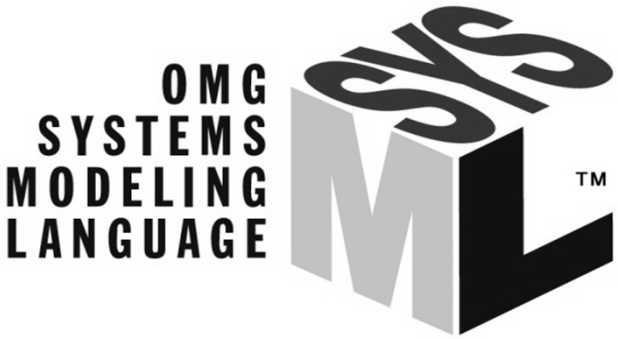
M. Caramihai, © 2020

PROGRAMAREA ORIENTATA OBIECT

CURS 6

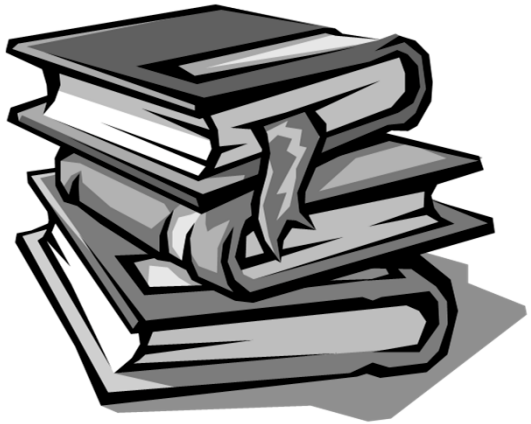
SysML *vs* UML



O parte din suportul de curs a fost realizat cu ajutorul studentului Tudor Nicolau

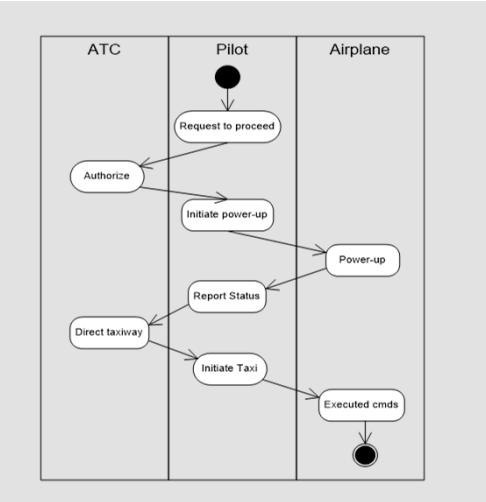
Modul de descriere a sistemelor

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

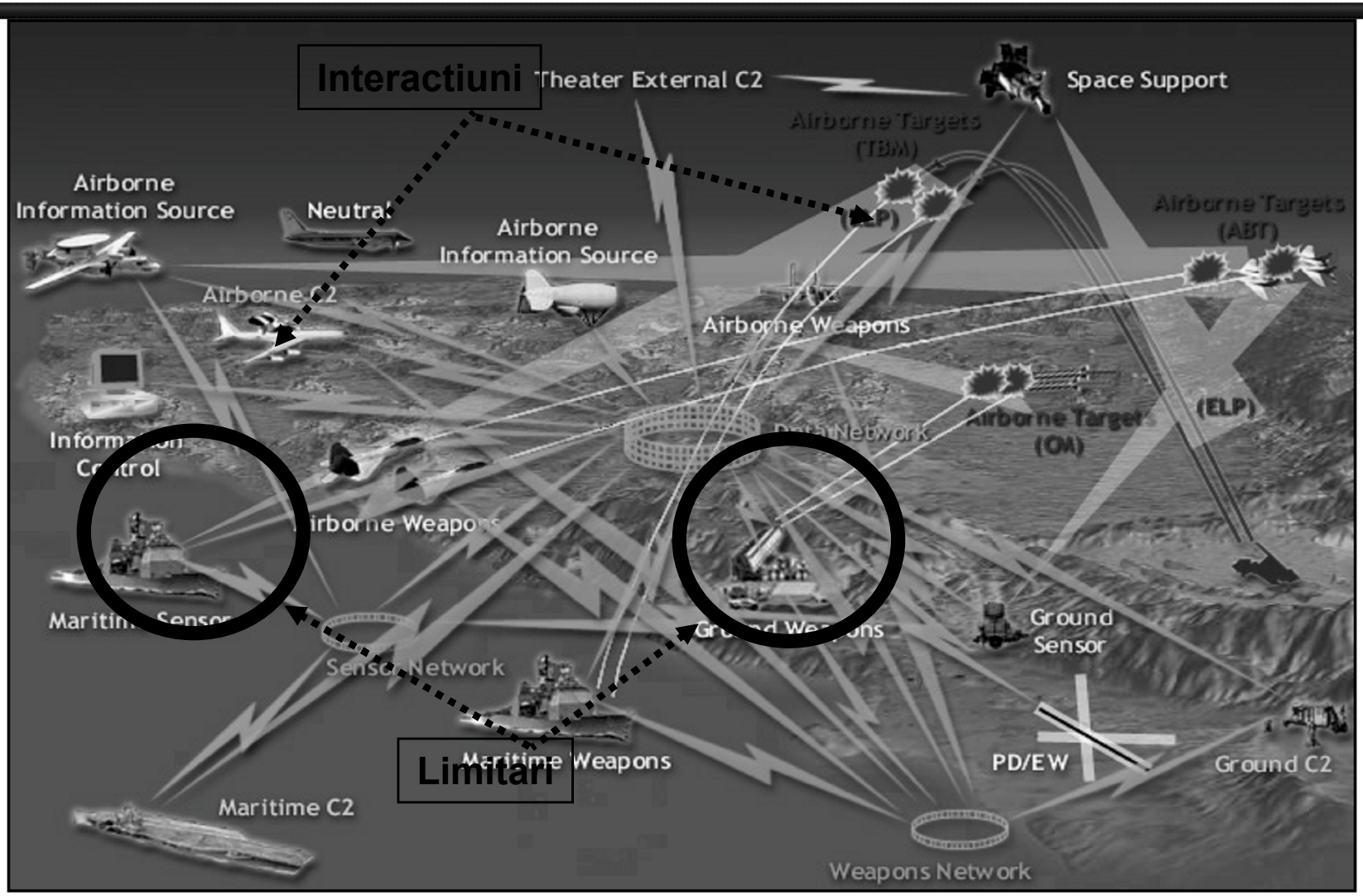
SysML (Systems Modeling Language) este un limbaj de modelare al aplicatiilor din ingineria de sistem.

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.

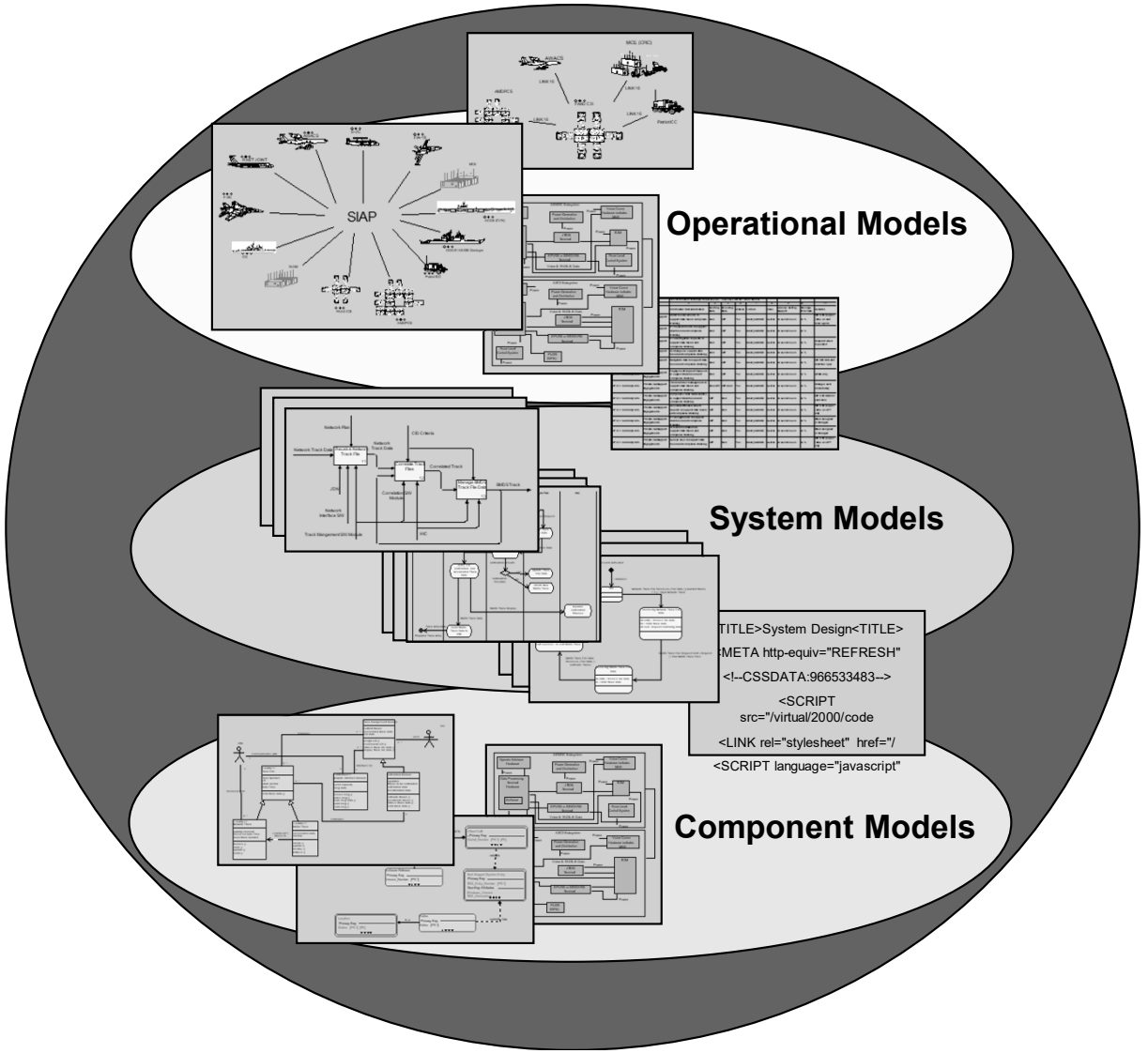
Ofera posibilitatea de a unifica concepte inrudite *software* si *non-software*, astfel umpland distanta ce se afla intre ele.

