemplu: un mediu de modelare SysML, un mediu integrat de dezvoltare (IDE)

Tehnică

- set de indicații despre cum se realizează o activitate sau o sarcină.

Exemplu: formularea „Sistemul trebuie să...” (Capitolul 10), interviul cu părțile interesate

Metodologia spune **ce** se face în fiecare fază; tehnica spune **cum**; modelul este **rezultatul**; instrumentul ajută la **producerea** lui.

Filozofia Agile

- Agile este o filozofie, nu o metodologie completă: un set de valori pentru proiecte desfășurate într-un mediu necunoscut și în schimbare rapidă [17], [20].

Răspunsul la schimbare	<i>mai mult decât</i>	respectarea unui plan
Indivizii și interacțiunile	<i>mai mult decât</i>	procesele și instrumentele
Produsul funcțional	<i>mai mult decât</i>	documentația exhaustivă
Colaborarea cu beneficiarul	<i>mai mult decât</i>	negocierea contractului

Modelarea agilă

- modelează cu un scop
- puține modele, simple
- mai multe modele complementare
- feedback rapid
- conținutul contează mai mult decât forma

Agile nu înseamnă „scrie cod și repară”, nici renunțarea la modele: modelele rămân esențiale pentru a fixa decizii, a înțelege cerințele și a gândi soluțiile [17].

Procesul unificat (UP)



Fig. 15.5. Cele patru faze ale Procesului unificat. Sursa: adaptat după [17, cap. 10].

Discipline de dezvoltare

modelarea afacerii · cerințe · proiectare · implementare · testare · livrare

Discipline de suport

configurație și modificări · managementul proiectului · mediul de lucru

Fazele arată **accentul**, nu o succesiune strictă: fiecare iterație conține activități din toate disciplinele. În Elaborare, arhitectura și riscurile mari se rezolvă devreme, ca în capitolele 7 și 13.

Scrum și programarea extremă (XP)

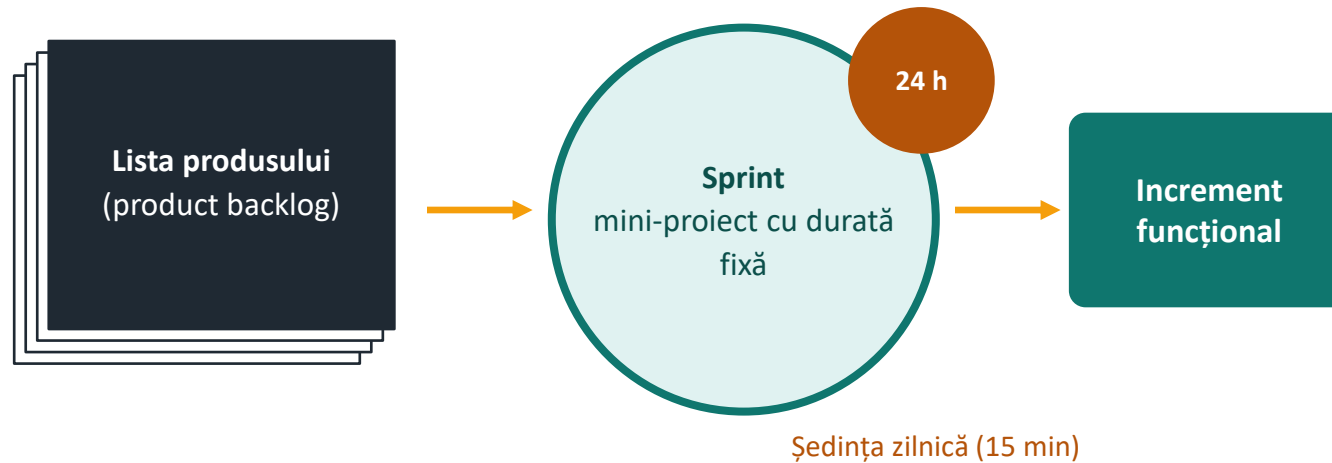


Fig. 15.6. Ciclul Scrum: de la lista produsului la incrementul funcțional. Sursa: adaptat după [17, cap. 10].

- **Proprietarul produsului:** reprezentantul beneficiarului; gestionează și prioritizează lista.
- **Scrum master:** facilitator, aplică regulile Scrum și înlătură obstacolele.
- **Echipa:** auto-organizată; în ședința zilnică fiecare spune ce a făcut, ce va face și ce îl împiedică.

