

M. Caramihai, © 2022

PROGRAMAREA ORIENTATA OBIECT

CURS 6

Alte limbaje vizuale (UML vs. SysML, BPMN, IDEF)

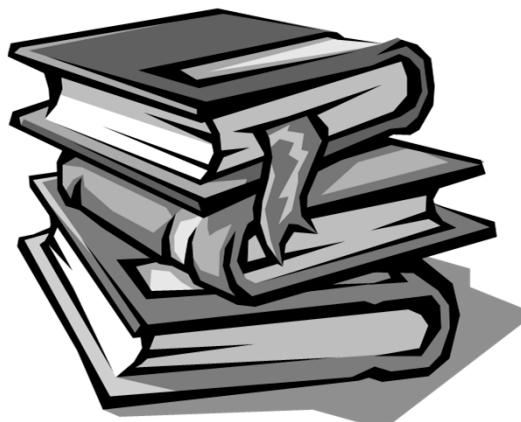


SysML



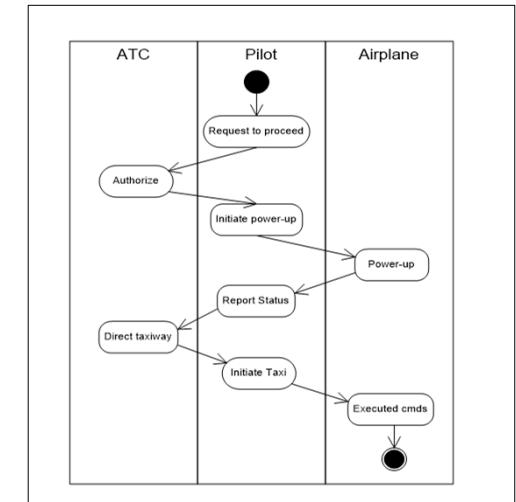
Modul de descriere a sistemelor

Inainte



- **Specificatii**
- **Interfata**
- **Proiectare sistem**
- **Analiza & implementare**
- **Testare**

In viitor



Trecerea de la “modelul documentar” la “modelul sistemic”

Ce este SysML (1)?

- Un limbaj de modelare vizuala – raspuns la **UML for Systems Engineering RFP** dezvoltat OMG, INCOSE, si AP233. adoptat de OMG in Iunie 2006
- Suporta specificarea, analiza, proiectarea, verificarea si validarea sistemelor (in sens general): se include hardware, software, date, personal, proceduri si facilitati
- Suporta modelarea si schimbul de date prin intermediul standardului XMI ® (XML Metadata Interchange)

Ce este SysML (2)?

- **Este** un limbaj de modelare vizuala care ofera:
 - Semantica = intelegerea conceptelor
 - Notatii= reprezentarea conceptelor
- **Nu este** o metodologie sau un instrument:
 - SysML este independent de orice limbaj de programare

UML – Software Engineering
SysML – System Engineering

SysML - Concepte

SysML (Systems Modeling Language) este un limbaj de modelare al aplicatiilor din ingineria de sistem.

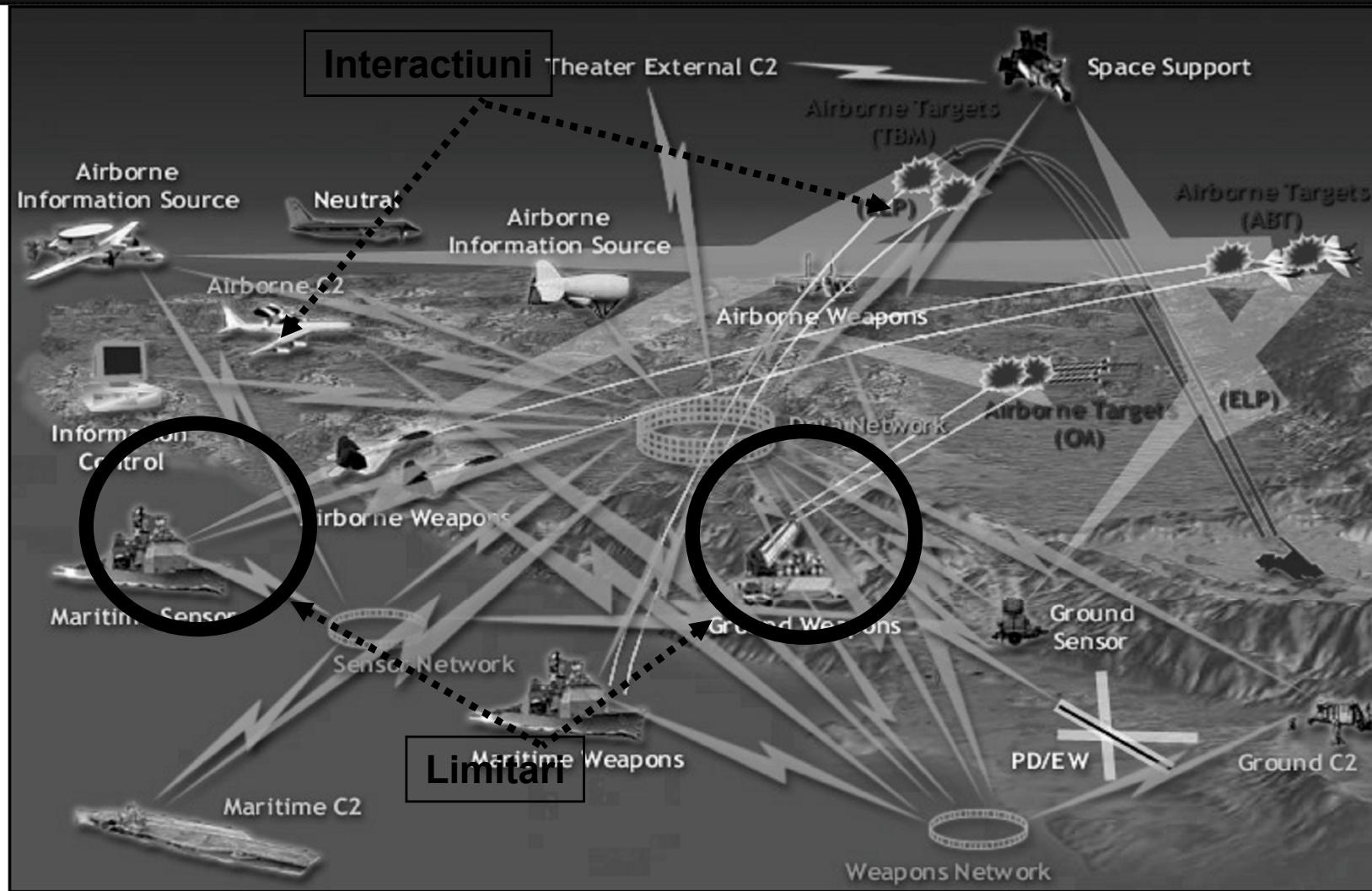
Suporta *specificarea, analiza, designul, verificarea si validarea* unei game vaste de sisteme si sisteme de sisteme. Aceste sisteme pot include *hardware, software, informatii, procese si facilitati*.

Ofera posibilitatea de a unifica concepte inrudite *software si non-software*, astfel umpland distanta ce se afla intre ele.

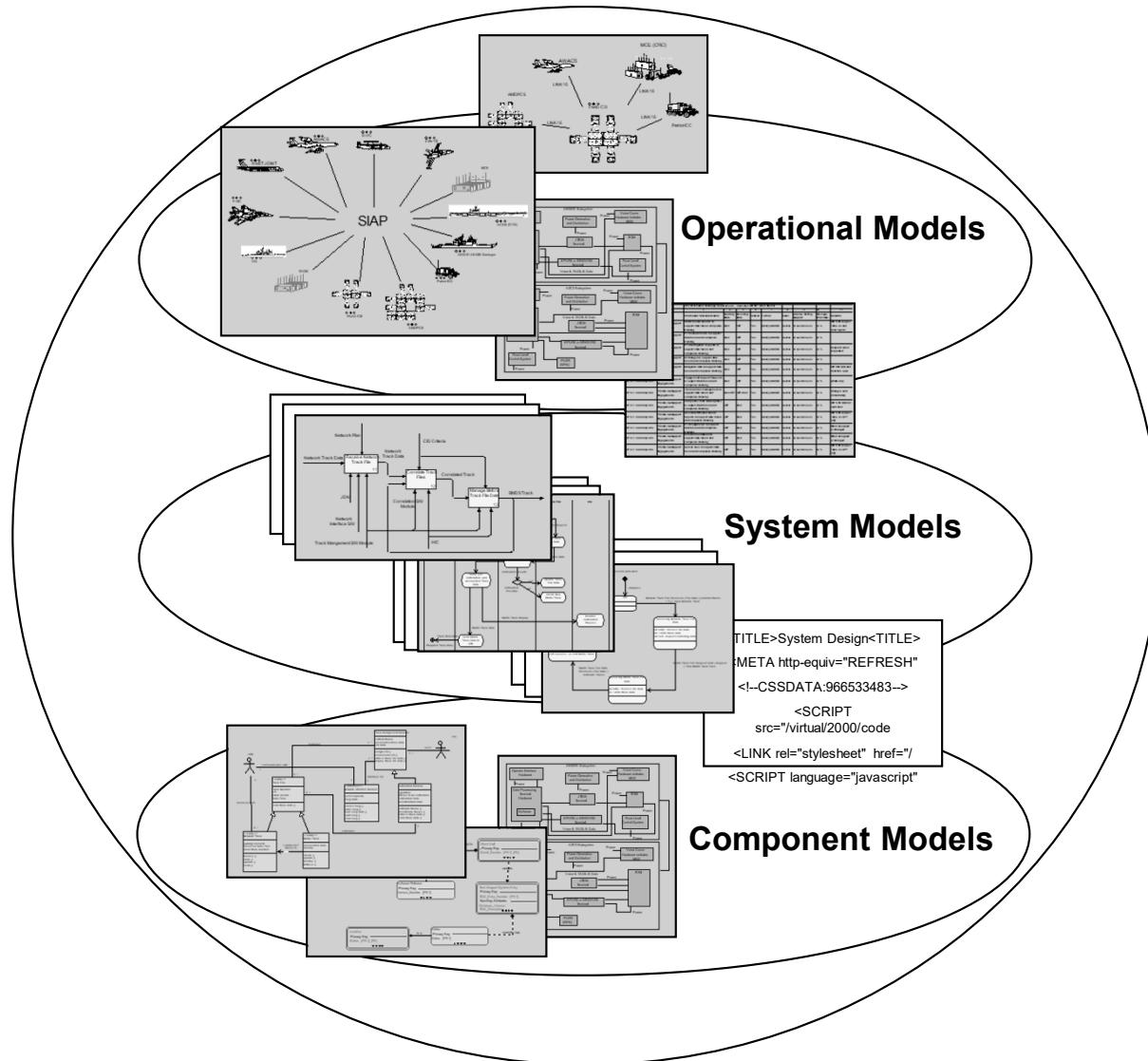
Modelarea sistemelor trebuie sa permita integrarea tuturor elementelor aferente acestora.

Reprezentarea unui "sistem de sisteme"

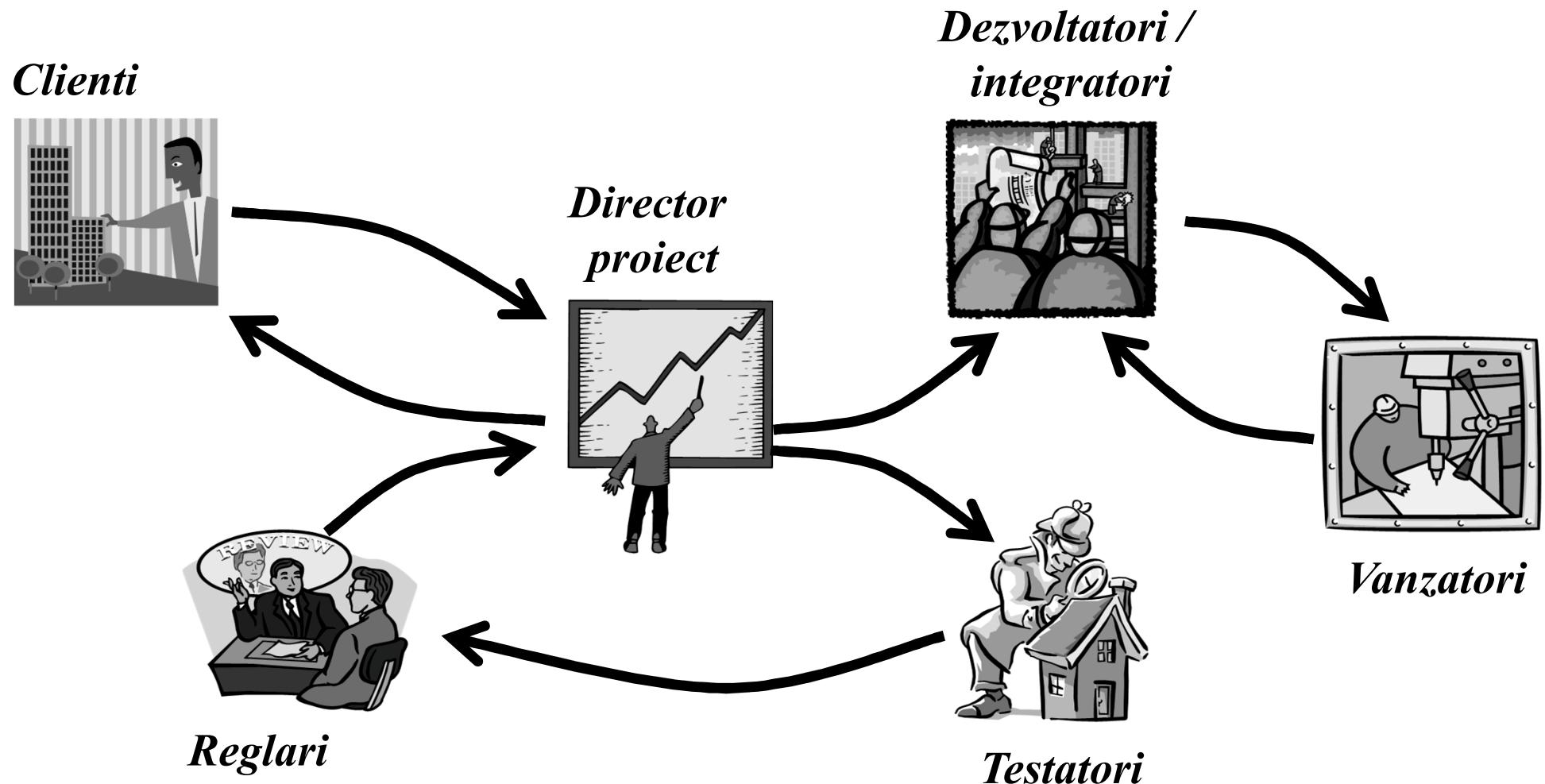
Copyright © 2006-2008 by Object Management Group.