Modelarea sistemelor trebuie sa permita integrarea tuturor elementelor aferente acestora.

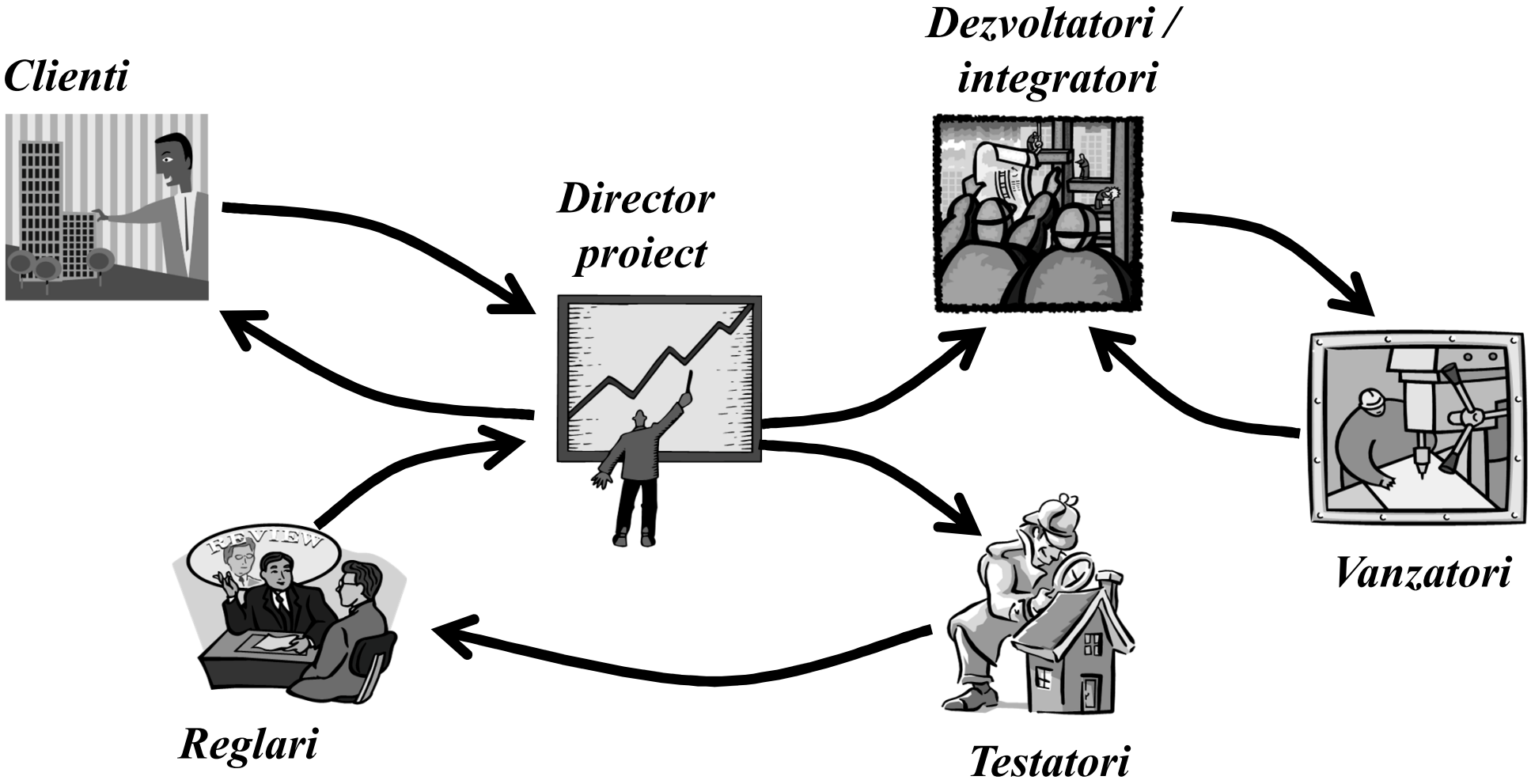
Reprezentarea unui "sistem de sisteme"