Programarea extremă (XP)

- **Valori:** comunicare, simplitate, feedback, curaj.
- **Practici:**
 - testele se scriu înainte de cod
 - programare în pereche
 - refactorizare și proiectare simplă
 - integrare continuă, livrări mici
 - beneficiarul lucrează alături de echipă
 - proprietate colectivă asupra codului

Ce abordare alegem pentru un sistem tehnic?

Tabelul 15.2. Abordarea predictivă și abordarea adaptivă: comparație.

Criteriu	Predictivă	Adaptivă
Cerințele	stabile, bine înțelese	incerte, în evoluție
Riscul tehnic	mic, tehnologie cunoscută	mare, tehnologie nouă
Planificarea	completă, la început	pe iterații, ajustată continuu
Implicarea beneficiarului	la început și la recepție	continuă, în cadrul echipei
Livrarea	o singură dată, la final	incremente funcționale frecvente

Sistemele tehnice complexe folosesc de regulă o abordare **hibridă**: arhitectura, interfețele și integrarea urmează un plan predictiv (modelul V, Capitolul 16), iar subsistemele cu incertitudine mare, în special software-ul, se dezvoltă iterativ [3].

- **Firul roșu:** platforma și caroseria automobilului se planifică pe ani, predictiv; software-ul de asistență și de infotainment evoluează iterativ, inclusiv după vânzare, prin actualizări de la distanță.

Concluzii

- 1 Dezvoltarea inginerescă duce sistemul de la arhitectură la soluții concrete.
- 2 Cel mai mare consum de resurse și cele mai multe decizii ireversibile.
- 3 Pașii se suprapun; arhitectura îi ghidează pe toți.
- 4 Implementarea respectă arhitectura, nu o schimbă.
- 5 Abordarea, predictivă sau adaptivă, se alege după stabilitatea cerințelor și nivelul riscului.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

Pentru fiecare subsistem din arhitectura sistemului ales (Capitolul 13), numiți două decizii de proiectare care se iau acum și interfața pe care trebuie să o respecte.

Tema 2

Identificați trei riscuri specifice dezvoltării ingineresti pentru același sistem și propuneți câte o măsură de atenuare (Capitolul 7).

Tema 3

Stabiliți care subsisteme ale aceluiași sistem se pot dezvolta predictiv și care adaptiv. Justificați prin stabilitatea cerințelor și riscul tehnic, apoi schițați primele trei iterații ale unui subsistem adaptiv.

CAPITOLUL 16

Integrarea, testarea și gestionarea modificărilor

Cuprinsul capitolului

16.1 Ce este integrarea sistemului

16.2 Integrarea progresivă

16.3 Modelul V: proiectare și testare în oglindă

16.4 De ce integrarea este etapa critică

16.5 Testarea, un proces continuu

16.6 Gestionarea modificărilor

16.7 Studiu de caz: unde apar problemele în integrarea ATC

Concluzii · Teme de aplicare

16.1 Ce este integrarea sistemului

Toate subsistemele au trecut testele. Funcționează sistemul? **Încă nu știm**; abia integrarea ne-o spune. Optimizarea componentelor nu garantează performanța sistemului (Capitolul 1).

Integrarea este procesul prin care subsistemele dezvoltate separat sunt conectate pentru a funcționa ca un sistem unitar [1, cap. 13]. Ea verifică:

compatibilitatea interfețelor

fluxurile de informații

comportamentul global

Integrarea nu este doar asamblare, ci **validarea reală a arhitecturii**: prima etapă în care sistemul este evaluat ca întreg.

16.2 Integrarea progresivă



Fig. 16.1. Integrarea progresivă: fiecare nivel se testează înainte de următorul.

- Integrarea progresivă (incrementală) evită **integrarea „big-bang”**, în care toate subsistemele sunt conectate deodată și sursa unei erori devine aproape imposibil de localizat [1, cap. 13].
- Nivelurile corespund tipurilor de testare din Capitolul 5: de integrare, de sistem, de acceptare, operațională.