Modelarea nivelerelor ierarhice

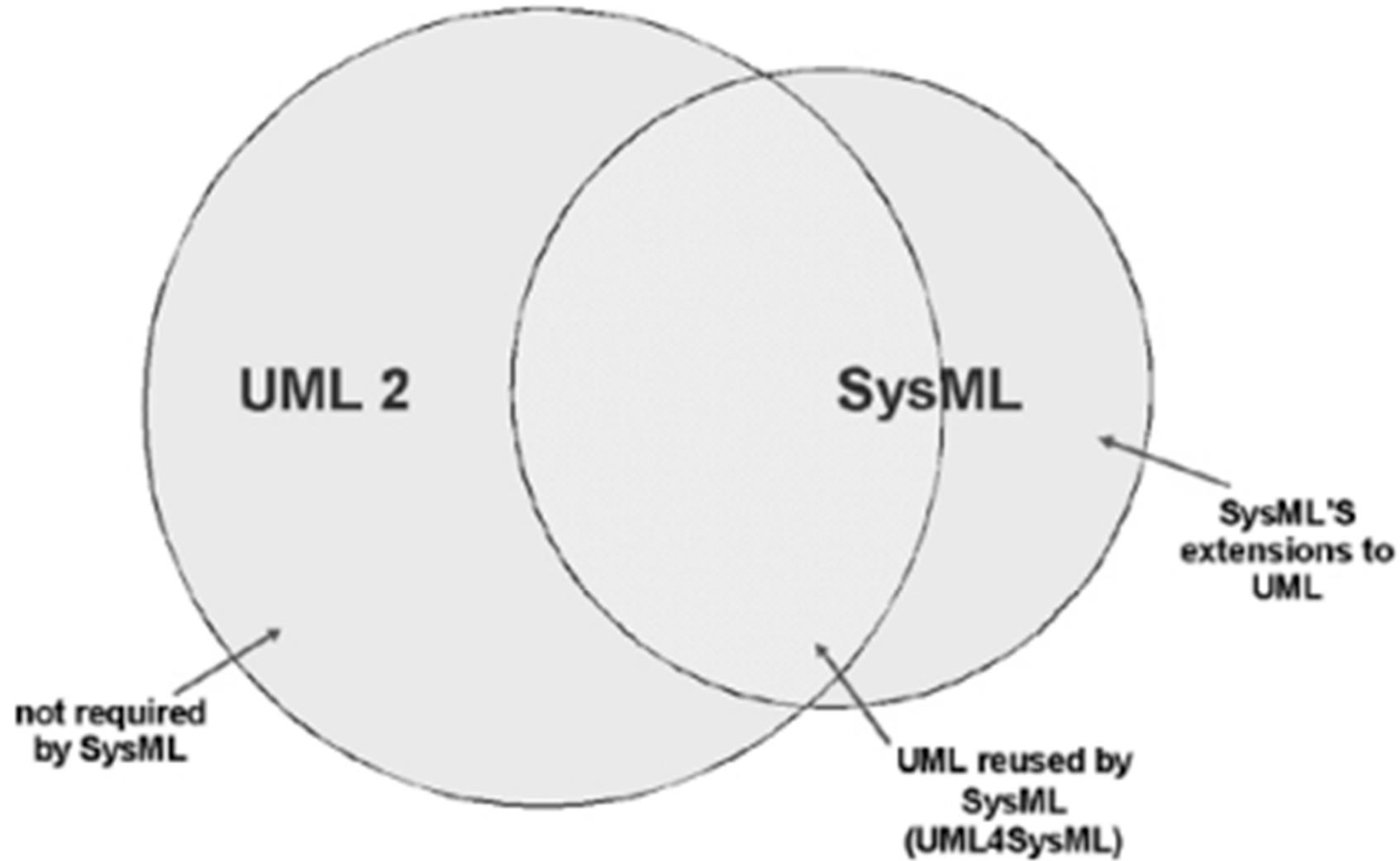


Modelarea comunicarii intre sisteme

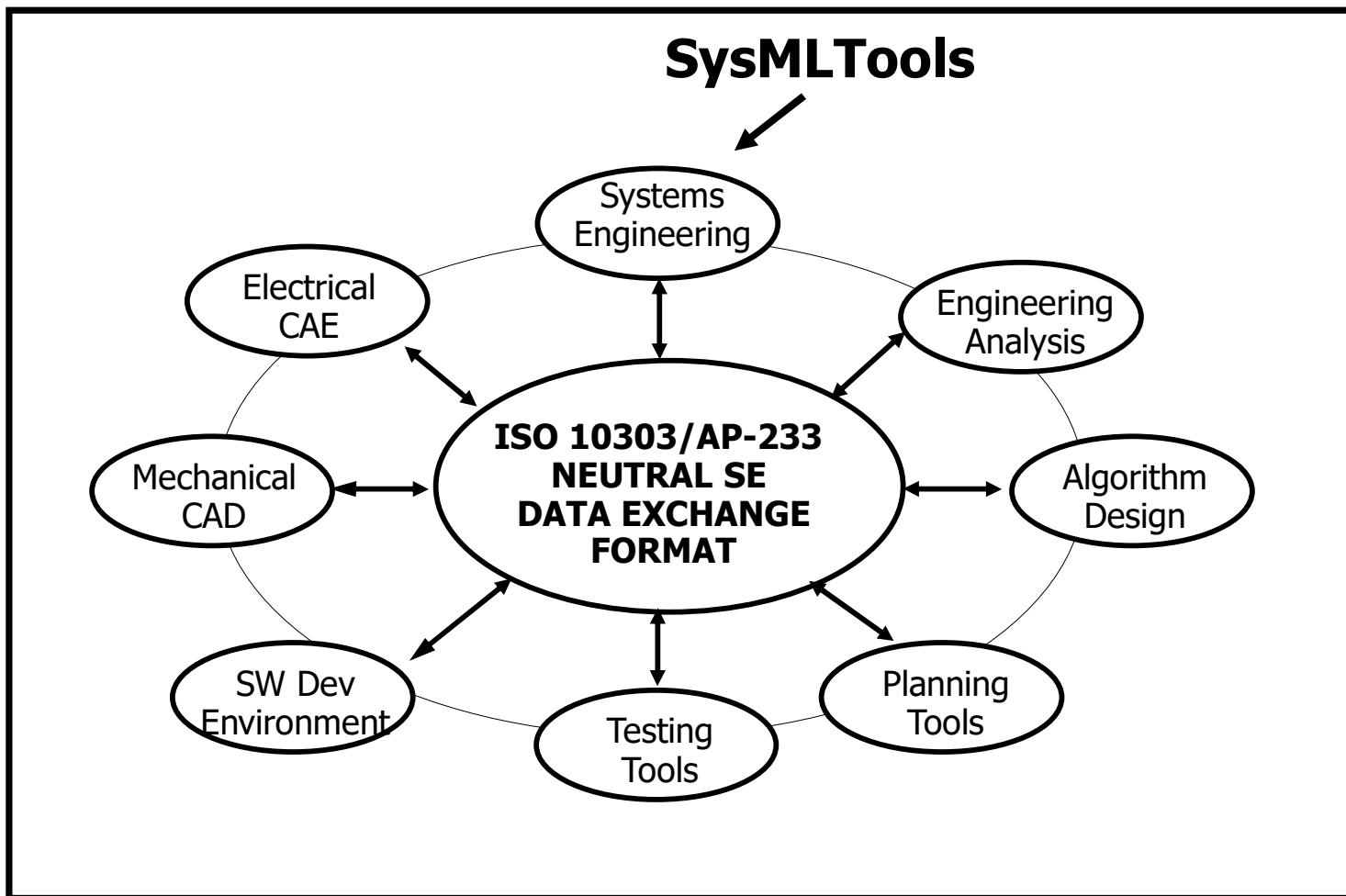


Legaturile intre SysML si UML

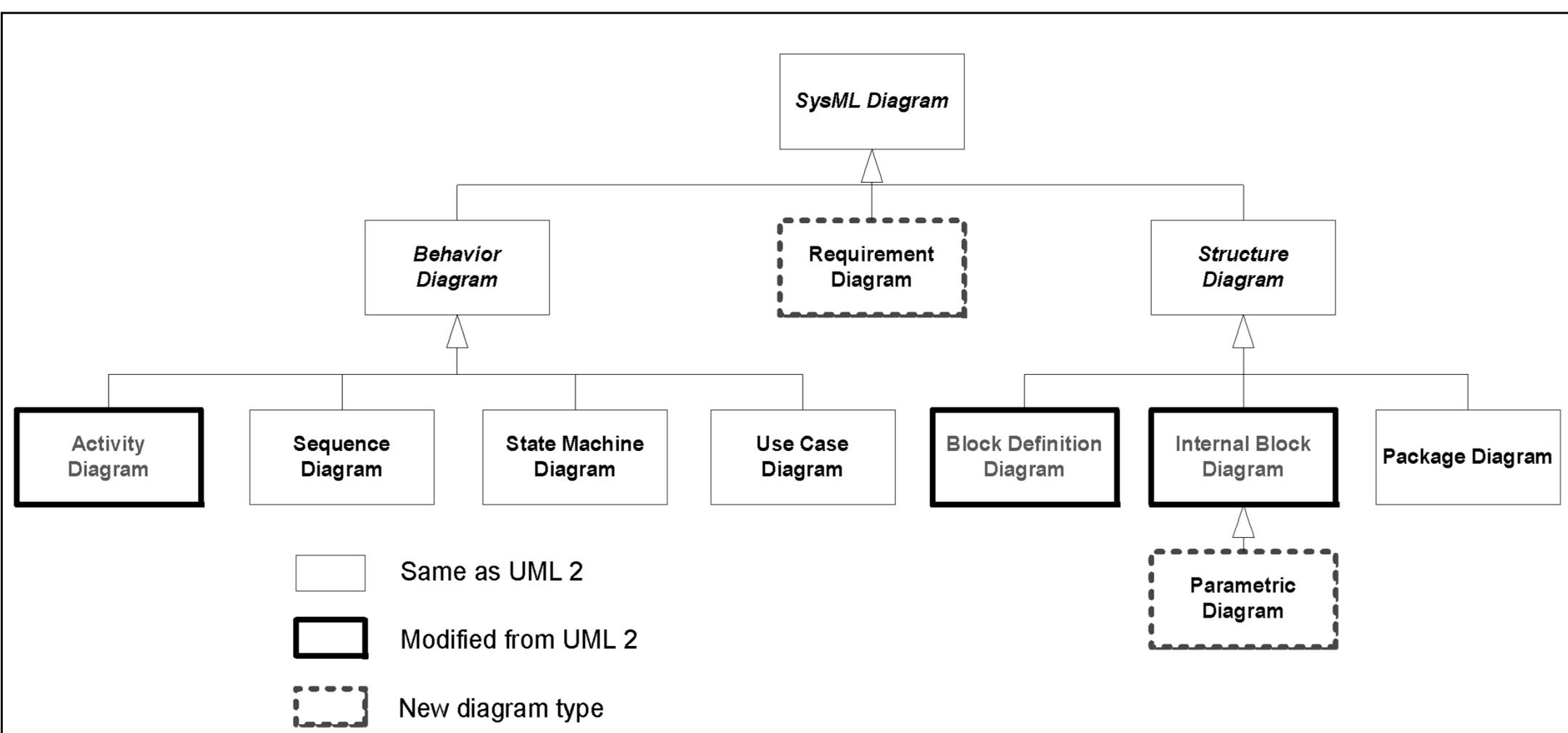
Copyright © 2006-2008 by Object Management Group.



SysML - Pozitionare



Taxonomia SysML



Blocuri (1)

- Blocurile sunt elementele structurale de baza (similar cu *clasele* din UML)
 - ◆ implica modelarea blocurilor in loc de modelarea claselor si furnizeaza un vocabular mai potrivit pentru ingineria sistemelor. Un bloc cuprinde software, hardware, date, procese, personal si facilitati.
 - Prezinta un concept unificator de descriere a structurii unui element / sistem
 - ◆ Sistem
 - ◆ *Hardware*
 - ◆ *Software*
 - ◆ Date
 - ◆ Proceduri
 - ◆ Facilitati
 - ◆ Persoane
- «block»
BrakeModulator

allocatedFrom
«activity»Modulate
BrakingForce

values
DutyCycle : Percentage

Blocuri (2)

- Pot exista mai multe “compartimente” ce pot descrie caracteristicile unui bloc:
 - Proprietati
 - Operatii
 - Constrangeri
 - Necesitati indeplinite in cadrul blocului
 - Compartimente definite de useri
- In SysML se face modelarea blocurilor in loc de modelarea claselor si se furnizeaza un vocabular mai potrivit pentru ingineria sistemelor.

Tipuri de proprietati (1)

- Proprietatea reprezinta o caracteristica structurala a unui bloc:
 - ➔ **Proprietati de compositie**
 - Utilizarea unui bloc in contextul unui alt bloc (bloc component)
 - Exemplu - elicea
 - ➔ **Proprietati de referinta**
 - O parte a unui bloc ce nu este in componenta blocului inclus (non-compozitie)
 - Exemplu – agregarea unor componente intr'un subsistem logic

Tipuri de proprietati (2)

- **Proprietate de valoare**

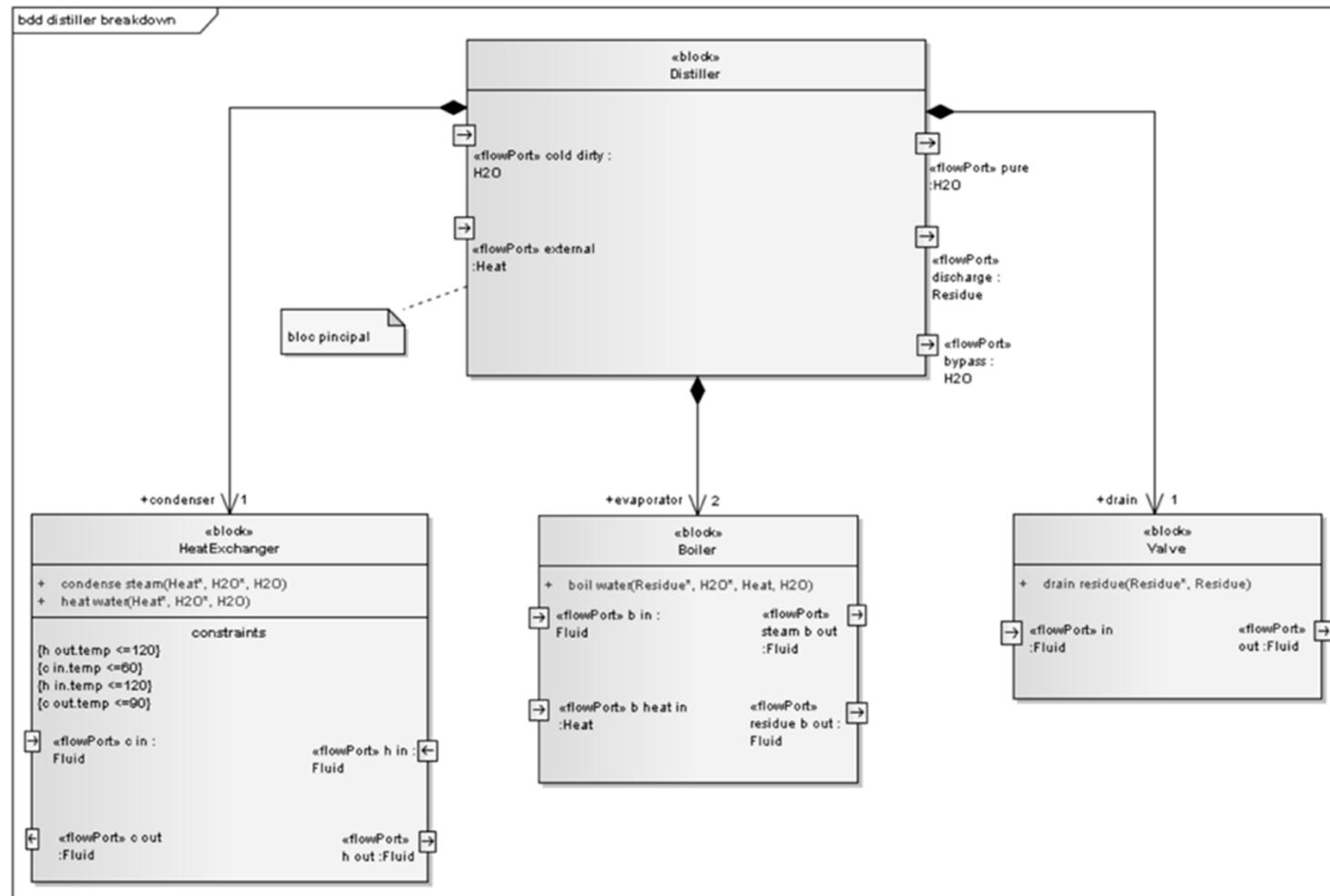
- O proprietate cuantificabila (i.e. unitati, dimensiuni sau distributii de probabilitate)
- Exemplu
 - *Valoare ne-distribuita:* tirePressure:psi=30
 - *Valoare distribuita:* «uniform» {min=28,max=32}
tirePressure:psi

Utilizarea blocurilor

- Se bazeaza pe diagrama de clase UML
 - Suporta caracteristici speciale
- **Diagrama de blocuri** descrie relatiile dintre acestea (compozitie, asociere, specializare)
- Diagramele interne de blocuri descriu structura interna a acestora prin intermediul proprietatilor asociate si a conexiunilor
- Blocurile pot fi caracterizate prin evolutie

Blocurile sunt utilizate pentru specificarea ierarhiilor si interconexiunilor

Exemplu



Exemplu

Blocurile sunt reprezentate sub forma de clase UML.