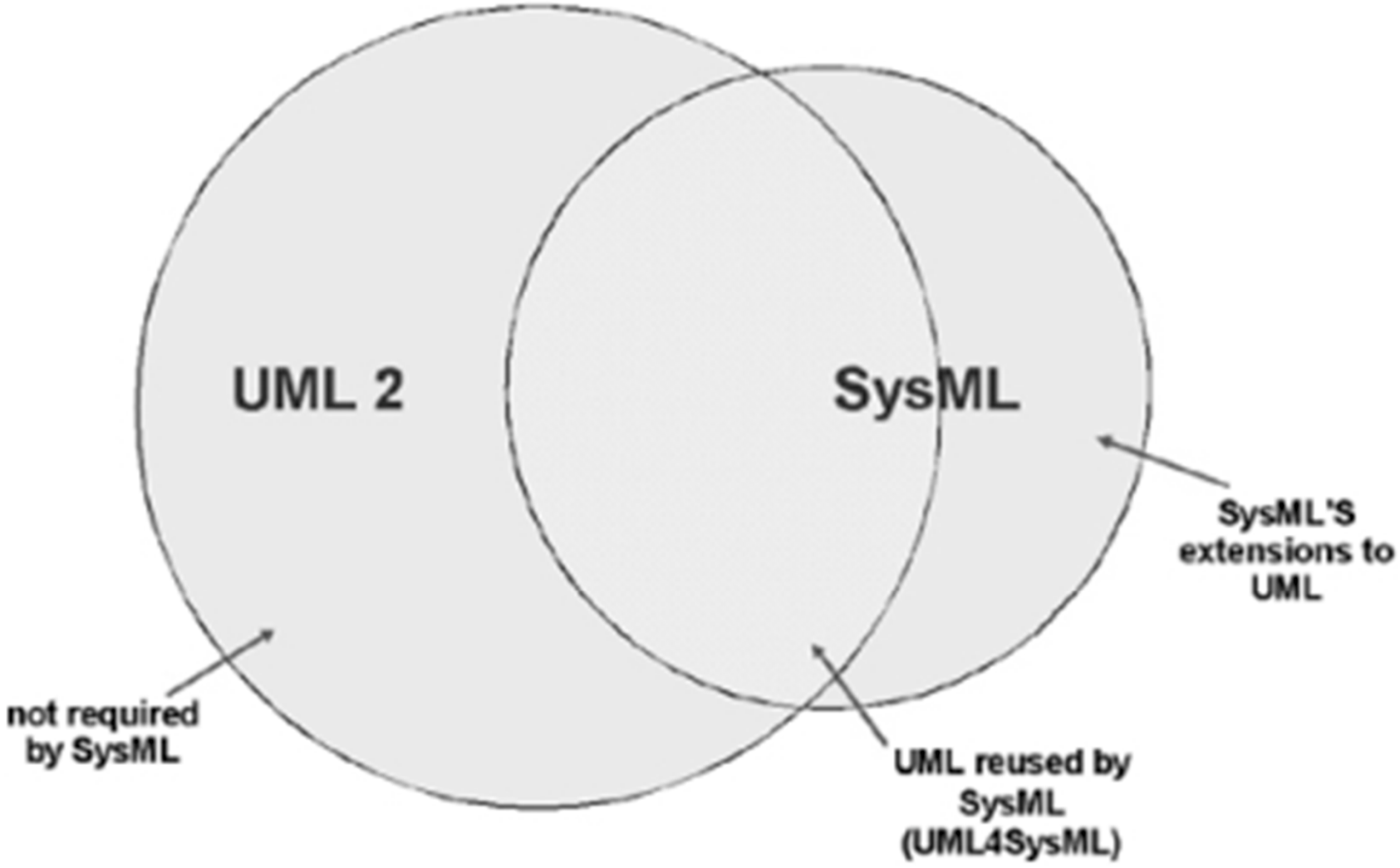
Modelarea nivelelor ierarhice



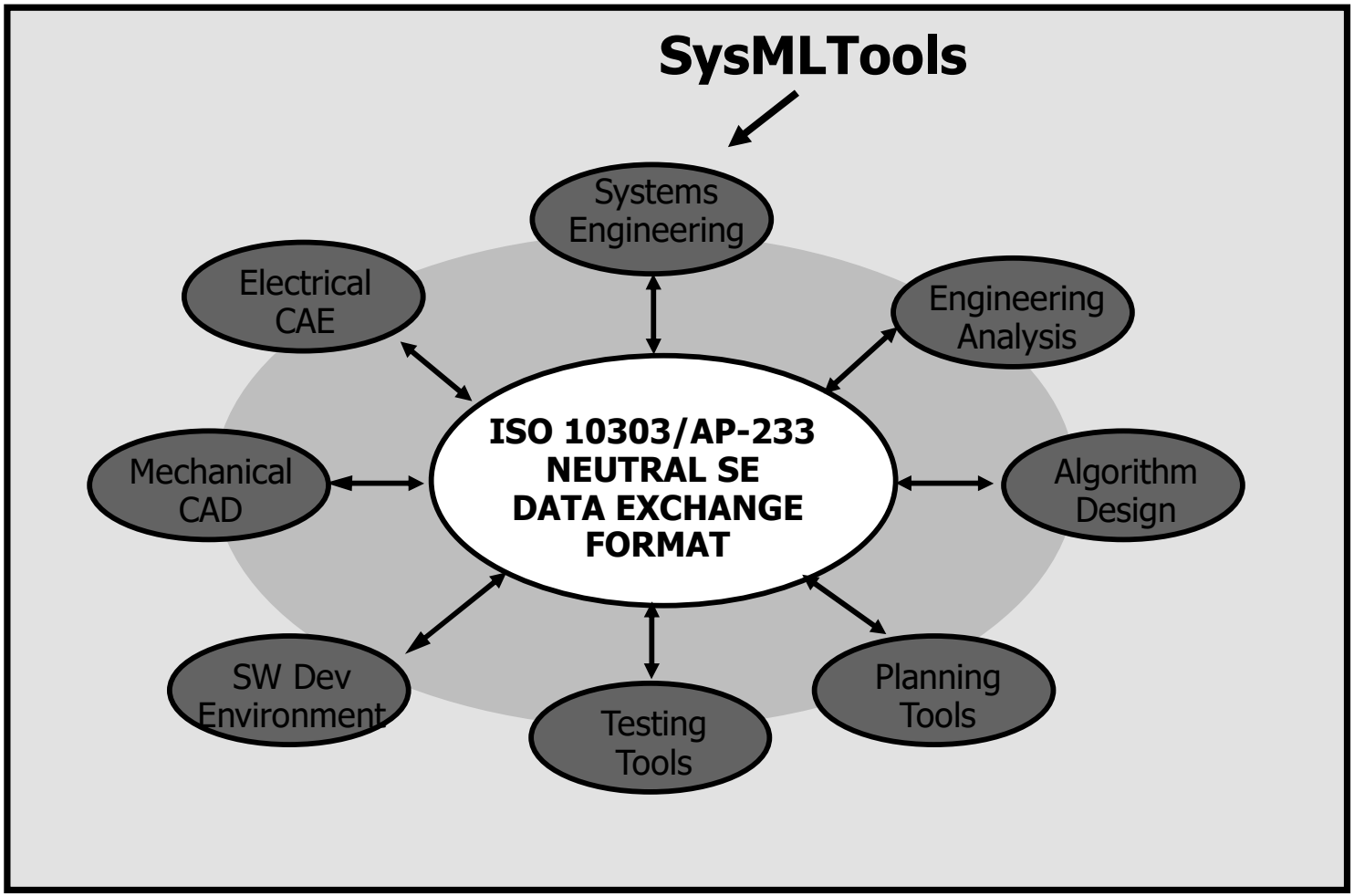
Modelarea comunicarii intre sisteme



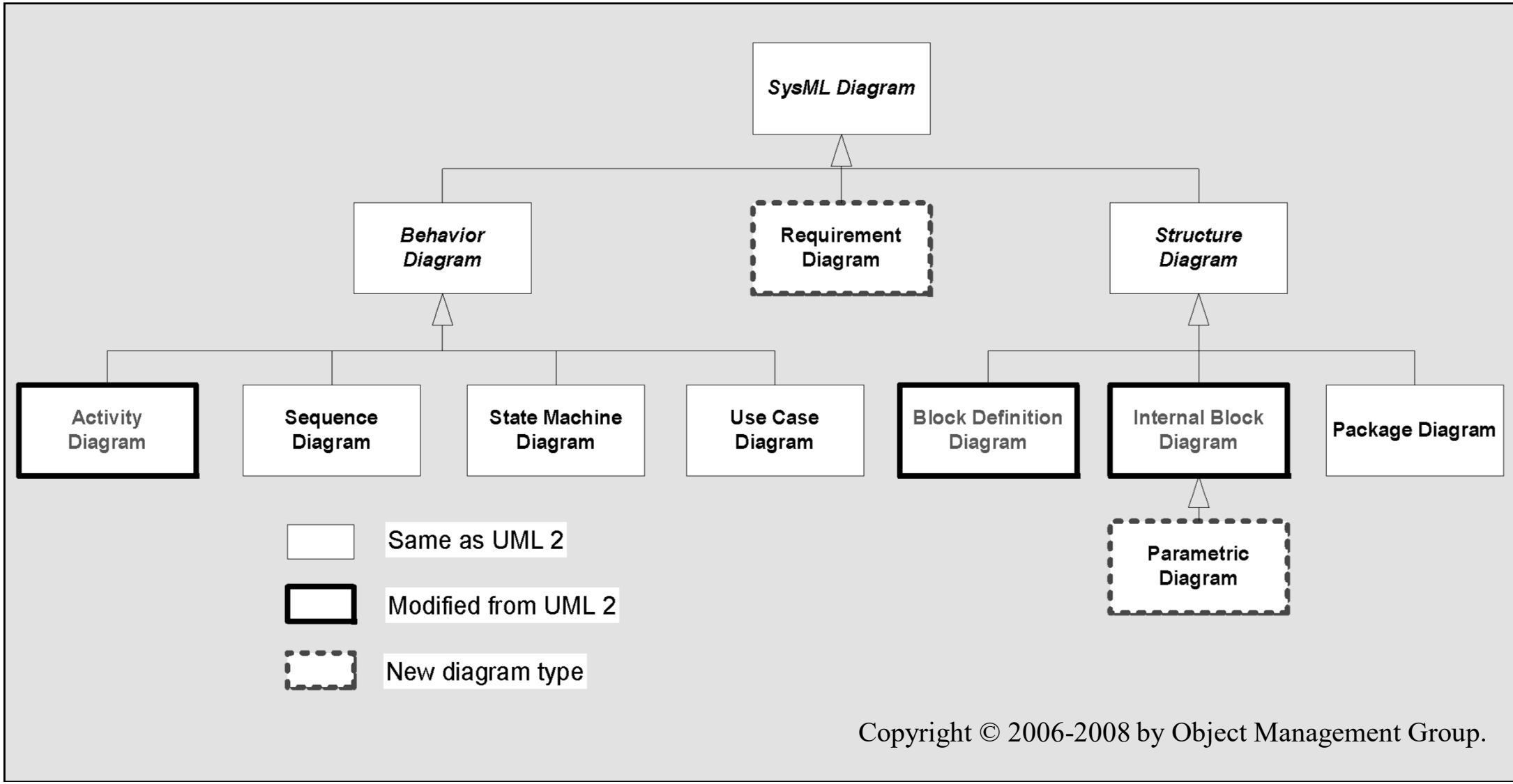
Legaturile intre SysML si UML



SysML - Pozitionare

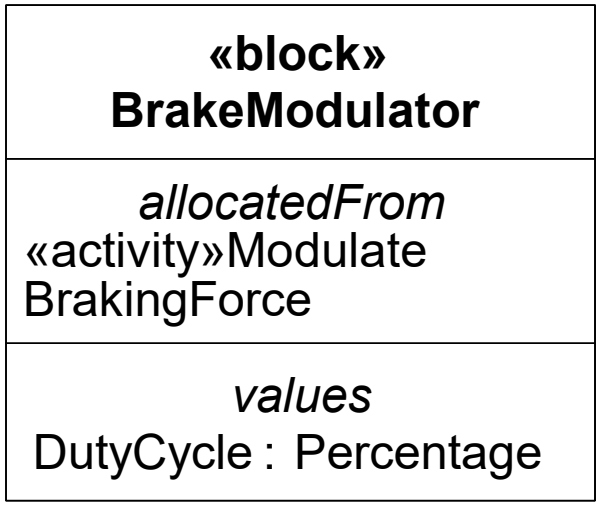


Taxonomia SysML



Blocuri (1)