16.3 Modelul V: proiectare și testare în oglindă

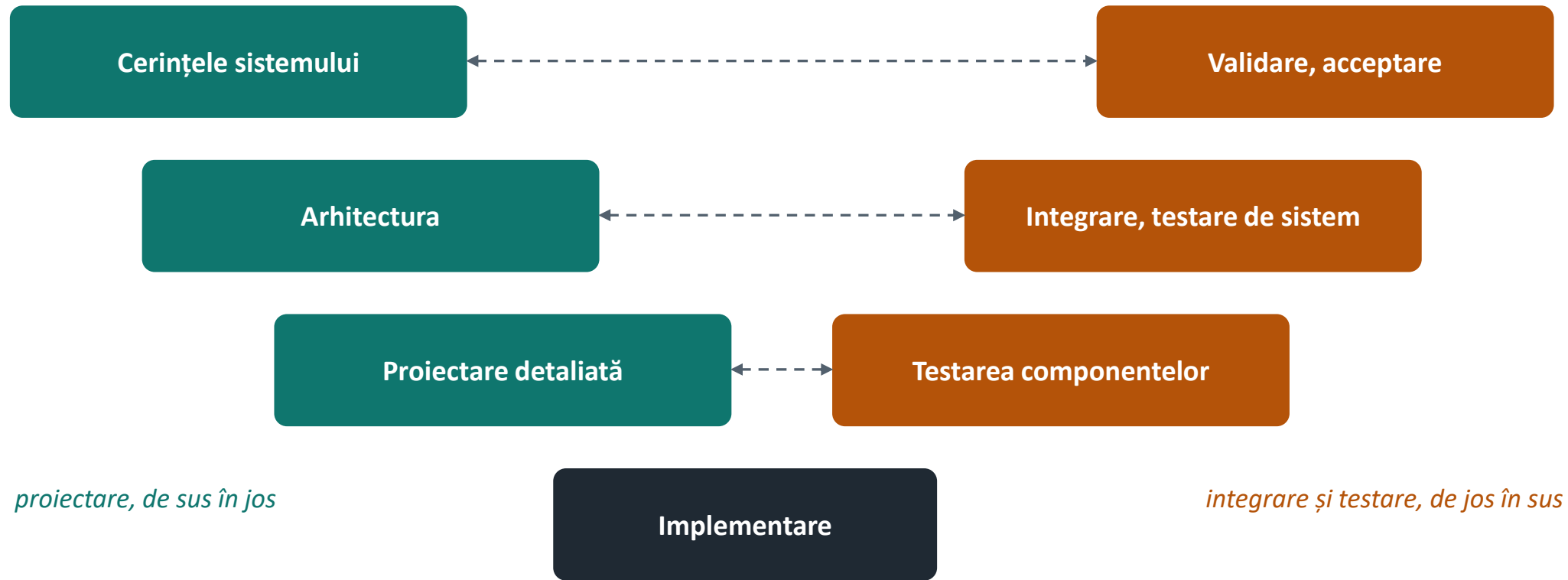


Fig. 16.2. Modelul V. Sursa: după [3], [4].

- Fiecare test verifică nivelul de proiectare aflat în oglindă. Testele se planifică **odată cu proiectarea** nivelului corespunzător, nu la final: când scrieți o cerință, vă gândiți deja cum o veți verifica (Capitolul 10).

16.4 De ce integrarea este etapa critică

Problemele apar deoarece

- interfețele sunt interpretate diferit (Cap. 4, 13)
- sincronizarea nu este corectă
- apar dependențe neprevăzute
- factorul uman este subestimat (Cap. 4)

Integrarea scoate la iveală

- erori de arhitectură
- cerințe incomplete (Cap. 10)
- decizii timpurii greșite (Cap. 11–12)

- Exemplu ATC: la integrarea lanțului colectare date → procesare → suport decizional → interfață operator → comunicații se verifică **latența** (un avion la 800 km/h parcurge peste 200 m pe secundă), **consistența datelor** (aceeași imagine a traficului peste tot) și **încărcarea operatorilor**.

Integrarea este „momentul adevărului”: ea nu creează probleme, ci le arată pe cele care existau deja.