Blocul principal defineste distilatorul, alcatuit la randul lui din trei tipuri de blocuri:

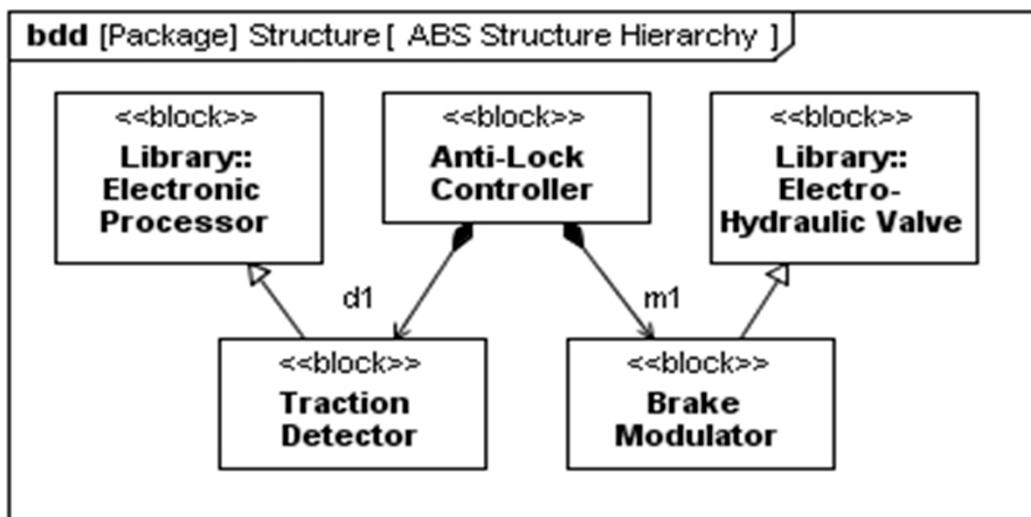
- o Un bloc pentru condensarea vaporilor
- o Un boiler care are rolul de evaporator (multiplicitatea indica ca doua boilere vor fi folosite)
- o O valva care are rolul de scurgere

Aceste blocuri apartin fizic blocului principal, de aceea asocierile din diagrama sunt compozitii reprezentate printr'un romb plin. Daca un bloc ar fi facut parte din blocul principal, dar nu ar fi apartinut fizic acestuia, ar fi fost numit referinta si ar fi fost reprezentat printr'un romb gol.

"Flow port" este o definitie noua a SysML. Acestea marcheaza ce poate trece (in/out) printr'un bloc, indiferent daca este vorba de date, materie sau energie. Diagrama de mai sus arata ca blocul distilatorului primeste ca intrari apa rece si caldura externa.

Definire blocuri vs. utilizare blocuri

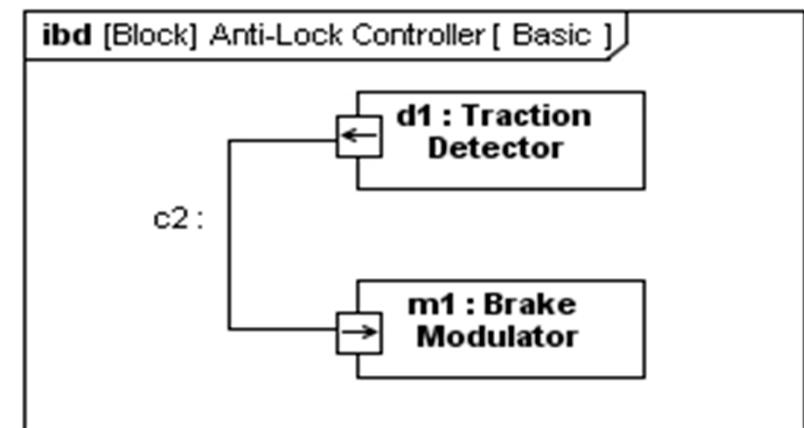
Diagrama definire blocuri



Definitie

- Blocul cuprinde o definitie
- Prezinta proprietati, etc.
- Reutilizare in contexte multiple

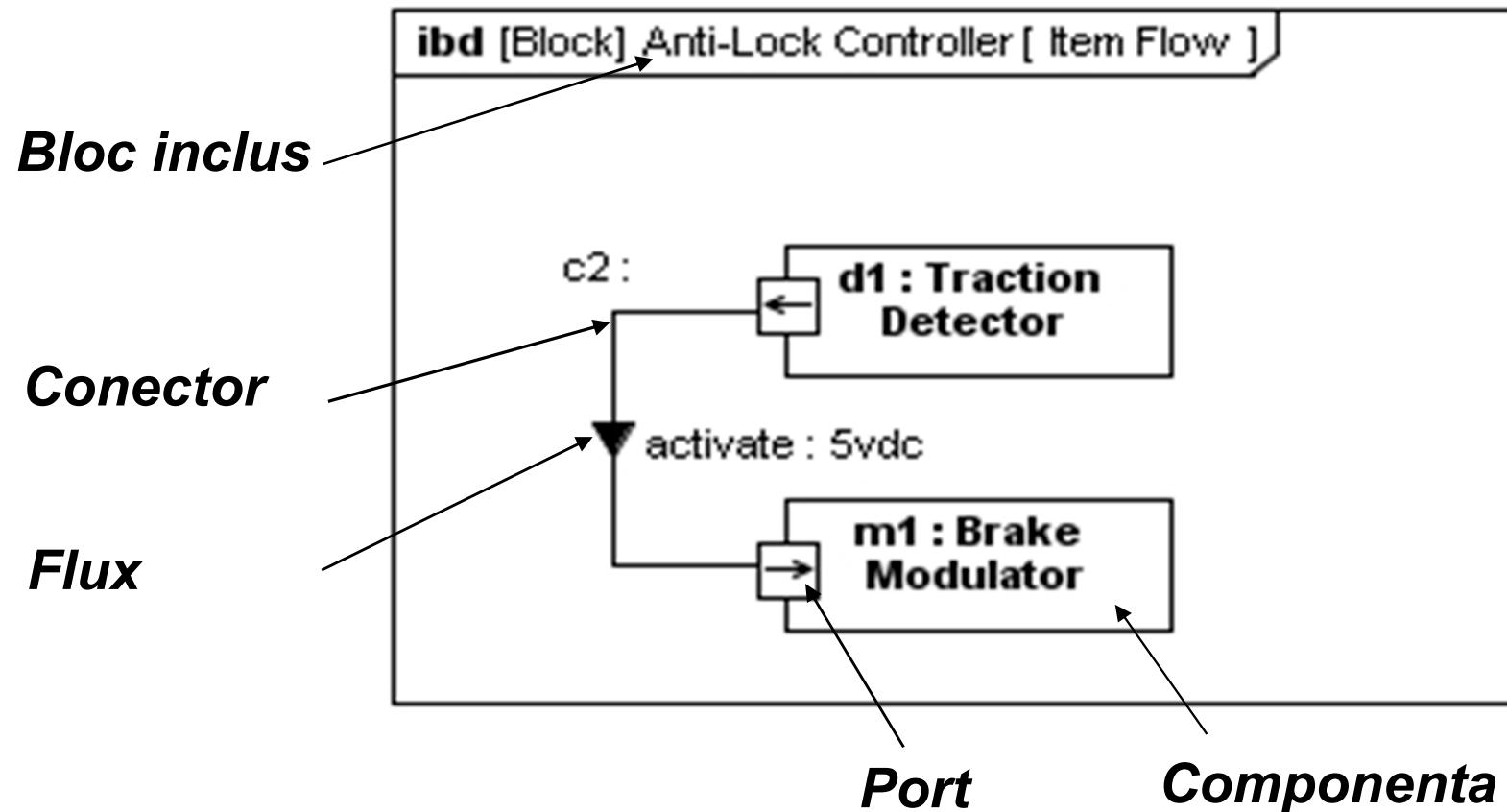
Diagrama interna (de blocuri)



Utilizare

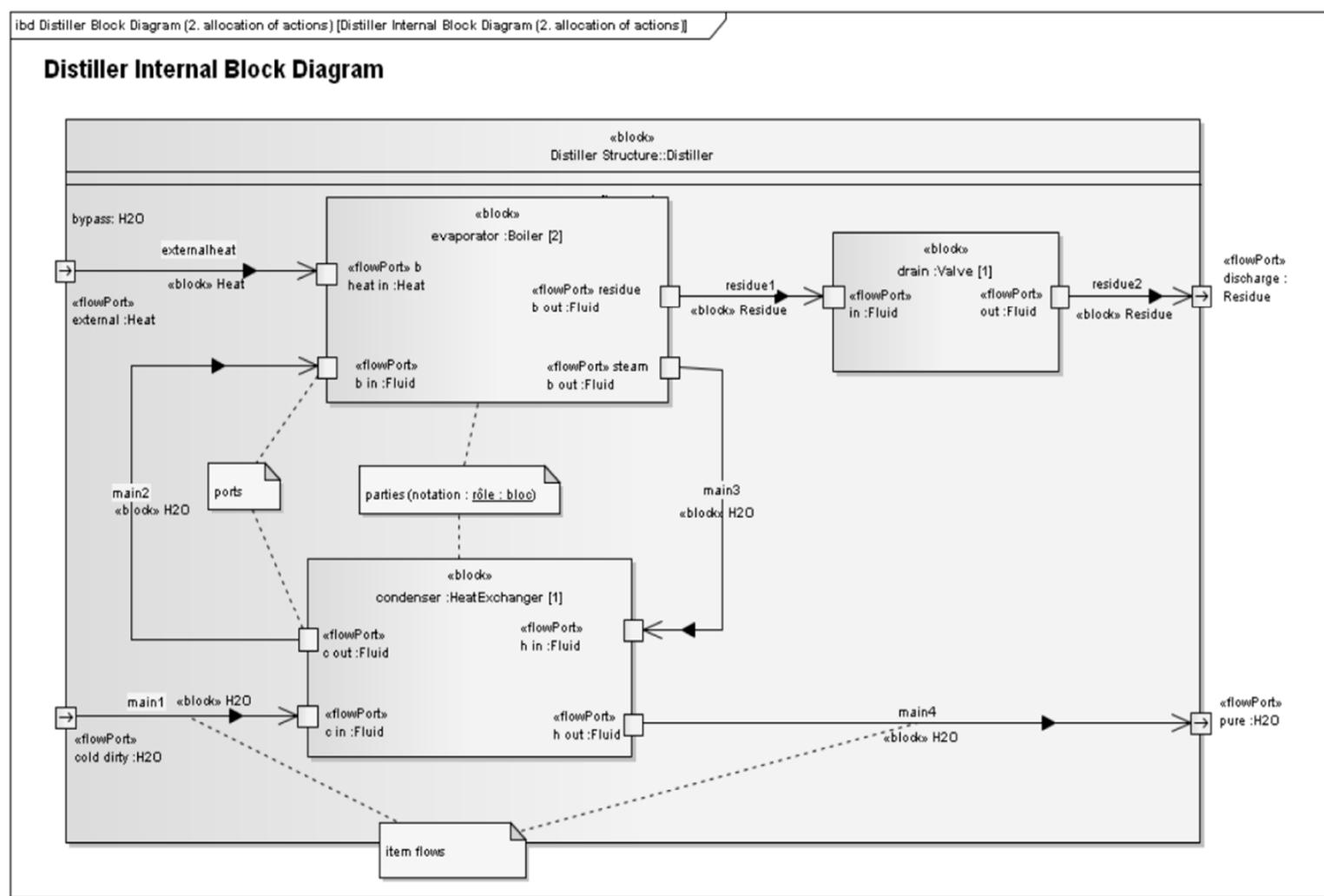
- O parte a unui bloc este utilizata in cadrul unui alt bloc
- Identic cu "rolul"

Diagrama Interna de Blocuri (DIB)



DIB specifică interconexiunea partilor

Exemplu



Exemplu

Blocul distilatorului a fost copiat din diagrama de definire a blocurilor

Blocurile componente specificate in diagrama de definire a blocurilor au fost instantiat ca parti in diagrama interna a blocurilor si sunt denumite astfel: "rol: numele blocului [multiplicitate]".

Multiplicitatea specificata in Diagrama de definire a blocurilor este consecventa cu cea a componentelor din Diagrama interna a blocurilor unde este reprezentata in componenta prin paranteze patrate. Unde multiplicitatea nu este reprezentata se considera ca multiplicitate implicita 1.

Porturile blocului principal sunt asociate cu porturi ale componentelor interne prin conectori. De exemplu apa rece care intra in distilator alimenteaza blocul de condensare.

"Flow port" are o proprietate de directie care poate fi definita ca intrare, iesire, intrare/iesire.

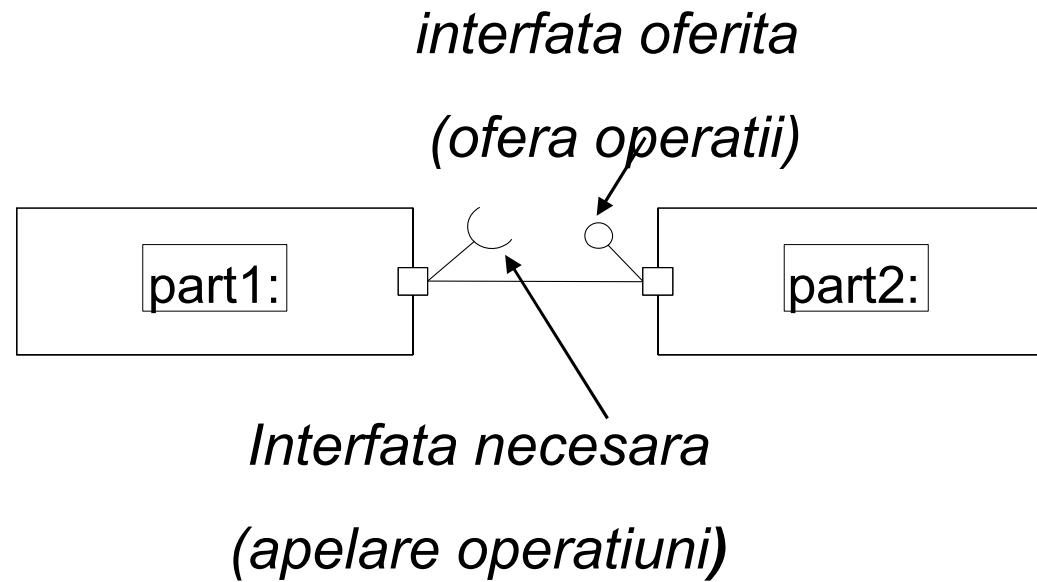
Porturi SysML

- Specifica punctele de interactiuni intre blocuri si partile componente
 - Integreaza structura cu evolutia
 - portName:TypeName
- Tipuri de porturi
 - **Port Standard (UML)**
 - Specifica un set de operatii / semnale necesare sau oferite de bloc
 - Se specifica la nivelul interfetei UML
 - **Port de flux**
 - Specifica fluxul de intrare / iesire dintr'un bloc (sau dintr'o componenta a acestuia)

Porturile standard si porturile de flux suporta concepte diferite de interfata

Notarea porturilor

**Port
Standard**



**Port
de flux**

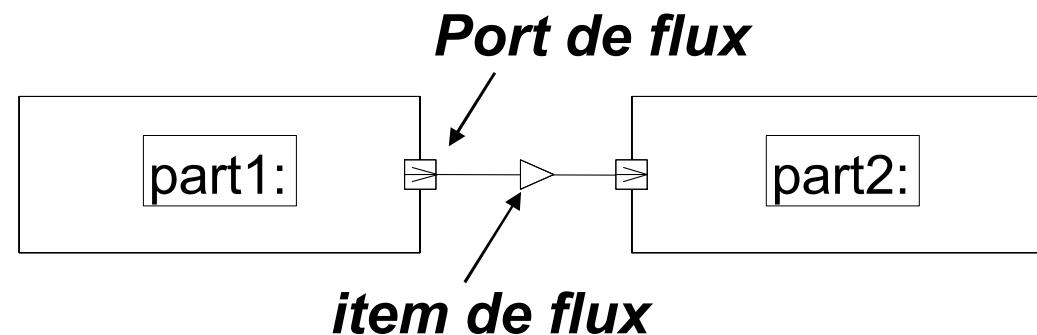
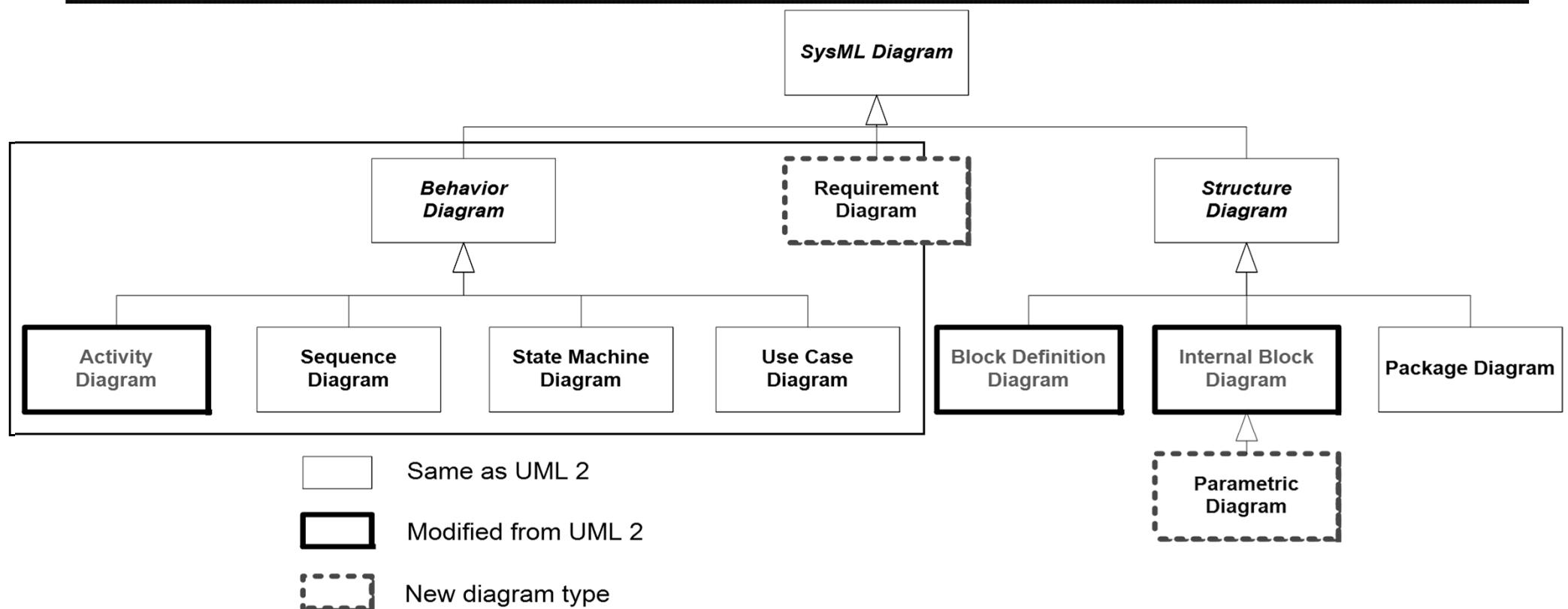