- Blocurile sunt elementele structurale de baza (similare cu *clasele* din UML
- Prezinta un concept unificator de descriere a structurii unui element / sistem
  - Sistem
  - *Hardware*
  - *Software*
  - Date
  - Proceduri
  - Facilitati
  - Persoane



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

Tipuri de proprietati (1)

- Proprietatea reprezinta o caracteristica structurala a unui bloc:
  - **Proprietati de compozitie**
    - 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

Definire blocuri vs. utilizare blocuri

Diagrama definire blocuri

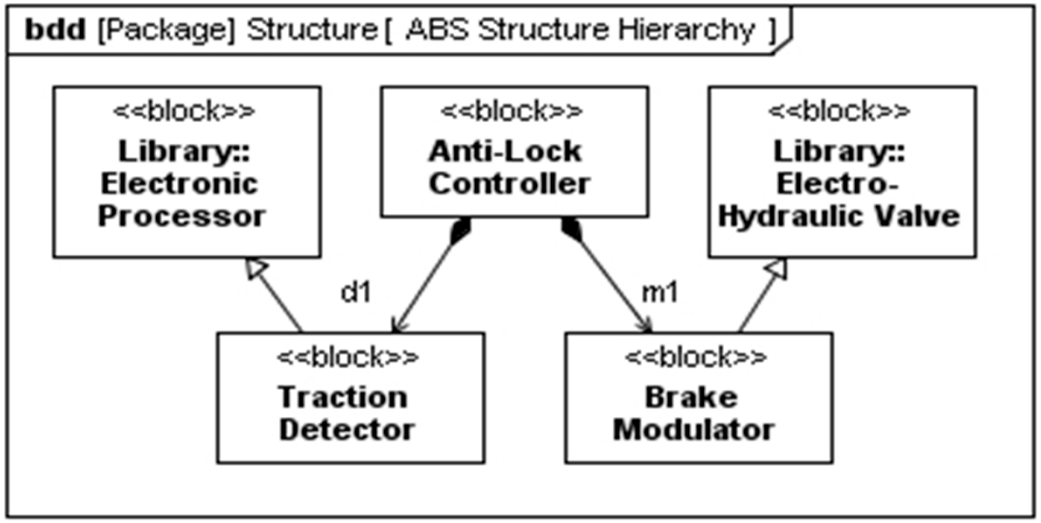
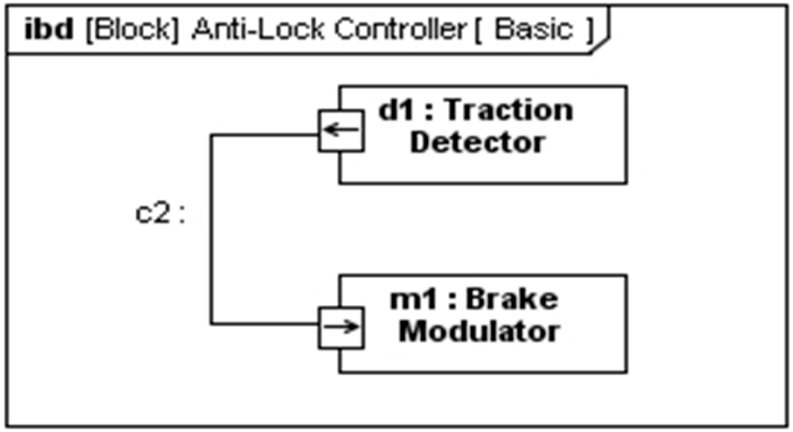


Diagrama interna (de blocuri)



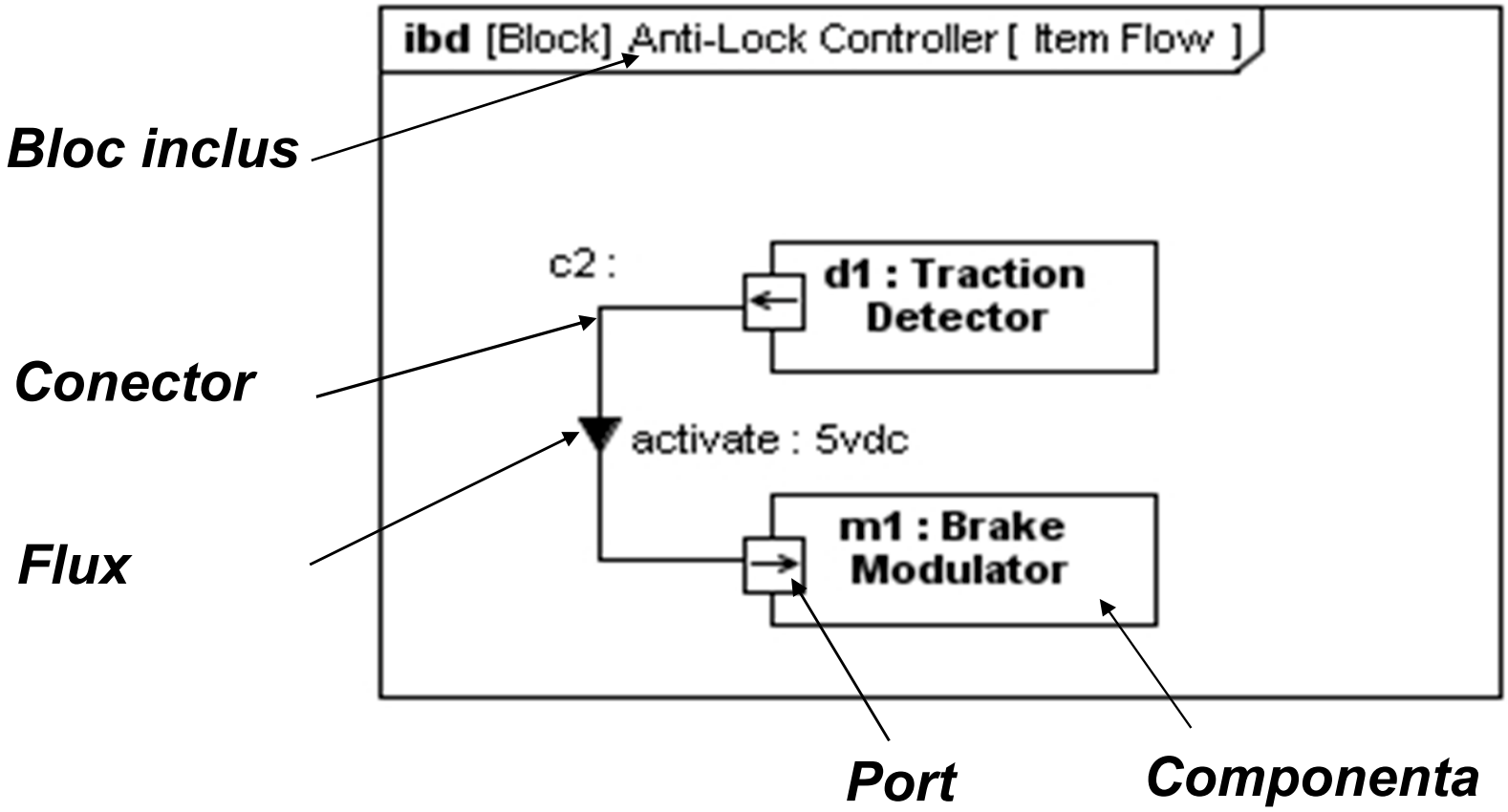
Definitie

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

Utilizare

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

Diagrama Interna de Blocuri (DIB)



DIB specifica interconexiunea partilor

Porturi SysML

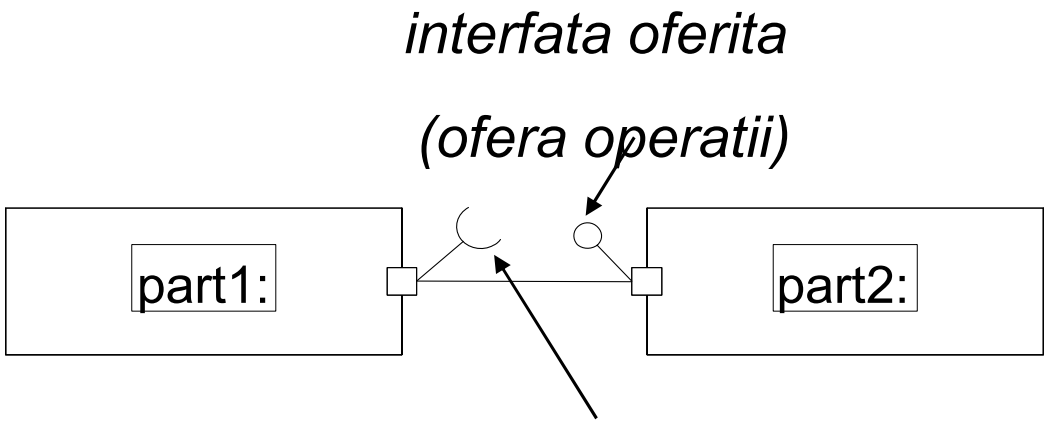
- Specifica punctele de interactiuni intre blocuri si partile componente
  - Integreza structura cu evolutia
  - portName:TypeName
- Tipuri de porturi
  - **Port Standard** (UML)
    - Specifica un set de operatii / semnale necesare sau oferite de bloc
    - Se specifica la nivelul interfetei UML
  - **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