16.5 Testarea, un proces continuu

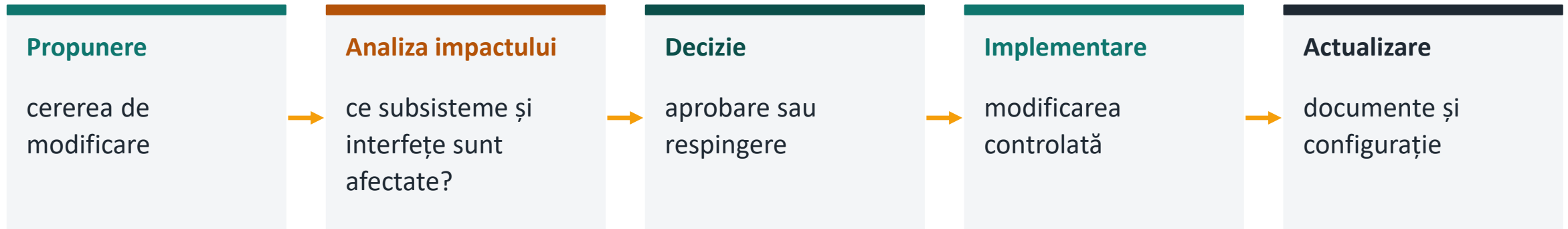
Tabelul 16.1. Cele patru niveluri de testare.

Nivel	Ce verifică	Mai aproape de
Integrare	interfețele dintre subsisteme	verificare: „construim corect?”
Sistem	sistemul complet, față de cerințe	verificare
Acceptare	sistemul, față de client și contract	validare: „construim ce trebuie?”
Operațională	sistemul, în mediul real de utilizare	validare

- Testarea verifică dacă implementarea respectă arhitectura și dacă cerințele sunt satisfăcute; începe în timpul integrării și continuă pe măsură ce sistemul evoluează [1, cap. 13]. Legătura cu distincția verificare–validare din Capitolul 10 este orientativă. Ingineria sistemelor coordonează toate nivelurile.

Testarea nu este o etapă finală, ci un proces continuu.

16.6 Gestionarea modificărilor



- Modificările sunt inevitabile și pot afecta costul, durata și arhitectura. Procesul controlat de modificare face parte din **managementul configurației**, unul dintre planurile coordonate prin SEMP (Capitolul 5) [3], [5].
- **Firul roșu:** furnizorul de celule de baterie se schimbă în faza de integrare. Impactul, urmărit prin interfețe: celule cu alte dimensiuni → spațiul de montaj din caroserie → recalibrarea software-ului BMS → repetarea testelor de autonomie și siguranță.

Interfețele definite în arhitectură fac analiza impactului posibilă. Modificările târzii sunt mai scumpe și mai greu de integrat.

16.7 Studiu de caz: unde apar problemele în integrarea ATC

Arhitectura este definită, subsistemele au fost dezvoltate separat, sistemul intră în integrare. Pentru fiecare situație: ce tip de problemă este și ce etapă anterioară ar fi putut-o preveni?

Situația	Tipul problemei	Ar fi prevenit-o
Colectarea datelor transmite cu întârziere.	interfață critică, performanță	cerințe de latență (Cap. 10), interfețe (Cap. 13)
Operatorii sunt supraîncărcați informațional.	factor uman, HMI	scenarii operaționale (Cap. 8), HMI (Cap. 4)
Un subsistem respectă arhitectura, altul o interpretează diferit.	interfețe ambigue	interfețe formalizate, SysML (Cap. 14)
O cerință importantă este clarificată târziu.	modificare târzie	validarea cerințelor (Cap. 10)
Un subsistem funcționează izolat, dar nu și integrat.	problemă de integrare	interfețe (Cap. 4), integrare progresivă

Aproape nicio problemă de integrare nu își are originea în integrare. Dezvoltarea inginerescă este despre **coordonare**, nu doar despre execuție.

Concluzii

- 1 Integrarea este „momentul adevărului”.
- 2 Modelul V: fiecare nivel de proiectare are testul său.
- 3 Testarea începe devreme și continuă.
- 4 Modificările se controlează: analiza impactului prin interfețe.



Teme de aplicare

Temele de aplicare propun exersarea conceptelor din capitol pe un sistem real, cunoscut de cititor.