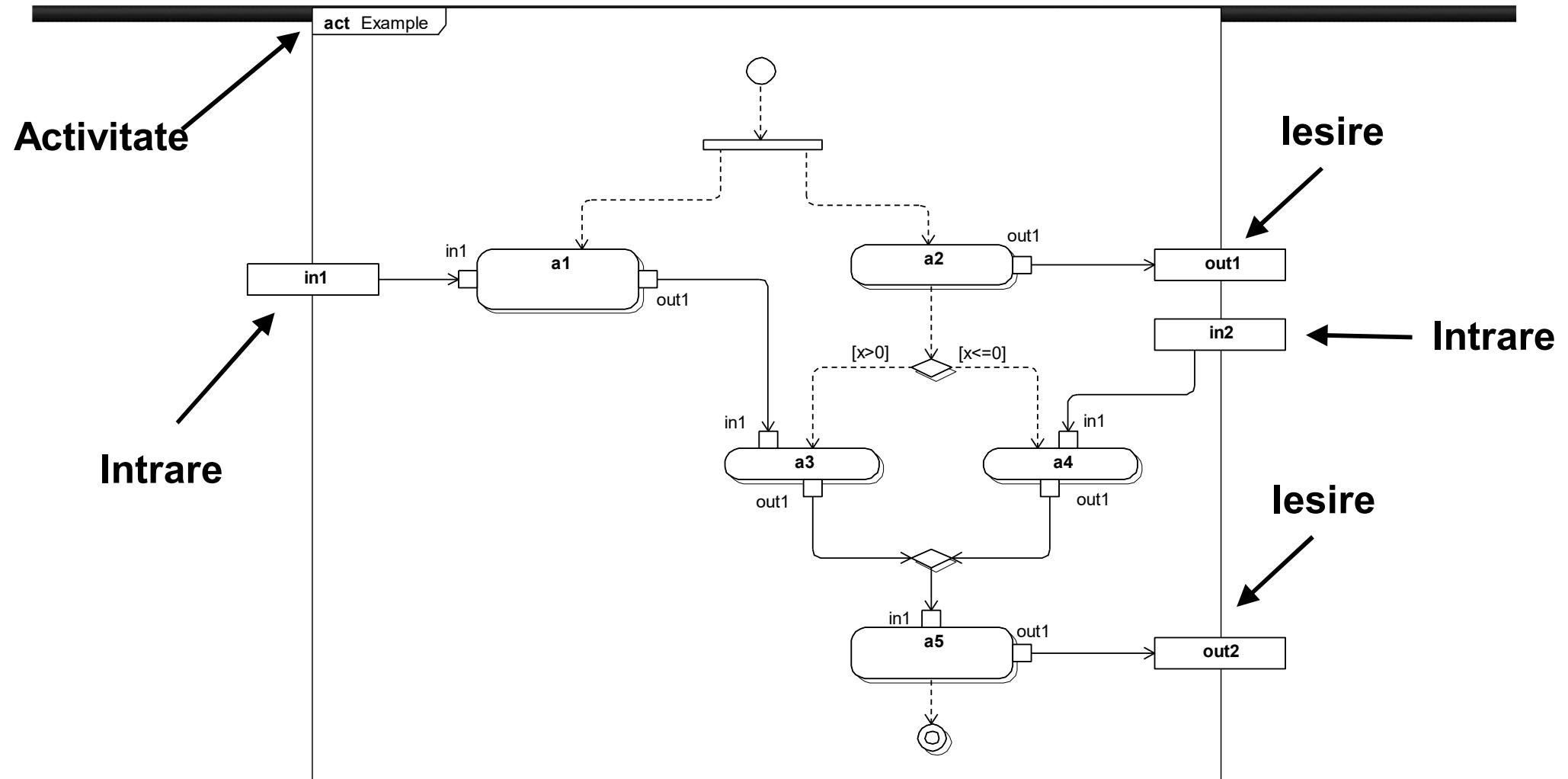
Diagramme de evolutie



Activitati

- **Activitatile** specifica transformarile intrarilor in iesiri prin intermediul unei sechente controlate de actiuni.
- Extensii SysML (ref. activitati):
 - Suport pentru modelarea (continua a) fluxurilor
 - Alinierea activitatilor la **Enhanced Functional Flow Block Diagram**
- Diagrama de activitate reprezinta pasii unui proces, de multe ori folosindu-se de „pini de iesire sau de intrare” care corespund tipului de element cerut ca intrarea unei activitati respectiv elementul generat ca o iesire.
- Daca o actiune sau activitate corespunde unei operatii de bloc, se poate asigura ca tipurile intrarii si iesirii acestei operatii sunt consecvente cu tipurile operatiei de bloc.
- Toate definitiile diagramei de activitate din UML sunt valabile si in SysML.

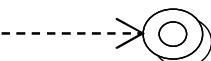
Diagrama de activitati (DA)



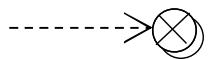
Reprezentari



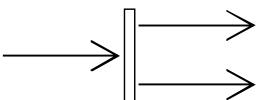
Nod initial



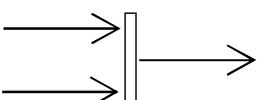
Nod final de activitate



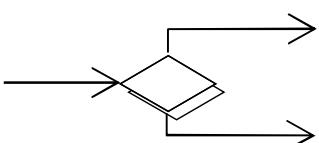
Nod final de flux



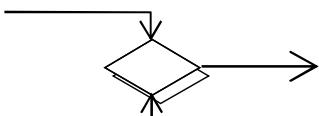
Nod de rascruce



Nod Join

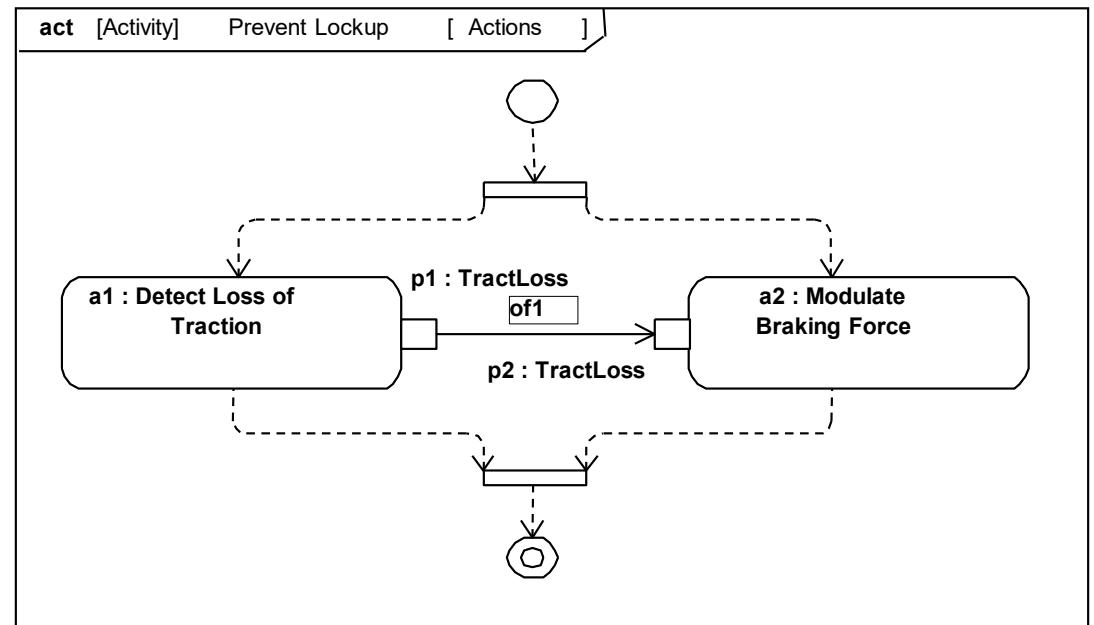
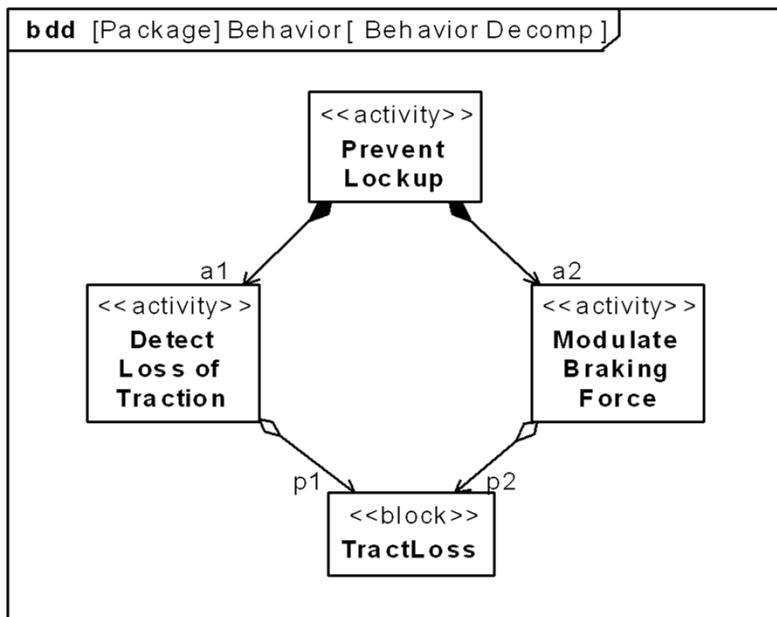


Nod de decizie



Nod de combinare

Descompunerea activitatilor



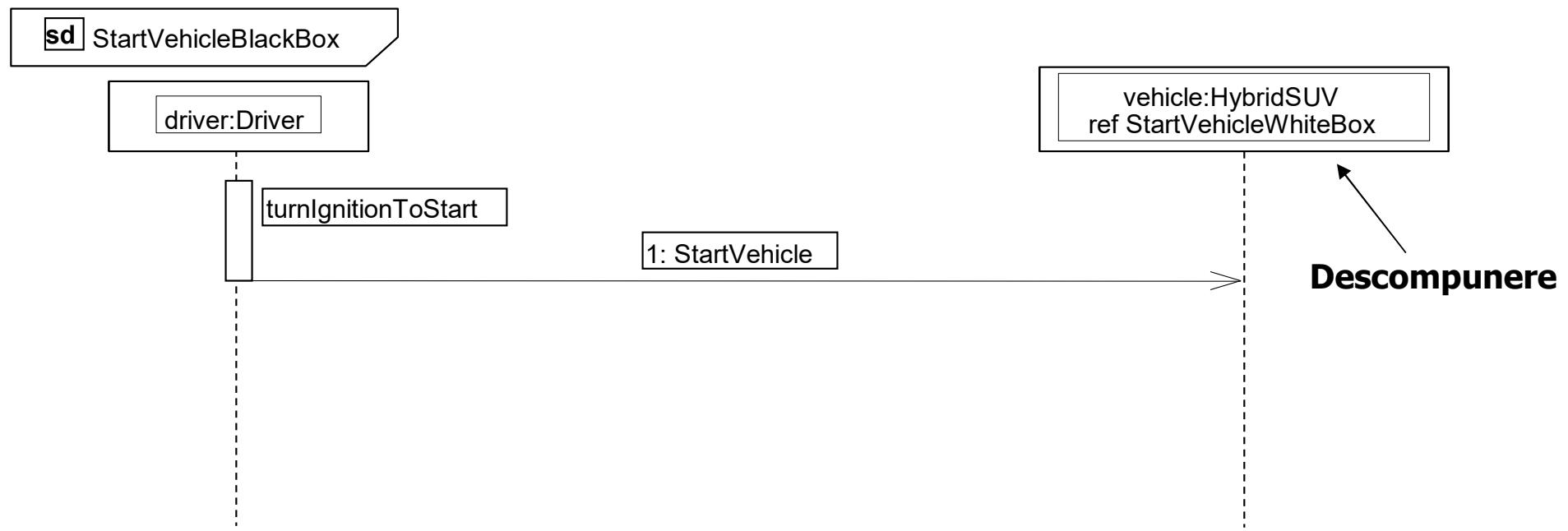
Definire

Utilizare

Interactiuni

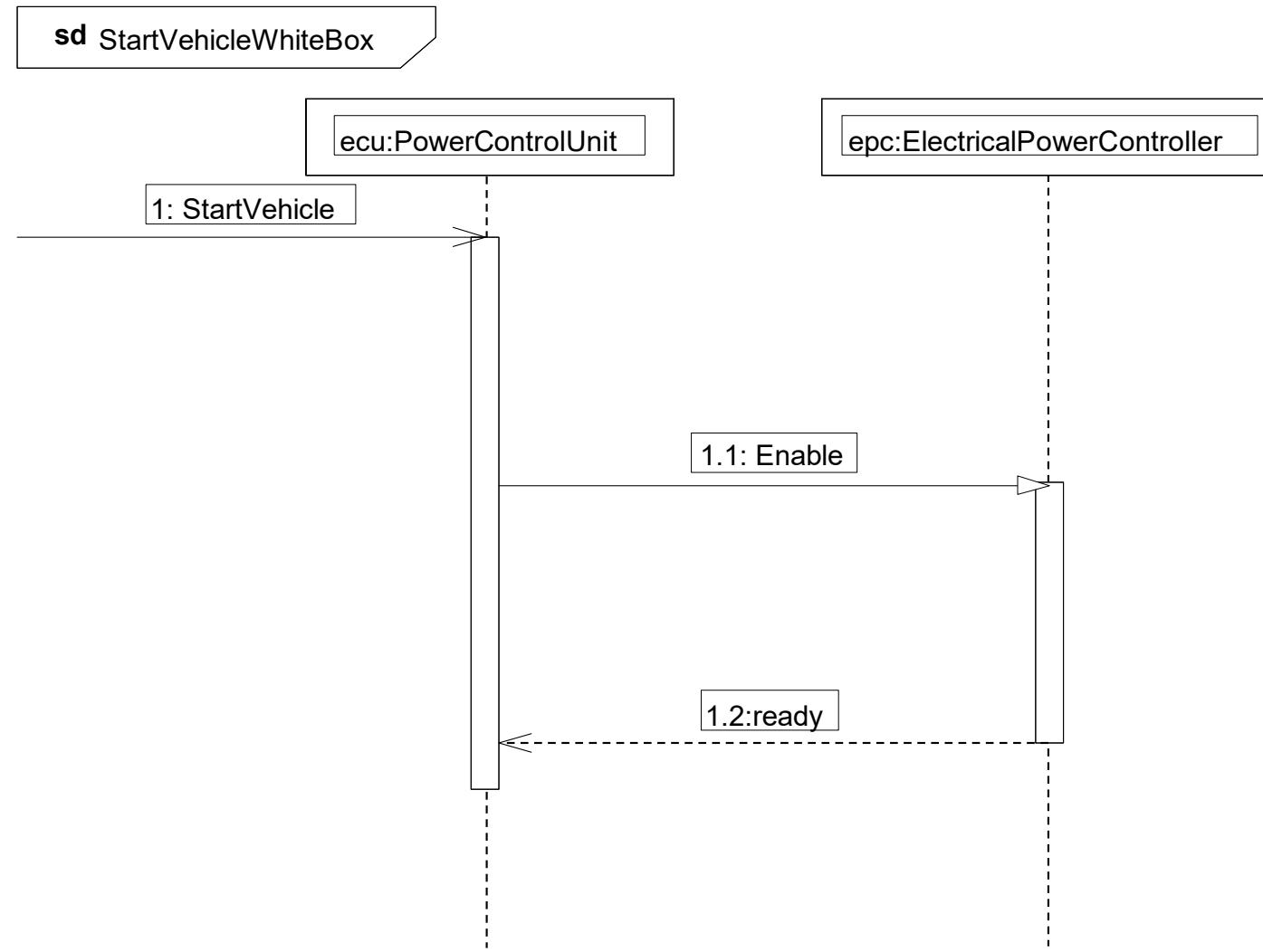
- **Diagrama de secente (DS)** reprezinta evolutia in timp a succesiunii de mesaje
 - Reprezentarea fluxului de control
 - Descrierea interactiunii dintre parti
- **DS** ofera mecanisme de reprezentarea a diverse tipuri de scenarii:
 - secente
 - control logic
 - Etc.
- SysML nu include *timing*, interactiuni sau diagrame de comunicatii

Exemplu (pornire vehicul)



Interactiuni *Black Box*

Exemplu (pornire vehicul)



Operatori de interactiune (1)

- **ref name**
 - Referinta la un fragment de DS definita in alt loc
- **opt [condition]**
 - O componenta va fi executata in functie de o conditie / valoare de stare
- **alt**
 - Are 2 sau mai multe componente; doar una va fi executat in functie de o conditie / valoare de stare
 - Un bloc de functii va fi executat (eticheta [else]) daca nu exista nici o alta conditie care sa fie indeplinita.