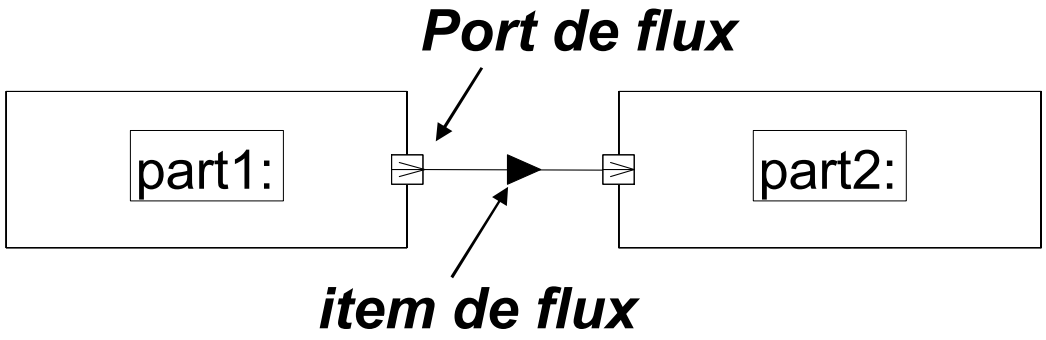
Port

Standard

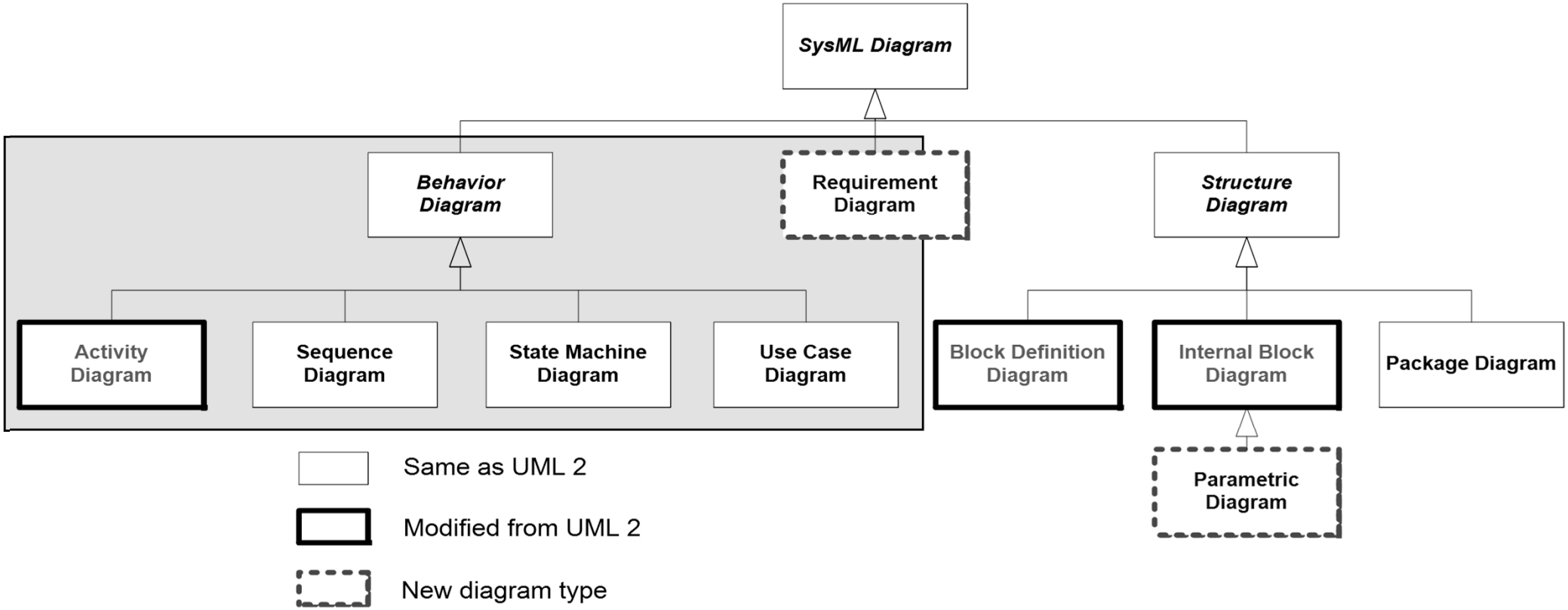


Port

de flux



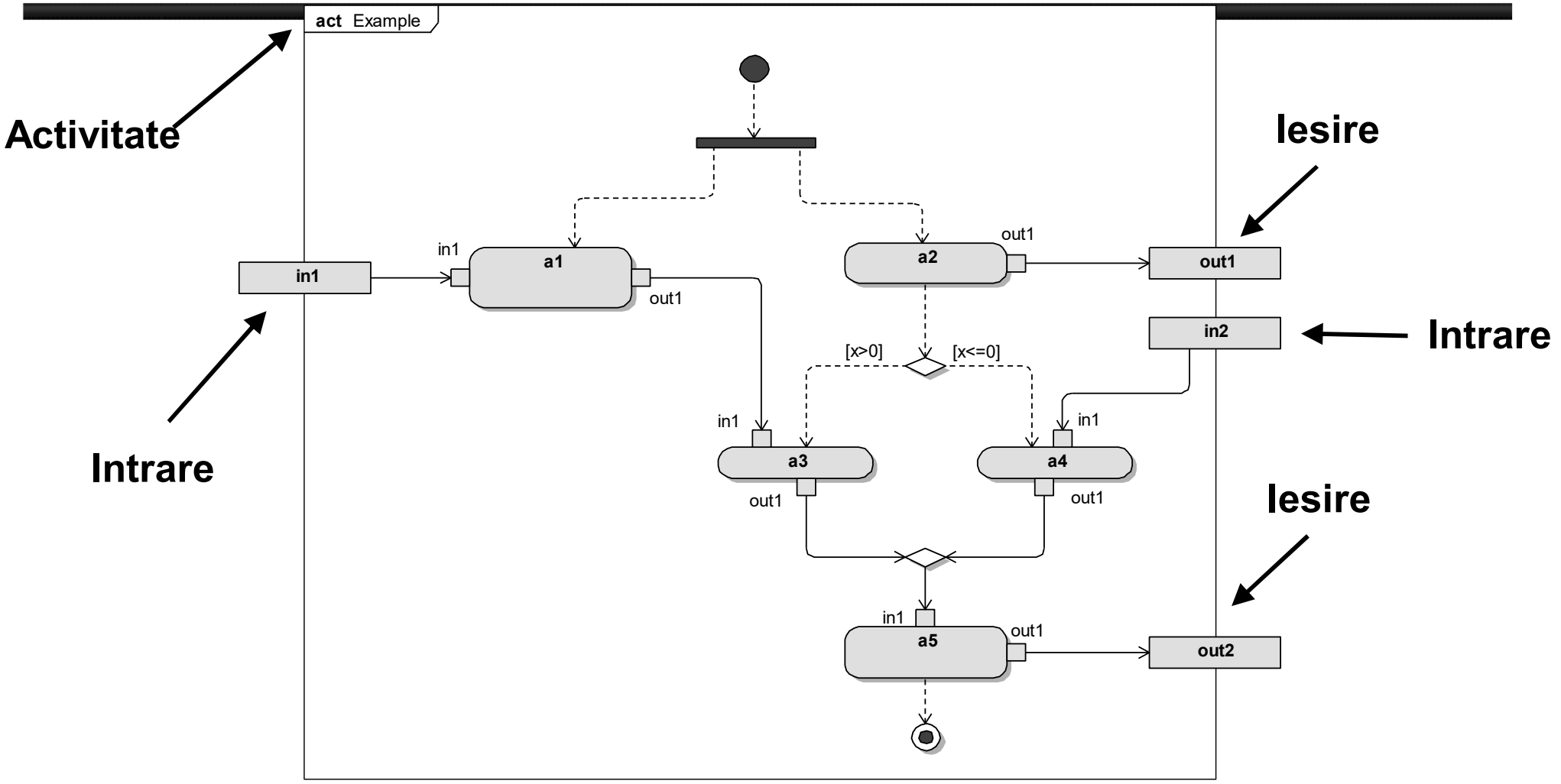
Diagrame de evolutie



Activitati

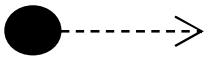
- **Activitatile** specifica transformarile intrarilor in iesiri prin intermediul unei secvente controlate de actiuni.
- Extensii SysML (ref. activitati):
  - Suport pentru modelarea (continua a) fluxurilor
  - Alinierea activitatilor la **Enhanced Functional Flow Block Diagram (EFFBD)**