Tema 1

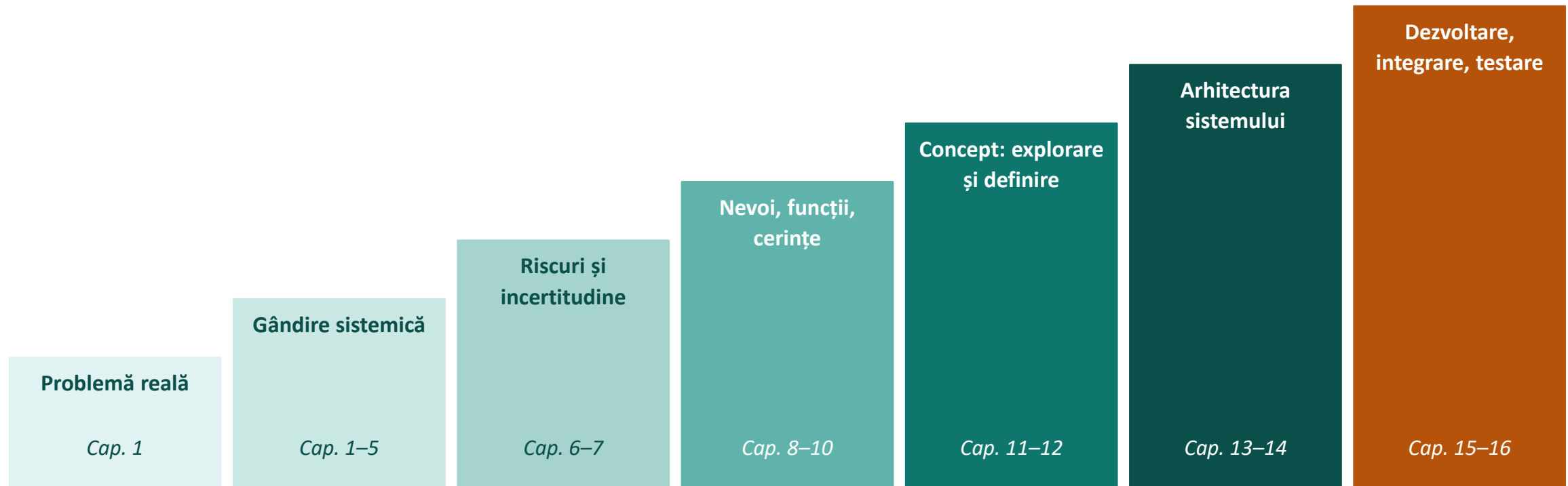
Pentru sistemul ales, propuneți un plan de integrare progresivă: ordinea în care integrați subsistemele și ce testați la fiecare pas.

Tema 2

Alegeți o modificare târzie plauzibilă pentru același sistem. Urmăriți-i impactul prin interfețe și estimați ce teste trebuie repetate.

Sinteza lucrării: firul logic

Întrebările lucrării: cum privim un sistem complex? · ce poate merge prost? · ce trebuie să facă sistemul? · ce tip de sistem alegem? · cum îl organizăm? · cum îl realizăm și îl integrăm?



Fiecare treaptă construiește pe cea anterioară: nu se poate face arhitectură fără concept, nici concept fără cerințe.

Exemplul integrator: controlul traficului aerian

De la problemă la cerințe · Cap. 6–10

- sistem critic, system-of-systems, timp real
- riscuri identificate devreme
- nevoi, funcții, cerințe corecte

Concept și arhitectură · Cap. 11–14

- centralizat, distribuit, asistat de AI
- selectat: control distribuit
- cinci subsisteme, interfețe critice

Dezvoltare și integrare · Cap. 15–16

- decizii tehnologice abia acum
- integrarea, momentul adevărului
- problemele apar la interfețe

Nu pornim de la soluții, ci de la problemă și risc. Deciziile timpurii reduc riscurile târzii; sistemele eșuează la interfețe.

Firele roșii ale lucrării

Tabelul S.1. Automobilul, de-a lungul lucrării.

Capitole	Automobilul
1–5	sistem complex, multidisciplinar; ținta UE 2035
6–10	riscul actualizărilor OTA; nevoi impuse de reglementări
11–12	patru concepte după 2035; conceptul electric definit
13–14	arhitectura, frânarea recuperativă; cerințele de autonomie
15–16	decizii de proiectare; o modificare târzie propagată prin interfețe

Controlul accesului

- scenarii și obiective (Cap. 8)
- funcții (Cap. 9)
- cerințe verificabile (Cap. 10)
- concepte alternative (Cap. 11)
- concept definit (Cap. 12)
- funcții alocate (Cap. 13)

Același sistem, privit cu instrumente tot mai detaliate: aceeași întrebare, pusă din ce în ce mai precis.