Operatori de interactiune (2)

- **par**

- Contine 2 componente ce se executa concurrential:
 - Concurinta NU impune simultaneitate; pur si simplu ordinea nu poate fi determinata: (A then B), (B then A), sau (A and B interleaving) ...

- **loop min..max [escape]**

- Are un numar min de executii, (optional) un numar maxim de executii si (optional) o conditie de iesire

- **break [condition]**

- Conditie de iesire: daca este adevarata, continutul este executat, pana la instructiune; restul nu mai este executat

Operatori de interactiune (3)

- **critical**
 - ➔ In DS exista o regiune critica. Regiunea este considerata a fi atomica.
- **neg**
 - ➔ Fragmentul din DS este interzisa.
- **consider** (list of messages)
ignore (list of messages)
 - ➔ Luate in considerare: Sunt listate mesajele relevante din sevenita
 - ➔ Ignorate: Sunt listate mesajele care "ajung" dar nu sunt "interesante" dpdv al aplicatiei

Cazuri de utilizare

- Ofera posibilitatea descrierii functionalitatii de baza a sistemului si interaciunea cu actorii:
 - Dependente de metodologie
 - Pot fi insotite de *use case descriptions*
- Functionalitatea uzuala poate fi influentata de extensii de tip: «include» sau «extend»
- Fara schimbari fata de UML

Cazurile de utilizare definesc domeniul functional al sistemului.

Exemplu

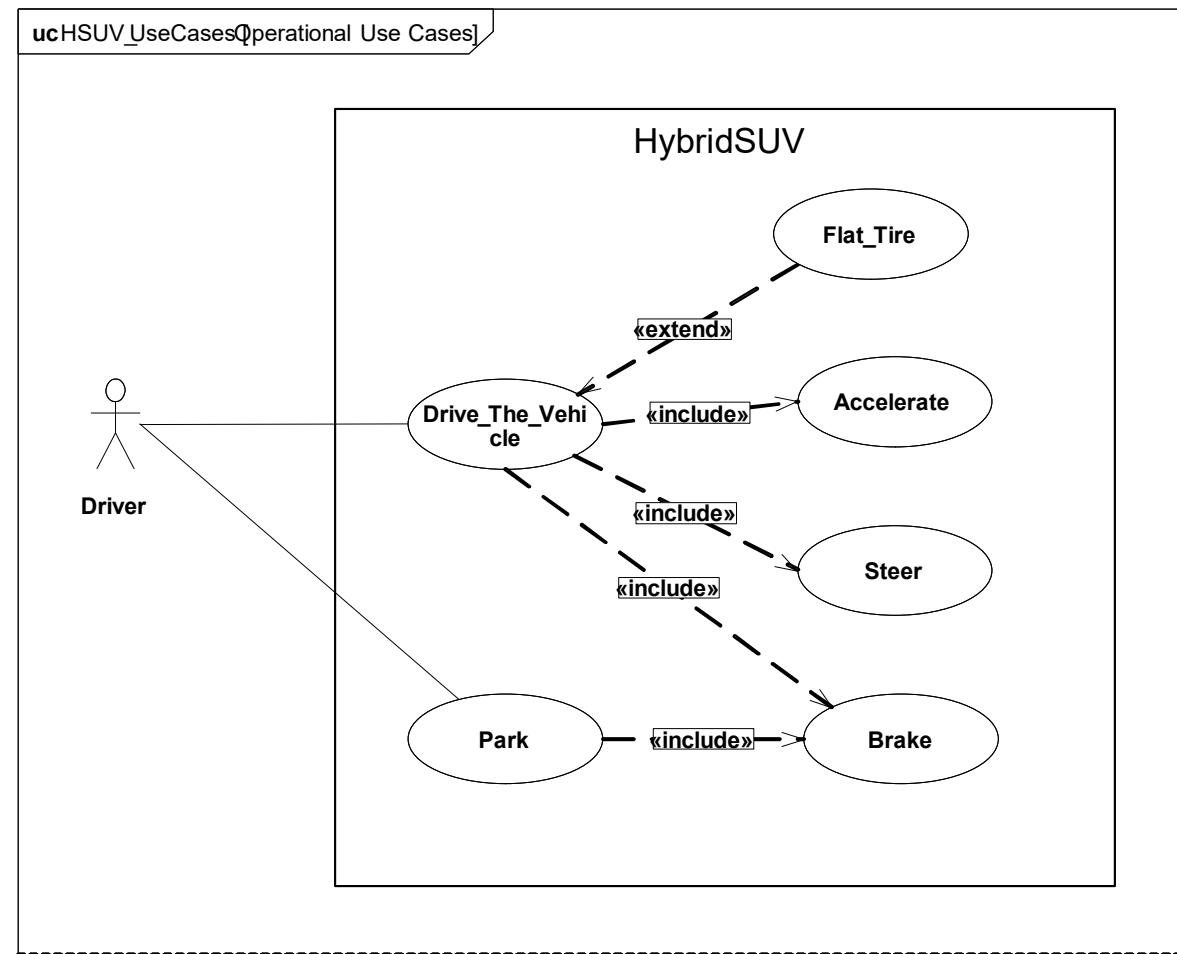
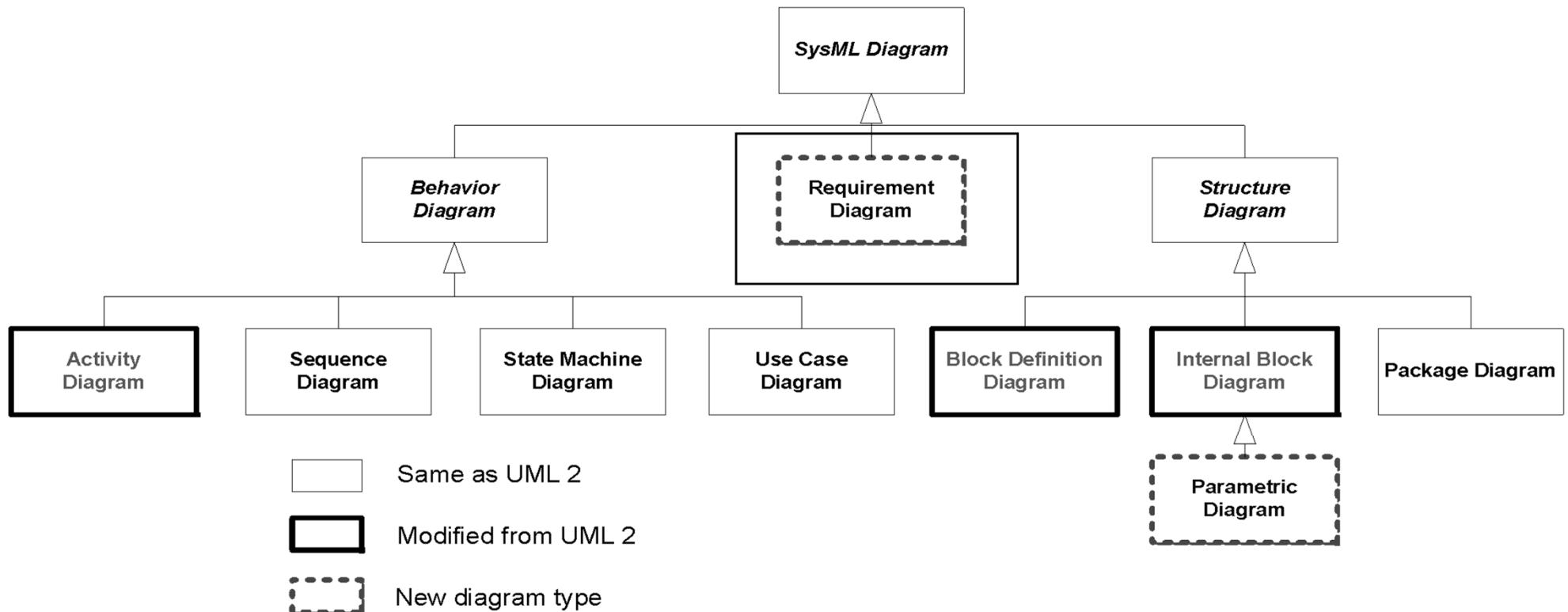


Diagrama de necesitati

- Alocari
- Cerinte



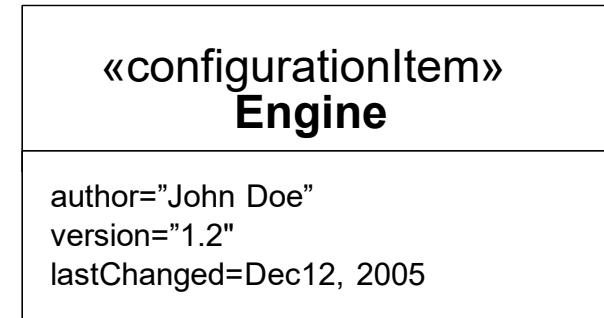
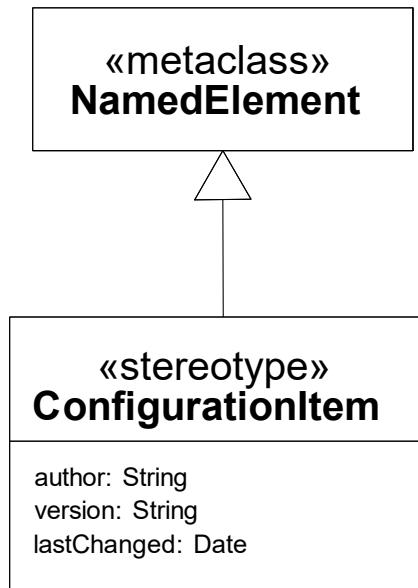
Alocari

- Reprezinta acel tip de relatii ce mapeaza un element (apartenand unui model) cu un altul (dintr'un alt model)
- Tipuri de alocari:
 - Evolutiv (i.e., functie → component)
 - Structural (i.e., logic → fizic)
 - Software → Hardware
 -

Cerinte

- Stereotipul «requirement» reprezinta o solicitare de tip text:
 - Include id si proprietatile textului
 - Pot fi adaugate proprietati definite de utilizator (d.e. metode de verificare)
 - Pot fi adaugate categorii definite de utilizator (d.e. functional, *interface*, performanta)
- Ierarhia de cerinte descrie cerintele continute intr'o specificatie
- Relatiile dintr'o diagrama de cerinte include: *DeriveReqt*, *Satisfy*, *Verify*, *Refine*, *Trace*, *Copy*

Stereotypes

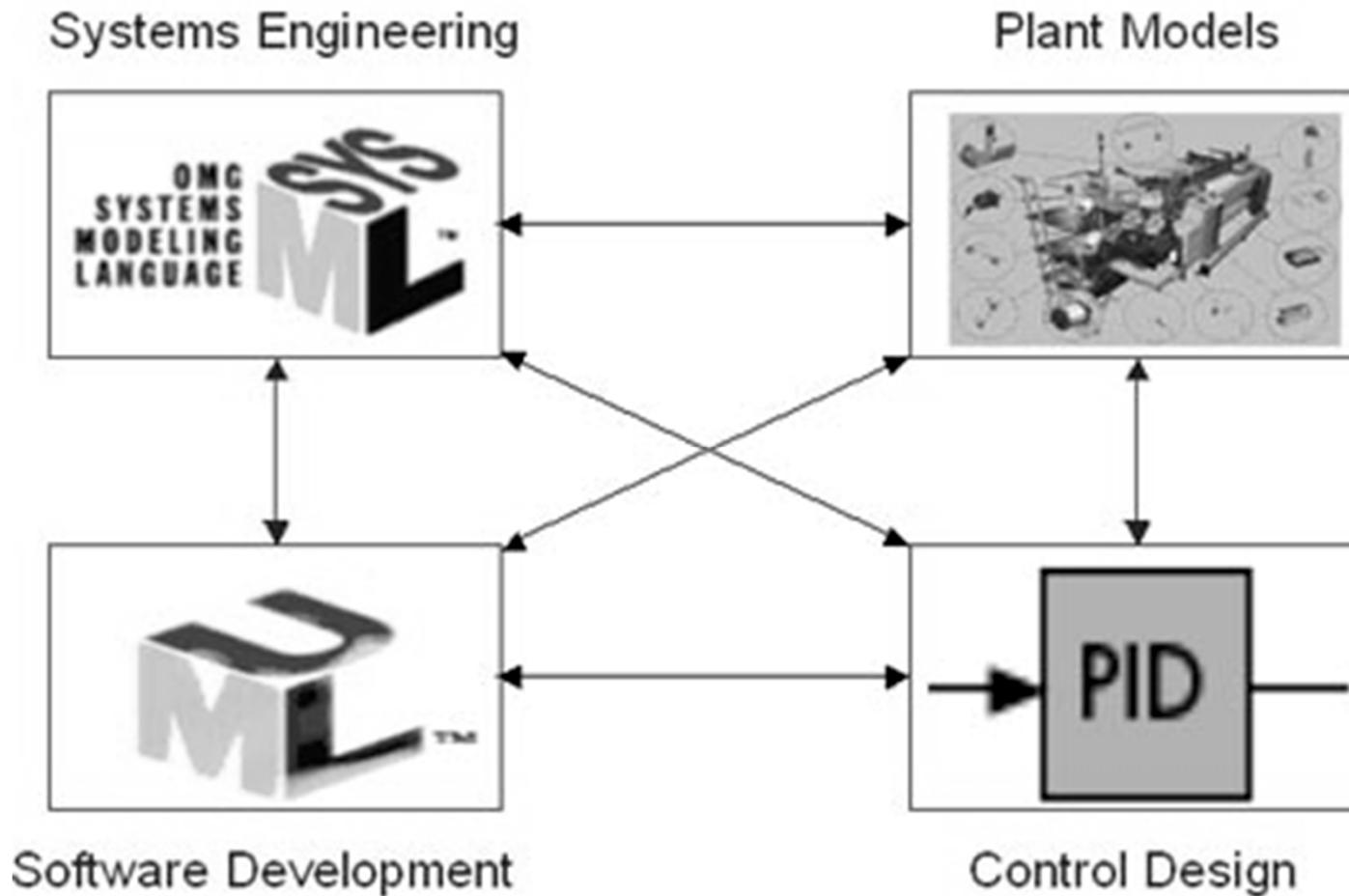


Definire Stereotip

Aplicare Stereotip

Exemplu de aplicatie integrata

Copyright © 2006-2008 by Finn Overgaard Hansen and
Peter Gorm Larsen



UML vs SySML (1)

DIAGRAMA SYSML

Diagrama de activitate (Activity diagram)

Arata comportamentul sistemului, cum controlul si datele decurg. Folositor la analiza functionala. Compara diagrame de blocare al fluxului extins functional Extended Functional Flow Block diagrams (EFFBDs), deja des folosite printre inginerii de sistem.

Diagrama de definire a blocurilor (Block Definition diagram)

Arata structura sistemului in componente cu proprietatile, operatiile si relatiile lor. Folositoare pentru analiza si designul sistemului.

Diagrama de blocuri interne (Internal Block diagram)

Arata structura interna a componentelor, incluzand partile si conectorii lor. Folositoare pentru analiza si designul sistemului.

Diagrama pachet (Package diagram)

Arata organizarea in pachete, vizualizari si puncte de vedere a unui model. Folositor pentru managementul modelului.