Diagrama de activitati (DA)

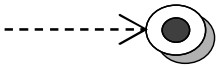


DA specifica o secventa controlata de activitati

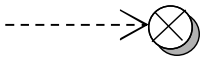
Reprezentari



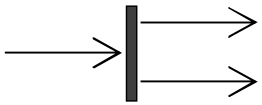
Nod initial



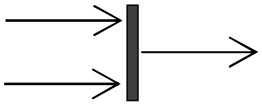
Nod final de activitate



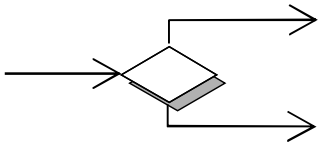
Nod final de flux



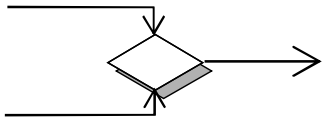
Nod de rascruce



Nod Join

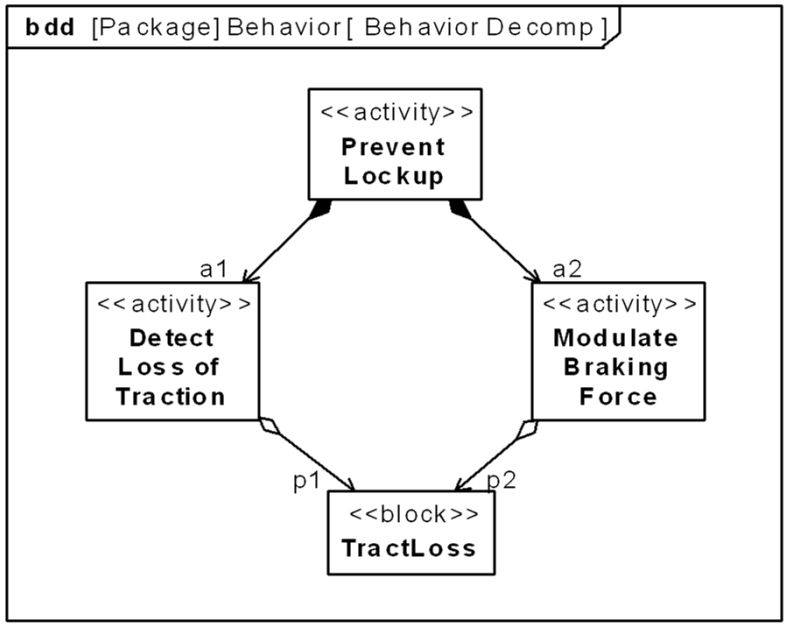


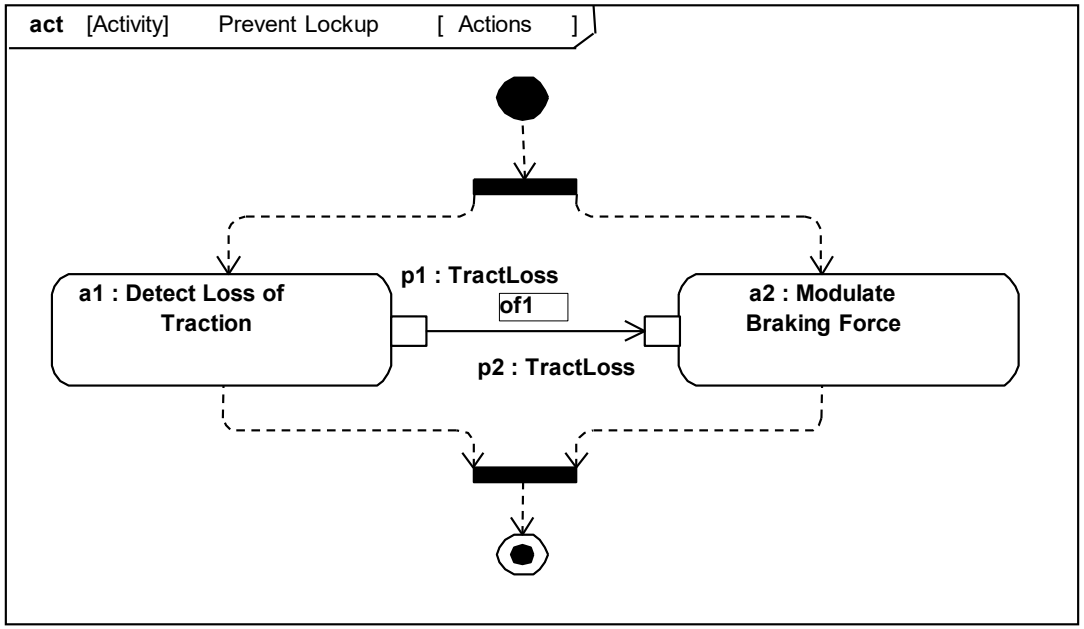
Nod de decizie



Nod de combinare

Descompunerea activitatilor





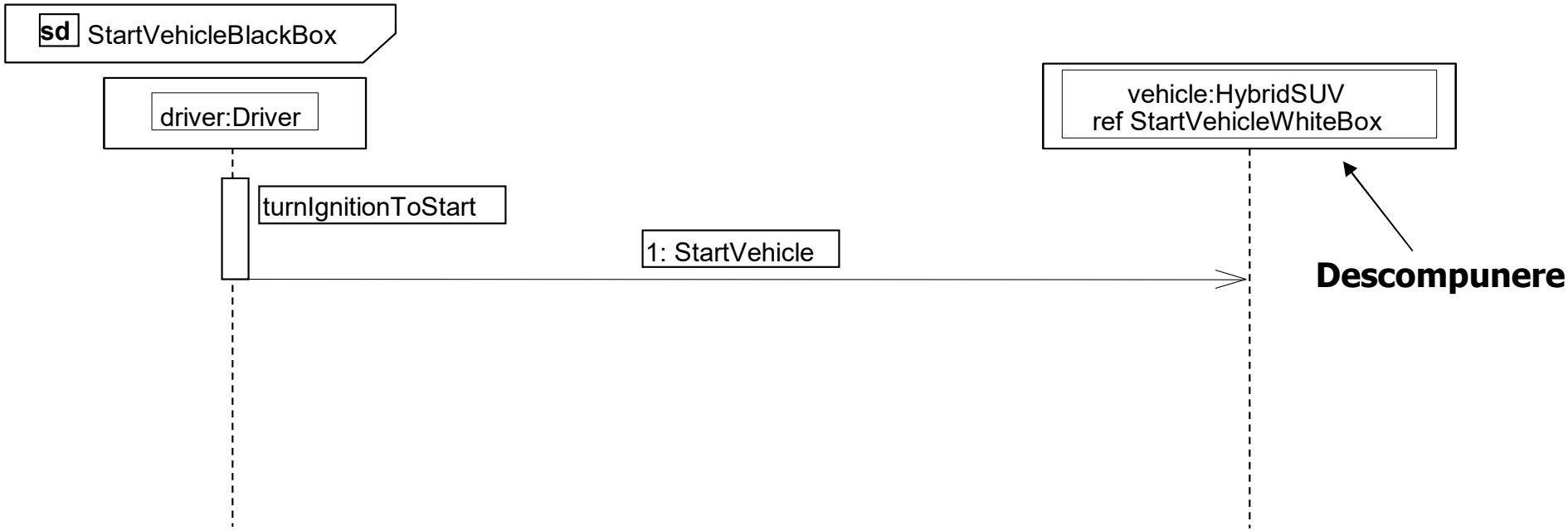
Definire

Utilizare

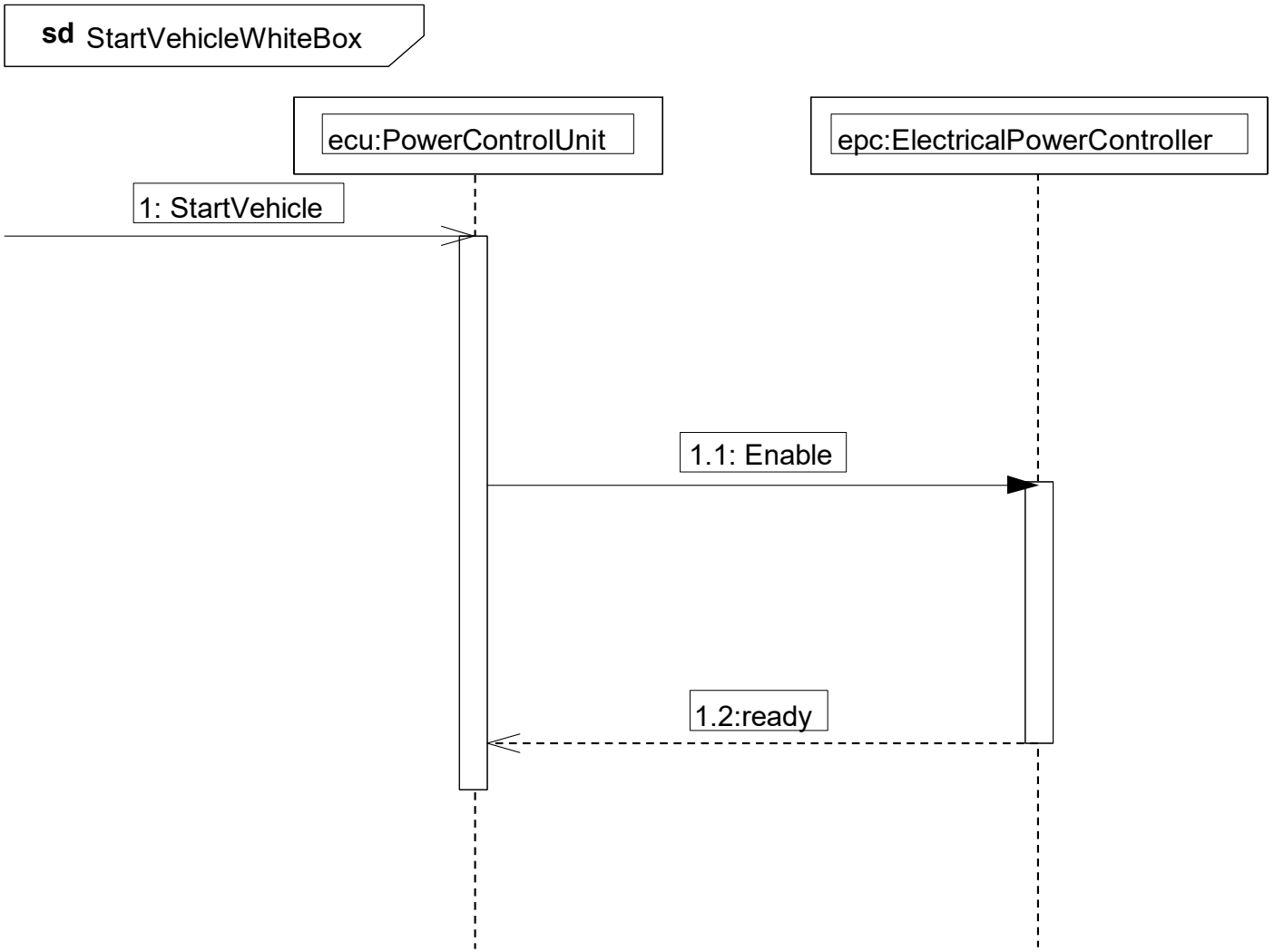
Interactiuni

- **Diagrama de secvente (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:
  - secvente
  - control logic
  - Etc.
- SysML nu include *timing*, interactiuni sau diagrame de comunicatii

Exemplu (pornire vehicul)



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 executatat 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 concurential:
    - Concurenta 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 nju 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 secventa
  - 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 interactiunea 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

Exemplu

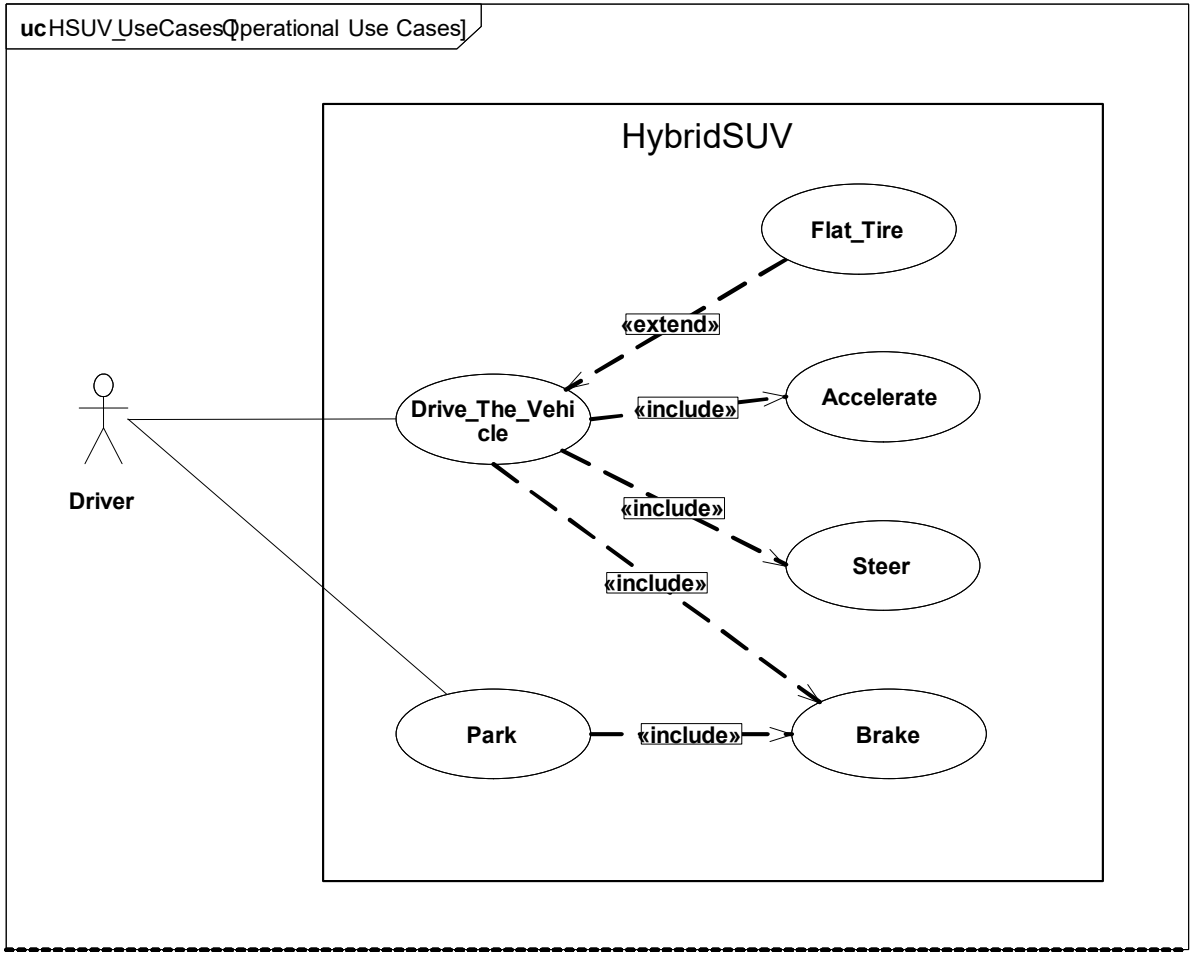
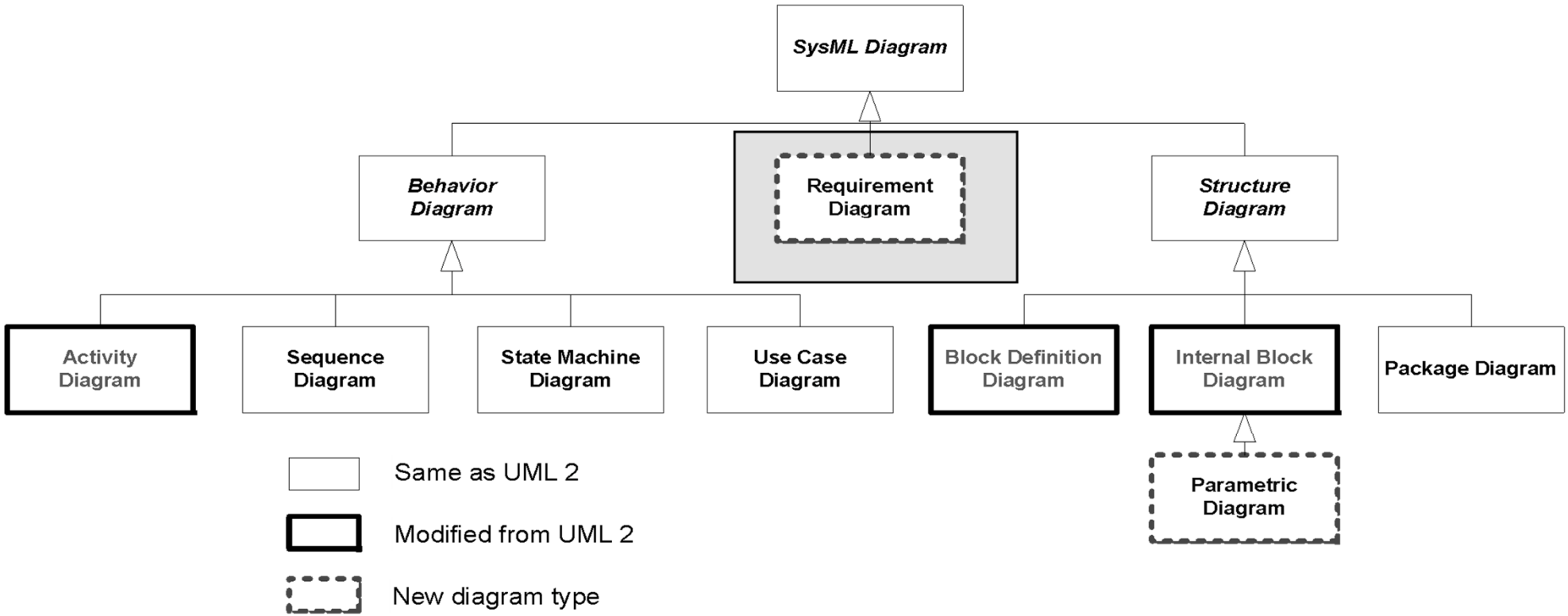

uc HSUV_UseCases [Operational Use Cases]
HybridSUV
Flat_Tire
«extend»
Drive_The_Vehi
cle
«include»
Accelerate
«include»
Steer
«include»
Driver
Park
«include»
Brake