Ideea centrală a lucrării

Deciziile timpurii reduc riscurile târzii.

Sistemele eșuează la interfețe, nu la componente.

Cele două idei leagă toate capitolele: tot ce se face cu grijă în dezvoltarea conceptului și în arhitectură înseamnă mai puține surprize în dezvoltarea inginerască.

Bibliografie

- [1] Kossiakoff, A., Sweet, W. N., Seymour, S. J., Biemer, S. M. (2011). *Systems Engineering Principles and Practice*, ed. a 2-a. Hoboken, NJ: John Wiley & Sons.

- [2] ISO/IEC/IEEE 15288:2023. *Systems and software engineering — System life cycle processes*. Geneva: International Organization for Standardization.

- [3] INCOSE (2023). *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, ed. a 5-a (D. D. Walden et al., eds.). Hoboken, NJ: John Wiley & Sons.

- [4] NASA (2016). *NASA Systems Engineering Handbook*, NASA/SP-2016-6105 Rev2. Washington, DC: National Aeronautics and Space Administration.

- [5] Blanchard, B. S. (2004). *System Engineering Management*, ed. a 3-a. Hoboken, NJ: John Wiley & Sons.

- [6] Wasson, C. S. (2015). *System Engineering Analysis, Design, and Development: Concepts, Principles, and Practices*, ed. a 2-a. Hoboken, NJ: John Wiley & Sons.

Bibliografie (continuare)

- [7] Sheard, S. A., Mostashari, A. (2009). Principles of complex systems for systems engineering. *Systems Engineering*, 12(4), 295–311.

- [8] Bajzek, M., Fritz, J., Hick, H. (2021). Systems engineering principles. În: H. Hick, K. Küpper, H. Sorger (eds.), *Systems Engineering for Automotive Powertrain Development*. Cham: Springer, pp. 149–194.

- [9] Regulamentul (UE) 2023/851 al Parlamentului European și al Consiliului din 19 aprilie 2023 de modificare a Regulamentului (UE) 2019/631 în ceea ce privește consolidarea standardelor de performanță în materie de emisii de CO₂ pentru autoturismele noi și vehiculele utilitare ușoare noi. *Jurnalul Oficial al Uniunii Europene*, L 110, 25.4.2023.

- [10] Comisia Europeană (2025). Propunere de regulament de modificare a Regulamentului (UE) 2019/631 privind standardele de performanță în materie de emisii de CO₂ pentru autoturismele noi și vehiculele utilitare ușoare noi, 16.12.2025.

- [11] ADAC e.V. (2022). Infografic: numărul de vehicule (BEV, FCEV, e-fuel) care pot fi alimentate cu energia unei turbine eoliene de 3 MW; date VDE, 04.2022.

- [12] Liu, D. (2016). *Systems Engineering: Design Principles and Models*. Boca Raton, FL: CRC Press.

Bibliografie (continua)

- [13] ISO 31000:2018. *Risk management — Guidelines*. Geneva: International Organization for Standardization.

- [14] UNECE (2021). *UN Regulation No. 155 — Uniform provisions concerning the approval of vehicles with regards to cyber security and cyber security management system*. Geneva: United Nations Economic Commission for Europe.

- [15] ISO/IEC/IEEE 29148:2018. *Systems and software engineering — Life cycle processes — Requirements engineering*. Geneva: International Organization for Standardization.

- [16] Object Management Group (2019). *OMG Systems Modeling Language (OMG SysML™)*, Version 1.6. Needham, MA: OMG.

- [17] Satzinger, J. W., Jackson, R. B., Burd, S. D. (2016). *Systems Analysis and Design in a Changing World*, ed. a 7-a. Boston, MA: Cengage Learning.

- [18] Object Management Group (2025). *OMG Systems Modeling Language (SysML™)*, Version 2.0, și *Kernel Modeling Language (KerML™)*, Version 1.0. Needham, MA: OMG.

Bibliografie (continuare)

- [19] Seidewitz, E. (2026). *Introduction to the SysML v2 Language: Textual Notation*, Release 2026-08. OMG Systems Modeling Community / Model Driven Solutions, Inc. Licență CC BY 4.0.
- [20] Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., et al. (2001). *Manifesto for Agile Software Development*. <https://agilemanifesto.org>

Ingineria sistemelor nu înseamnă mai multă teorie, ci mai puține surprize în proiecte complexe.