SCOP

DIAGRAMA UML ANALOG

Diagrama de activitate (Activity diagram)

Diagrama de clasa (Class diagram)

Diagrama de structuri compuse (Composite Structure diagram)

Diagrama pachet (Package diagram)

UML vs SySML (2)

DIAGRAMA SYML

Diagrama parametrica
(Parametric diagram)

Diagrama cerinta
(Requirement diagram)

Diagrama seventa
(Sequence diagram)

Diagrama de stare al masinii
(State Machine diagram)

SCOP

Arata constrangerile parametrice dintre elementele structurale. Folositoare la analiza performantei si analiza cantitativa.

Arata necesitatile sistemului si relatiile lor cu alte elemente. Folositoare la ingineria necesitatilor.

Arata comportarea sistemului ca si interacțiuni între componentele sistemului. Folositoare la analiza sistemului si design

Arata comportamentul sistemului ca o seventa de stari pe care o componenta sau o actiune o cunoaste ca raspuns unor actiuni. Folositoare la designul sistemului si generarea simularii/codului.

DIAGRAMA UML

N/A

Diagrama seventa
(Sequence diagram)

Diagrama de stare al masinii
(State Machine diagram)

UML vs SySML (3)

DIAGRAMA SYML

**Diagrama cazurilor de utilizare
(Use Case diagram)**

Tabele de alocare*
(Allocation tables)

*tabele deriveate dinamic, nu sunt chiar diagrame

N/A

N/A

SCOP

Arata cerintele functionarii sistemului ca tranzactii care sunt semnificative pentru folositorii sistemului. Folositoare la specificarea cerintelor functionalitatii.

Arata diferite tipuri de alocari (ex. alocari necesare, alocari functionale, alocari structurale). Folositoare pentru facilitarea verificarii si validarii automatizate (V&V) si analize de lipsuri.

DIAGRAMA UML

**Diagrama cazurilor de utilizare
(Use Case diagram)**

N/A

**Diagrama de componente
(Component diagram)**

**Diagrama de comunicare
(Communication diagram)**

UML vs SySML (4)

DIAGRAMA SYML

N/A

N/A

N/A

N/A

SCOP

DIAGRAMA UML

Diagrama de desfasurare (Deployment diagram)

Diagrama de privire generală asupra interacțiunilor (Interaction overview diagram)

Diagrama de obiecte (Object diagram)

Diagrama de timp (Timing diagram)

Concluzii – relatia SysML / UML

- UML este un “General Purpose Modeling Language (GPML)”, în timp ce SysML este un “Domain-Specific Modeling Language (DSML)” fiind definit ca o personalizare a lui UML 2.0
- **Avantaje:**
 - reutilizarea semanticii și notatiilor din UML 2.0
 - SysML personalizează mai bine semantica din ingineria sistemelor (prin adăugarea de două noi diagrame: *Requirement Diagrams* și *Parametric Diagrams*)
 - SysML este “mai mic” dpdv al numărului de diagrame (9 vs. 13)
- **Dezavantaj:**
 - mosteneste multe din problemele din UML (d.e. complexitatea notatiilor și semantica)

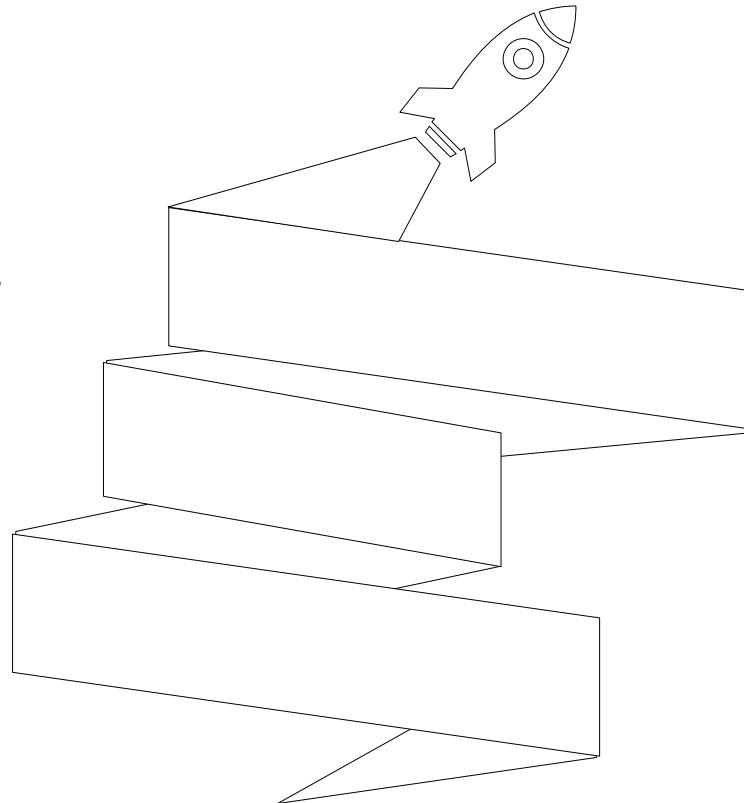
BPMN



Modelare BPMN

BPMN reprezinta o notatie grafica (standard) menita sa defineasca procesele de afaceri dintr-un workflow

Proces de afaceri
Colectie de activitati inrudite ce produce un anumit produs sau serviciu pentru un anumit client



Workflow
Sevente de operatii.
Abstractizarea unei activitati concrete. Sablon de activitate encapsulat intr-un process in vederea integrarii acestuia in sistem

Modelare BPMN

În cadrul diagramelor BPMN, etapele de modelare a unui proces sunt asemănătoare cu cele ale modelării unui proces folosind diagrama de activități UML. Există, totuși, o serie de bune practici în cadrul modelării BPMN care privesc atât etape de definire a procesului, dar și de redactare a diagramei.

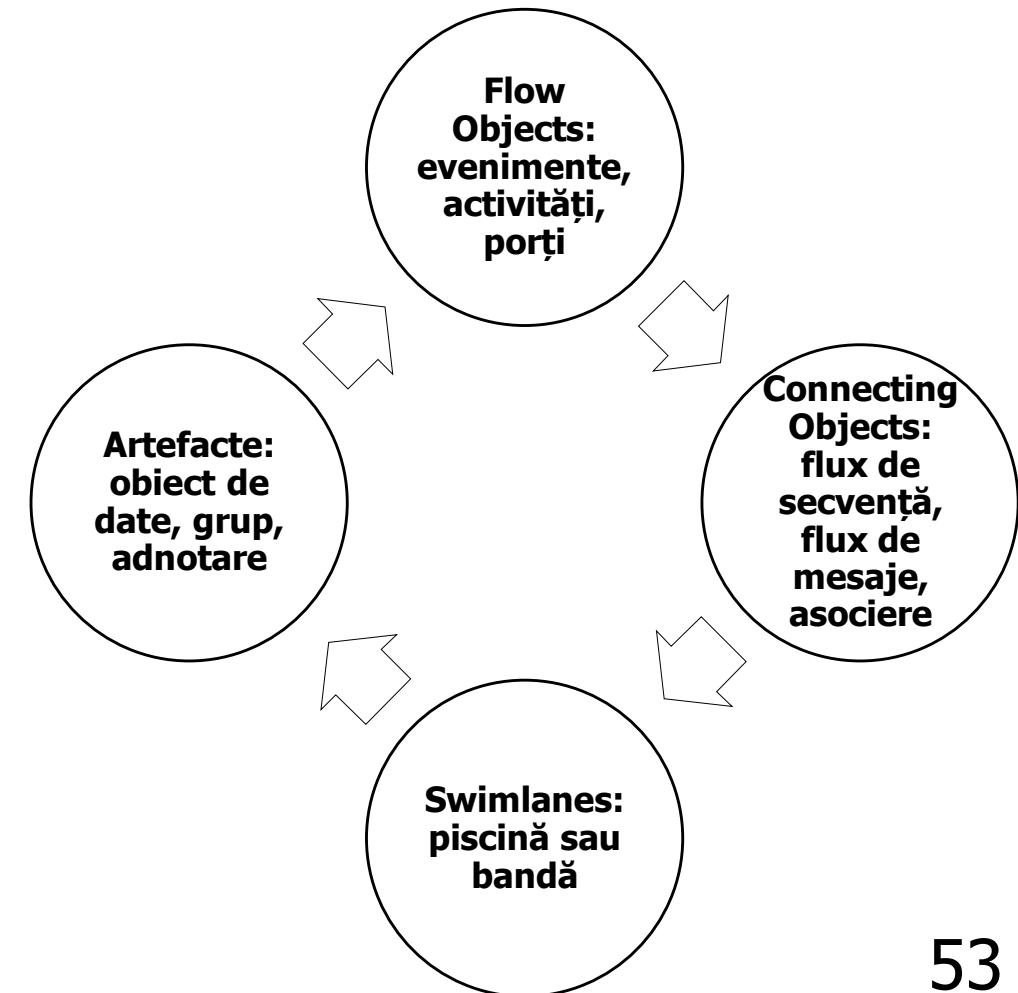
1. Scopul procesului
2. Aspectul diagramei
3. Structura procesului



Modelare BPMN

Modelare de tip **BPMN** este o metodă de diagramă de flux ce modelează pașii unui proces (de afaceri) planificat de la un capăt la altul. Descrie vizual o secvență detaliată a activităților și a fluxurilor de informații necesare pentru a finaliza un proces.

BPMN începe și se termină cu diagrama fluxului procesului. Aceasta reprezinta o hartă tehnică a fluxului și practicilor unei organizații, prezentată într'un limbaj standardizat și disponibilă pentru îmbunătățire, distribuire și urmărire de către utilizatori.



Flow Objects

- **Evenimente:** Un declanșator care pornește, modifică sau finalizează un proces.



- **Activitate:** O anumită activitate sau sarcină efectuată de o persoană sau de un sistem.



- **Poarta de acces (Gateway):** Punct de decizie care poate ajusta calea în funcție de condiții sau evenimente.



Connecting Objects

- **Fluxul secvențial:** Afisează ordinea activităților de efectuat. Este afișat ca o linie dreaptă cu o săgeată. Poate afișa un flux condiționat sau un flux implicit.

- **Fluxul de mesaje:** Înfățișează mesajele care curg prin „pool” sau granițele organizației, cum ar fi departamentele. Este reprezentat de o linie întreruptă cu un cerc la început și o săgeată la sfârșit.

- **Asociere:** Afișat cu o linie punctată, asociază un artefact sau un text unui eveniment, activitate sau gateway.

.....

Artefacte

Informații suplimentare pe care dezvoltatorii le adaugă pentru a aduce un nivel necesar de detaliu diagramei. Există trei tipuri de artefacte: **obiect de date, grup sau adnotare**.

- Un **obiect** de date arată ce date sunt necesare pentru o activitate.
- Un **grup** arată o grupare logică a activităților, dar nu modifică fluxul diagramei.
- O **adnotare** oferă explicații suplimentare unei părți a diagramei.



Exemplu

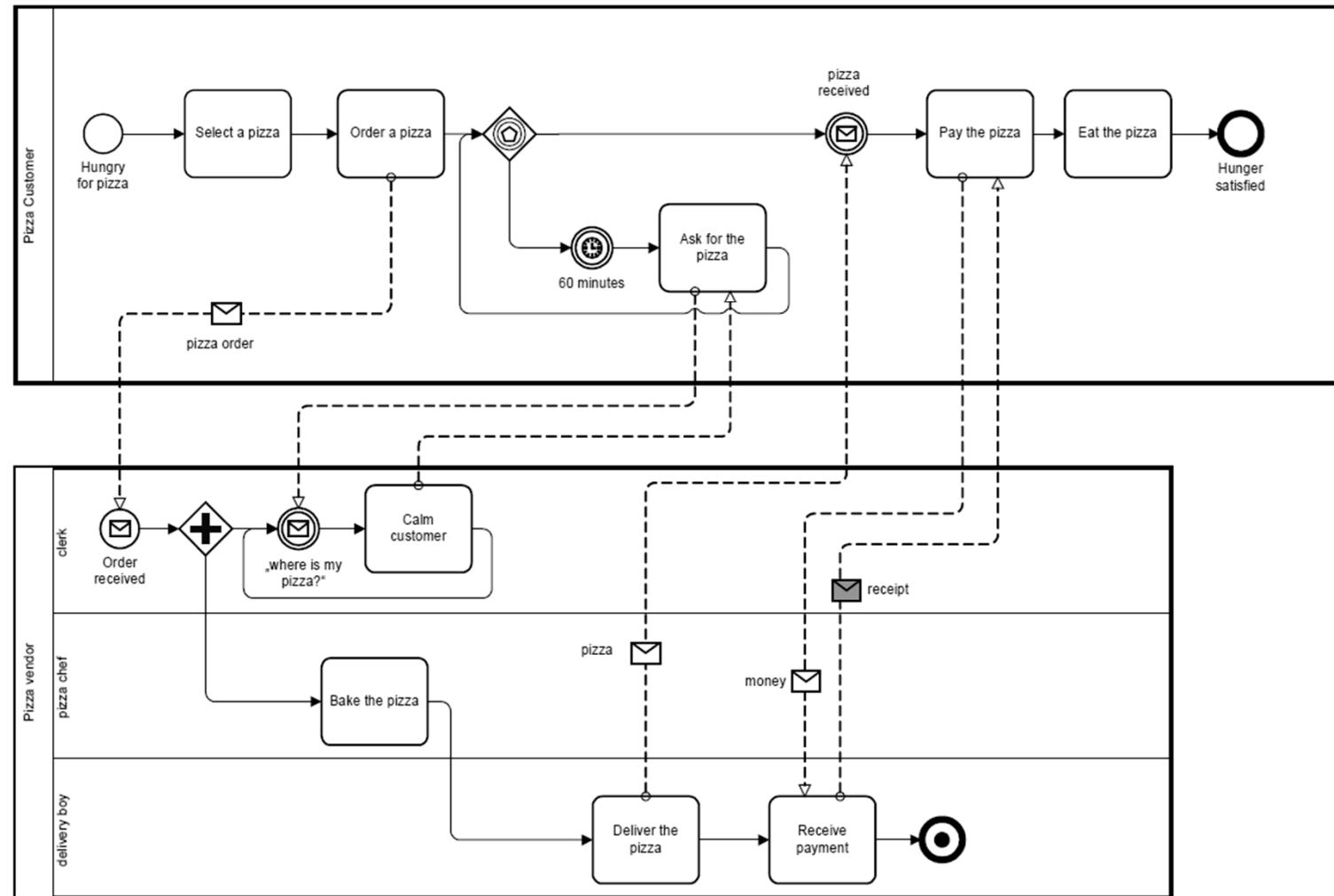
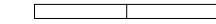


Figure 5.2: Ordering and delivering pizza

Modelare BPMN



Strengths – Puncte tari	Weaknesses – Puncte slabe
<ul style="list-style-type: none">• notare standardizată ISO• număr mare de elemente• capacitatea de a modela diferite tipuri de procese• un număr mare de instrumente de sprijin pentru modelarea și gestionarea proceselor de afaceri• suport industrial bun• schema xml standardizată• capabil de a fi executat• capacitatea de a fi extins și adaptat	<ul style="list-style-type: none">• complexitatea specificației (peste 500 de pagini)• complexitatea notației (peste 100 de elemente vizuale)• interoperabilitate slabă între instrumentele BPMN

BPMN vs UML

UML este un limbaj de modelare orientat pe obiecte, care utilizează o metodă orientată pe obiecte pentru a modela aplicațiile. UML se concentrează pe limbaje standard, mai degrabă decât pe procese standard. Notația UML își propune să dezvolte un meta-model universal și ușor de implementat, care poate unifica semantica și poate construi o notație universală.

Pe de altă parte, BPMN adoptă o metodă orientată spre proces pentru a modela sistemul.

Domenii de aplicare diferite: UML se adresează în primul rând persoanelor care modelează și construiesc sisteme software precum aplicații web și medii cloud. UML permite documentarea unui design care poate fi implementat într-o varietate de limbaje de programare.

BPMN se concentrează pe proiectarea modelelor de proces „așa cum sunt” și „a fi” și este utilizat deopotrivă de analiștii de afaceri, dezvoltatorii IT și oamenii de afaceri. Nu este folosit doar pentru proiectarea și îmbunătățirea sistemelor noi, ci permite și îmbunătățirea proceselor manuale.

Principala diferență dintre UML și BPMN o reprezinta diferența de perspectivă:

UML este orientat pe obiecte, iar BPMN este orientat pe proces. Acest lucru face ca BPMN să fie aplicabil pe scară largă atât în IT, cât și în procese de afaceri, în timp ce UML este mai potrivit pentru dezvoltarea sistemelor IT și mai puțin potrivit pentru îmbunătățirea sistemelor de procese.

IDEF



Limbajul IDEF

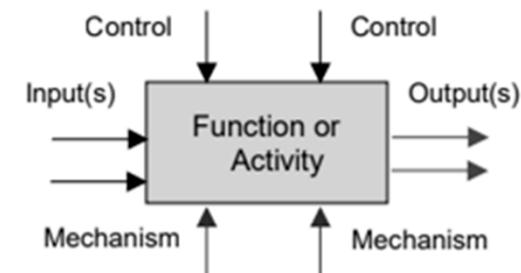
IDEF reprezintă o serie de limbaje de modelare concepute inițial pentru a fi utilizate în domeniul ingineriei software.

Notățiile de modelare IDEF au fost concepute pentru a modela entitățile dintr-o întreprindere cu scopul de a avea o imagine abstractă utilă a diferitelor puncte de vedere (**punctul de vedere al funcției și punctul de vedere al informației**).

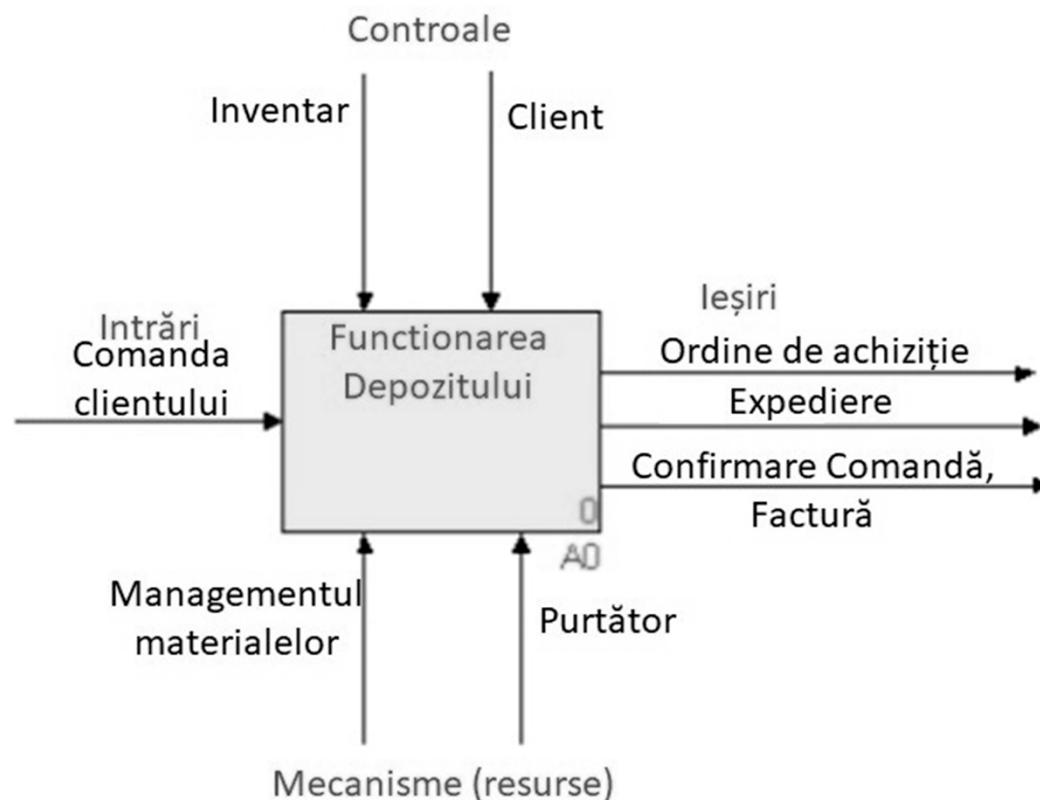
IDEF0 și IDEF3 sunt cele mai potrivite pentru modelarea proceselor

IDEFO

- folosește casete, săgeți și text pentru a comunica procedura de desfășurare a unui proces.
- identifică relațiile **dintre** activități și ofera o vedere a tuturor proceselor dintr'un sistem.
- Sagetile poarta denumirea de ICOM.
- IDEF0 este un set de activități ce preia anumite intrări și, utilizând un anumit mecanism, și supus unor controale, transformă intrările în ieșiri.

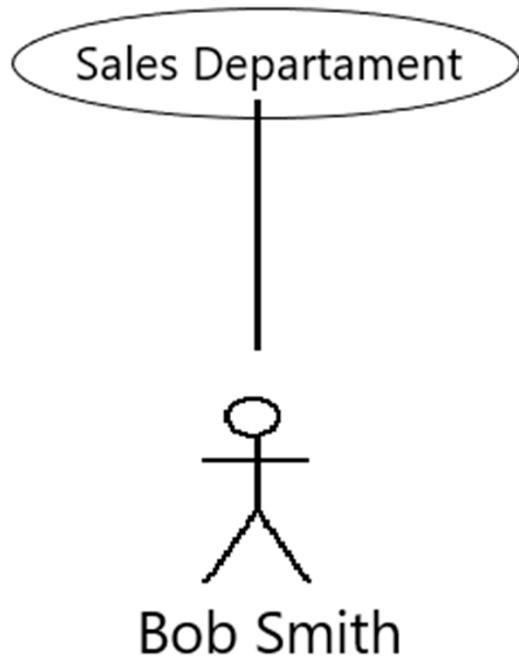


Exemplu

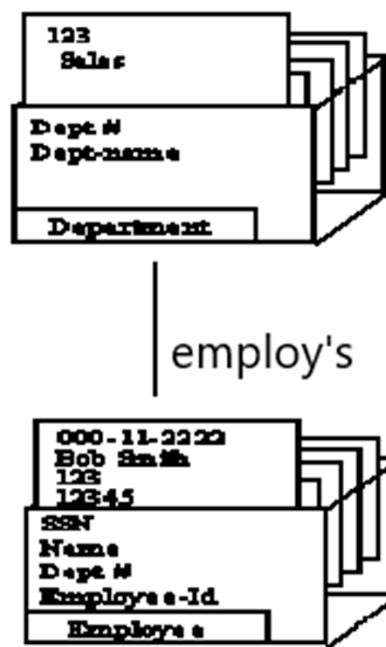


IDEF1

Real World Realm



Information Realm

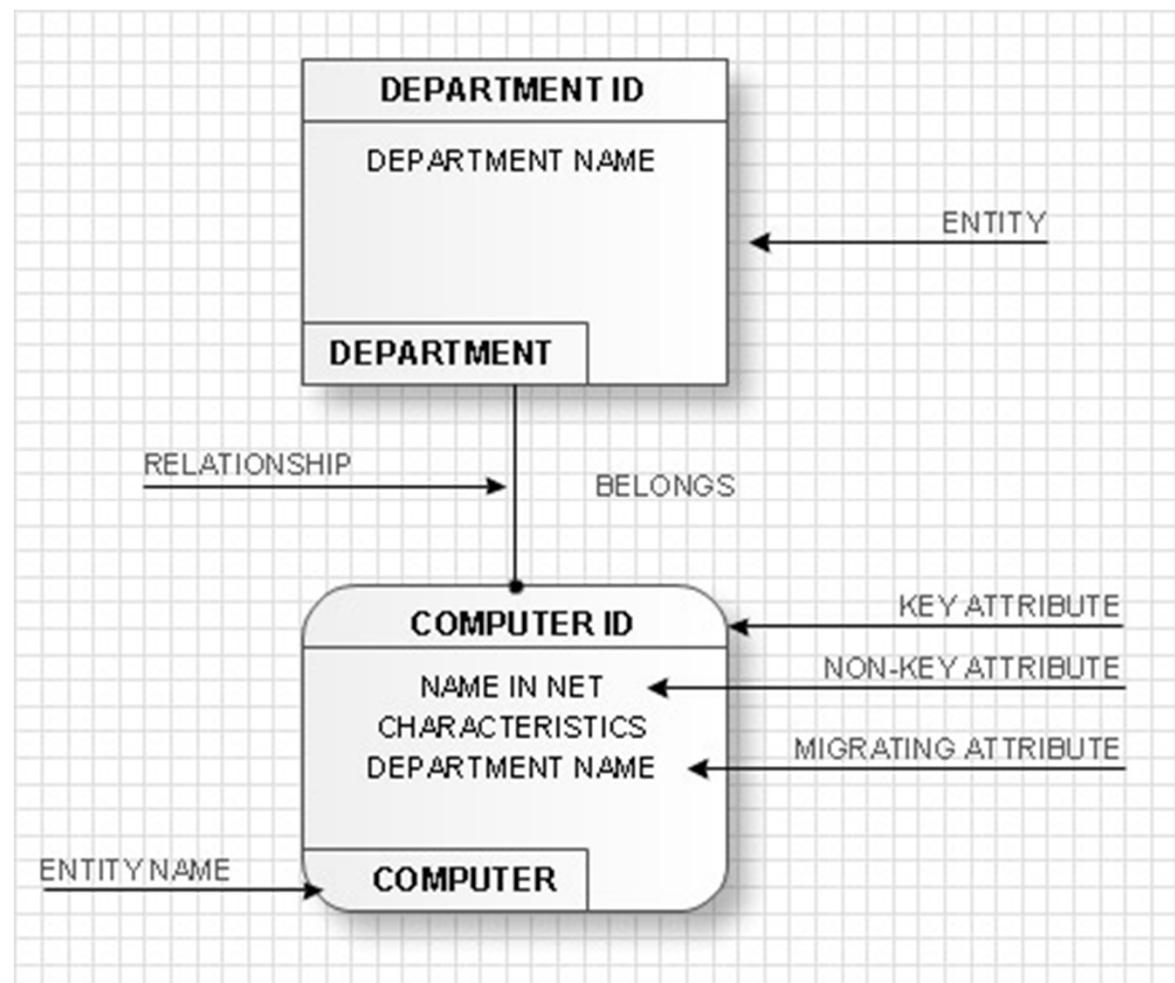


- Principalele concepte **IDEF1** sunt:

- **entitate**: informațiile disponibile într-o organizație despre obiecte fizice sau conceptuale (oameni, idei etc.). Entitatea este înțeleasă ca o imagine informațională.
- **relatii**: asociere între entități (adică imagini informaționale).
- **clase de entitate și relație**: şabloane pentru entitate și relație.

IDEF1x

- Elementele de bază ale IDEF1x sunt **entitatea** (care se referă la o colecție), **atributul** (asociat fiecărui membru al mulțimii) și **structura de clasificare** (pentru modelarea tipurilor de date logice).
- IDEF1x se bazează pe modelul de entitate-relație și este destinat proiectării bazelor de date logice.



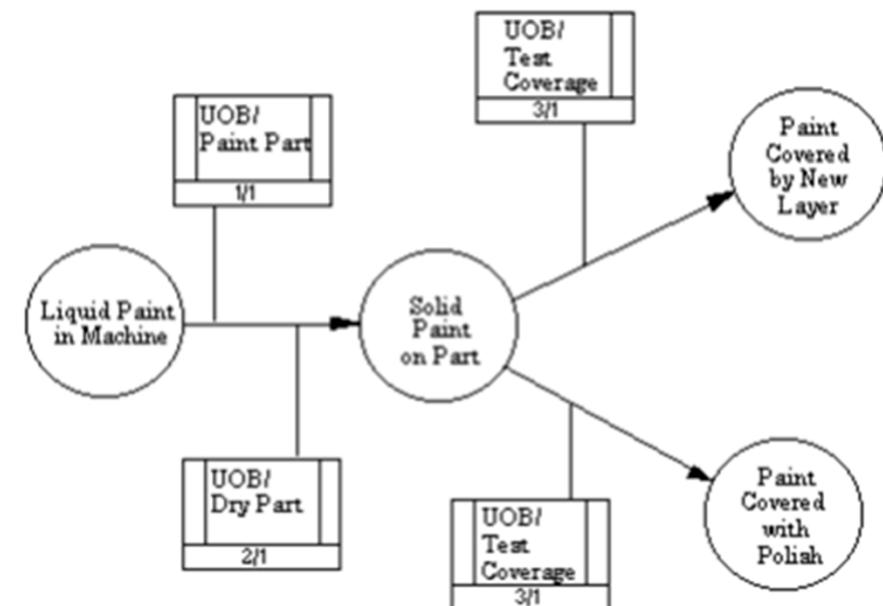
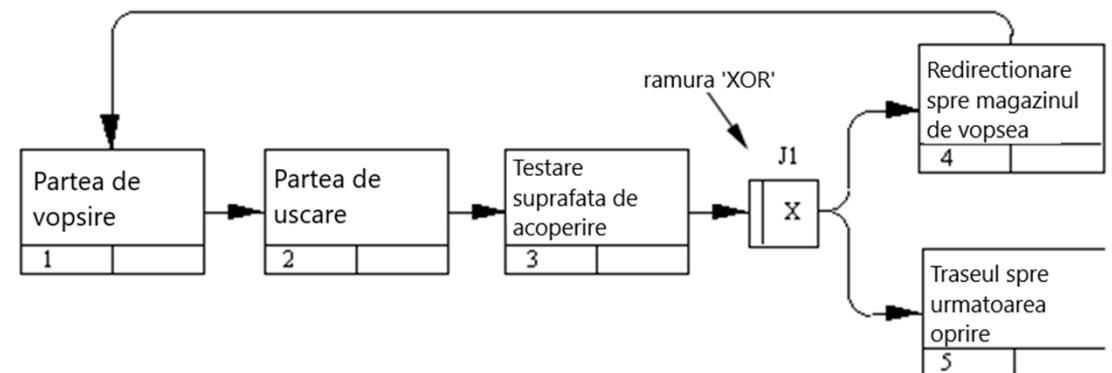
IDEF2

Modelarea prin simulare este un instrument de sprijinire a deciziilor care ajută la rezolvarea problemelor complexe din multe domenii de aplicatie.

Simularea permite construirea scenariilor „ce ar fi dacă” și analizarea posibilelor efecte ale acestora asupra unui sistem existent sau imaginar.

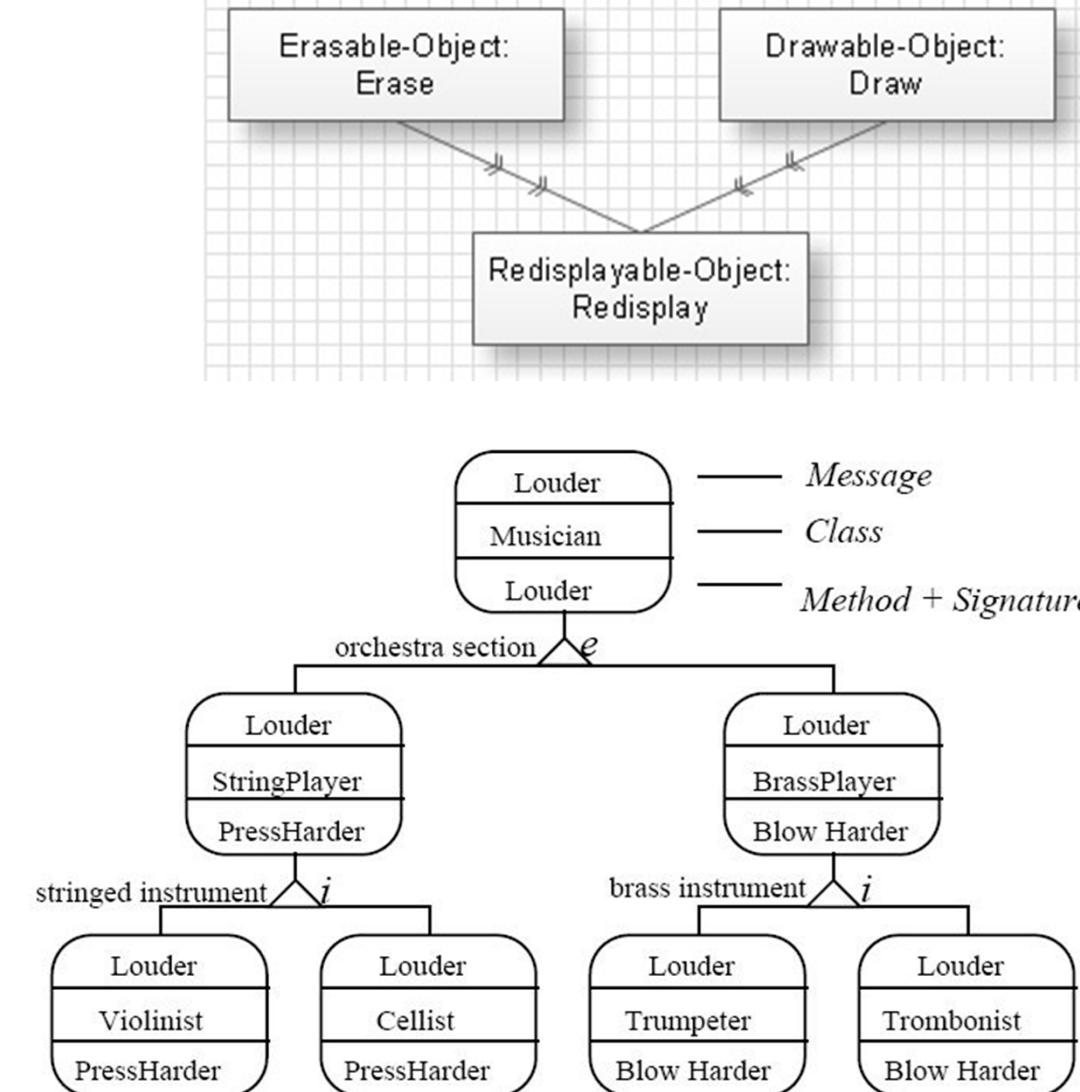
IDEF3

- construiește modele ale proceselor întreprinderii și, prin urmare, este similar cu IDEF0, însă există și o diferență majoră.
- IDEF3** conține două abordări de modelare:
 - Descrierea **fluxului de proces**.
 - Descrierea **tranzitiei stării obiectului**.

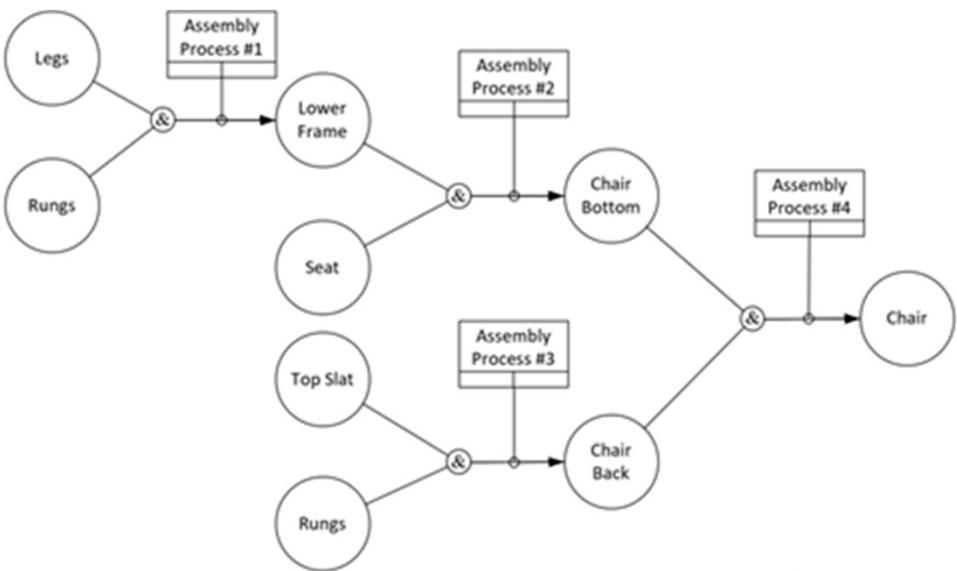


IDEF4

- Metoda de proiectare orientată pe obiecte IDEF4 a fost concepută pentru a ajuta la aplicarea corectă a tehnologiei orientate pe obiecte.
- IDEF4 conține elemente tipice, de ex. submodele de clase și metode, diagrame de moștenire, diagrame de protocol, diagrame client, diagrame de instanțiere etc.



IDEF5



- **IDEF5** este o metodă de inginerie software pentru a dezvolta și menține ontologii de domeniu utilizabile și precise.
- **IDEF5** oferă o metodă de a crea, modifica și menține ontologii, prin intermediul a două limbaje principale: limbajul schematic (grafic) și limbajul de elaborare (textual).

IDEF6 ... IDEF14

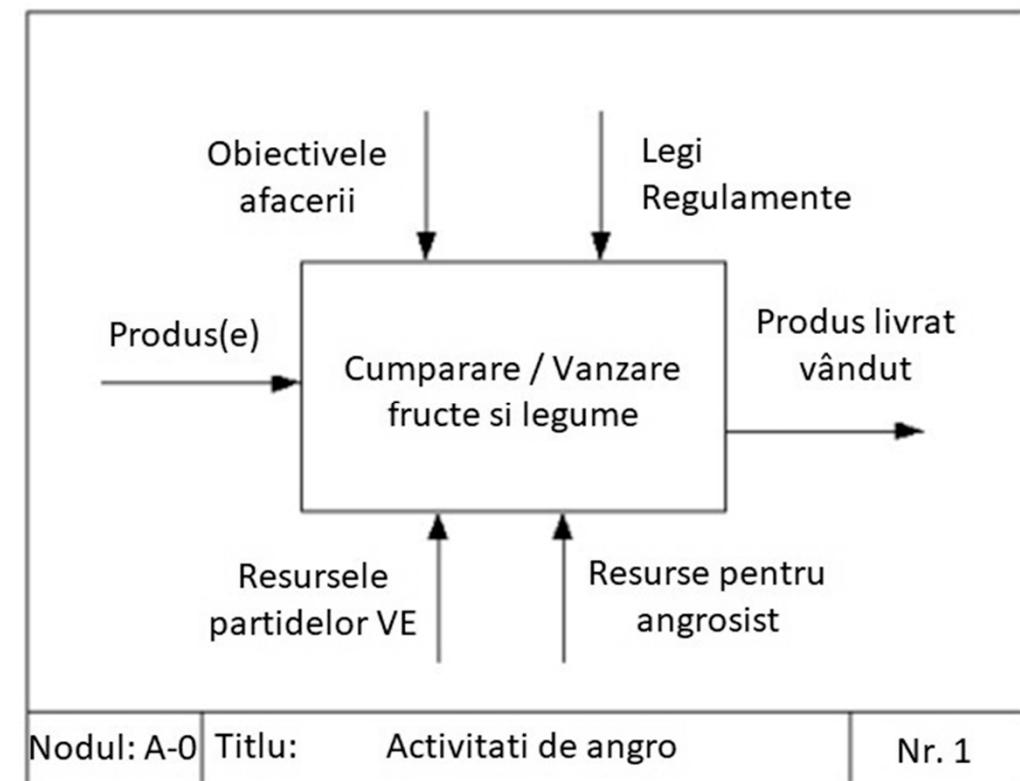
- Ceilalți membri ai familiei IDEF se ocupă cu o gamă largă de modelare de business și software.

In dezvoltare	IDEF6	Motivație de proiectare Captură
	IDEF8	Proiectarea interacțiunii om-sistem
	IDEF9	Descoperirea constrângerilor de afaceri
	IDEF14	Proiectarea rețelei
Avute în vedere	IDEF7	Auditul Sistemului Informatic
	IDEF10	Implem. Arhitectură .Modelare
	IDEF11	Modelarea artefactelor informaționale
	IDEF12	Metoda de proiectare organizațională
	IDEF13	3-Schema Archit Design Method

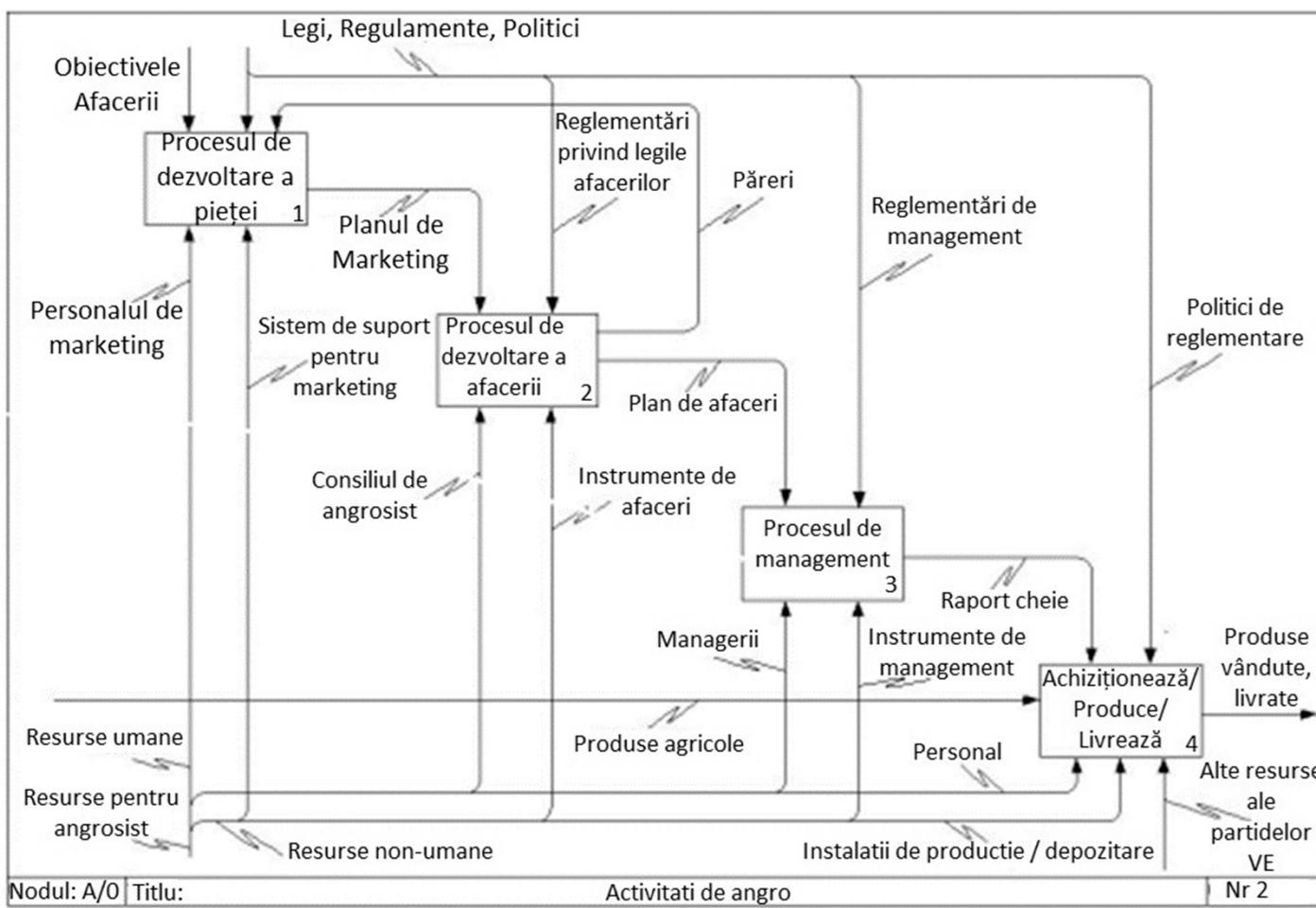
IDEFO (A-0)

Diagramele IDEF0 pot fi descompuse în diagrame de nivel inferior („copil”). Ierarhia este menținută printr-un sistem de numerație care organizează diagramele părinte și copil

Se începe cu diagrama de Nivel 0 și apoi se va detalia în continuare până când se obține un grad suficient de granularitate.

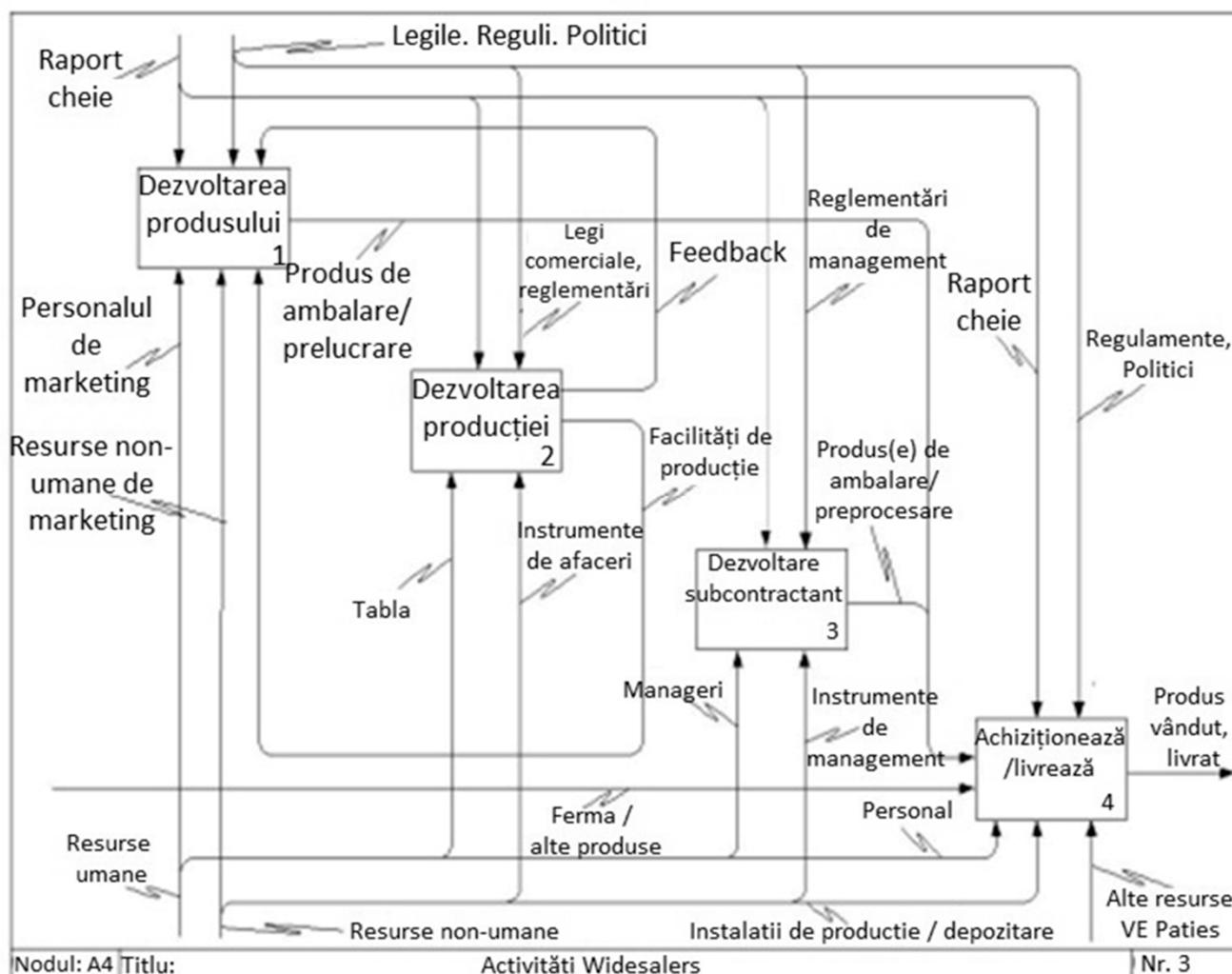


IDEFO (A-1)



Următorul nivel în jos din diagrama A-0 este diagrama A0, care oferă un grad mai mare de granularitate a procesului. Principalele procese de afaceri sunt identificate aici, împreună cu ICOMurile lor (Intrare (stânga), Ieșiri (dreapta), Controle (sus) și Mecanisme (jos)).

IDEFO (A-2)



Pornind de la diagrama A-0, orice funcție (caseta de pe diagramă) poate fi selectată pentru detalii suplimentare.

Procesul #4
(Procure/Producere/Livrare)
a fost ales pentru că
promite a fi cel mai
dinamic și mai interesant.

Comparatie

IDEF0 nu reprezintă condițiile necesare pentru a intra sau a ieși dintr'un proces. De aceea, IDEF0 este cel mai bine utilizat în combinație cu alte metode IDEF (de exemplu IDEF3).

IDEF1 nu poate fi utilizat direct în faza de implementare. Totuși, poate fi extrem de util în modelarea informațiilor din cadrul afacerii, fără constrângeri de implementare.

IDEF1x este un instrument bun pentru baza de proiectare a bazei de date, dar nu urmează regulile unui design grafic bun - simbolurile sale nu se mapează în mod corespunzător la conceptele pe care ar trebui să le modeleze.

IDEF3 este o metodă de captare a descrierilor și este concepută pentru a tolera descrierile parțiale și inconsecvențe. Foarte des, aceste inconsecvențe devin rădăcina problemelor unei organizații și ar trebui abordate în mod corespunzător în reprezentarea IDEF3 și nu neglijate.

IDEF vs. UML

- **IDEF0** poate găsi un echivalent în diagramele de activitate UML folosind extensiile de afaceri specializate pentru modelarea proceselor.
- **IDEF1** și **IDEF1x** sunt similare cu diagramele de clase și obiecte UML care exprimă aspectul static (arhitectural) al unui sistem.
- **IDEF2** poate fi echivalent - într-o măsură - cu diagramele de colaborare, diagramele de activitate și stare în limbajul UML.
- Diagrama de descriere a procesului **IDEF 3** este similară cu diagrama de activitate UML, în timp ce diagrama de tranziție a stării obiectului este similară cu diagramele de stare. IDEF3 este, de asemenea, similar cu diagrama de caz de utilizare privind conceptul de scenariu.
- **IDEF4** este orientat în mod special către dezvoltarea de software și, ca atare, are asemănări cu diagramele de clasă, obiect, activitate și stare din UML.
- **IDEF 5** poate fi comparat doar cu meta-meta-modelul (și poate cu nivelul meta-modelului) în UML, unde este definită ontologia limbajului.