Diagrama de necesitati

- Alocari
- Cerinte



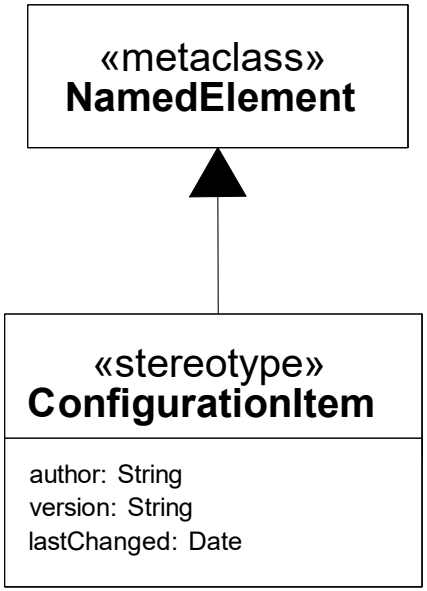
Alocari

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

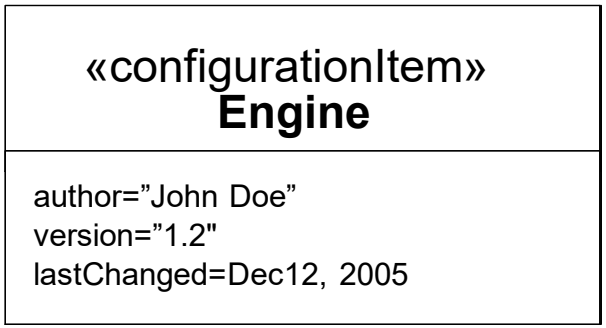
Cerinte

- 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

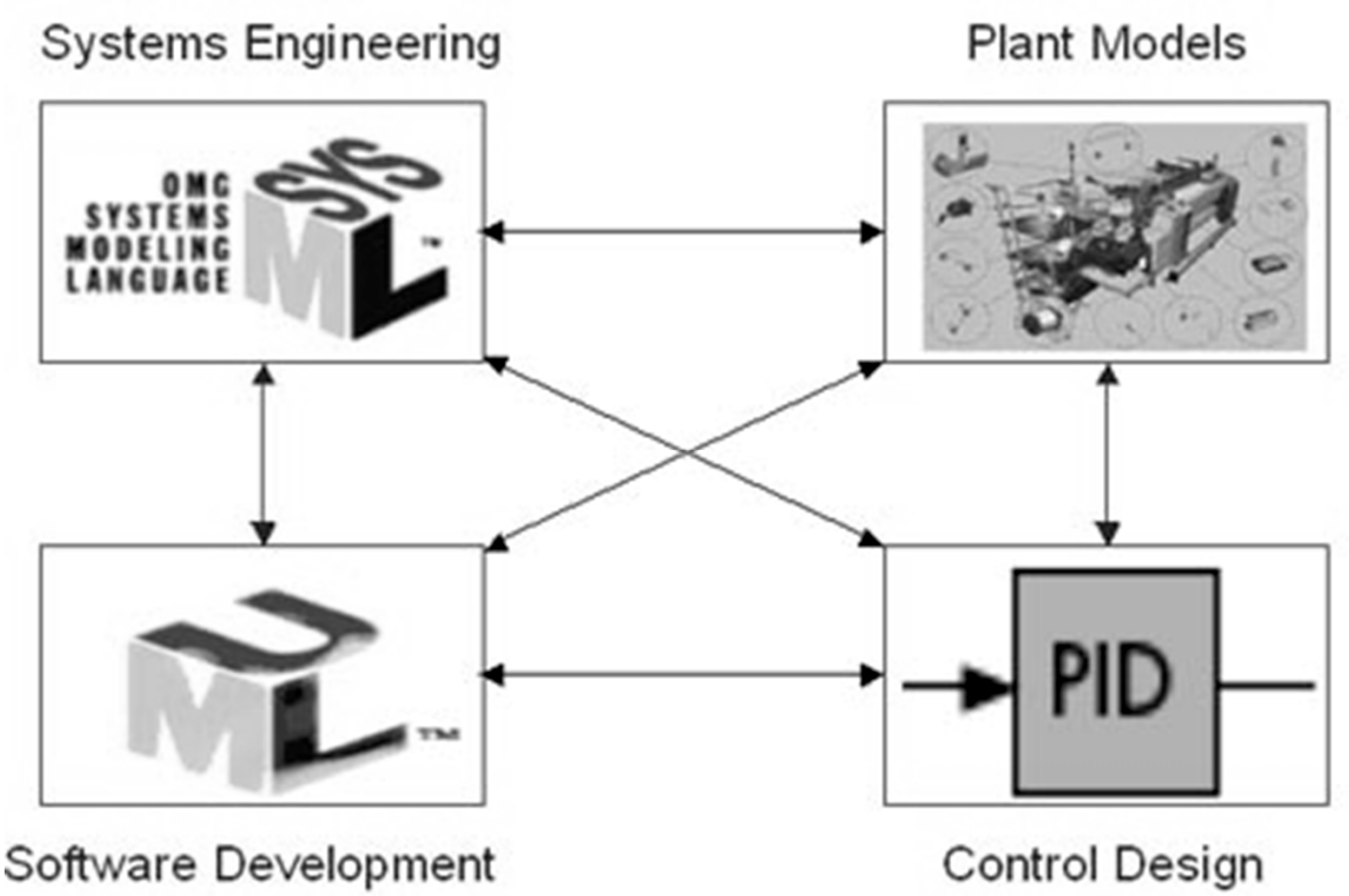


Definire Stereotip



Aplicare Stereotip

Exemplu de aplicatie integrata



UML vs SySML (1)

DIAGRAMA SYSML	SCOP	DIAGRAMA UML ANALOG
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 activitate (Activity diagram)**
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 clasa (Class diagram)**
Diagrama de blocuri interne (Internal Block diagram)	Arata structura interna a componentelor, incluzand partile si conectorii lor. Folositoare pentru analiza si designul sistemului.	**Diagrama de structuri compuse (Composite Structure diagram)**
Diagrama pachet (Package diagram)	Arata organizarea in pachete, vizualizari si puncte de vedere a unui model. Folositor pentru managementul modelului.	**Diagrama pachet (Package diagram)**

UML vs SySML (2)

DIAGRAMA SYSML	SCOP	DIAGRAMA UML
Diagrama parametrica (Parametric diagram)	Arata constrangerile parametrice dintre elementele structurale. Folositoare la analiza performantei si analiza cantitativa.	N/A
Diagrama cerinta (Requirement diagram)	Arata necesitatile sistemului si relatiile lor cu alte elemnte. Folositoare la ingineria necesitatilor.	N/A
Diagrama secventa (Sequence diagram)	Arata comportarea sistemului ca si interactiuni intre componentele sistemului. Folositoare la analiza sistemului si design	**Diagrama secventa (Sequence diagram)**
Diagrama de stare al masinii (State Machine diagram)	Arata comportamentul sistemului ca o secventa de stari pe care o componenta sau o actiune o cunoaste ca raspuns unor actiuni. Folositoare la designul sistemului si generarea simularii/codului.	**Diagrama de stare al masinii (State Machine diagram)**

UML vs SySML (3)

DIAGRAMA SYSML	SCOP	DIAGRAMA UML
Diagrama cazurilor de utilizare (Use Case diagram)	Arata cerintele functionarii sistemului ca tranzactii care sunt semnificative pentru folositorii sistemului. Folositoare la specificarea cerintelor functionalitatii.	**Diagrama cazurilor de utilizare (Use Case diagram)**
Tabele de alocare* **(Allocation tables)**	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.	N/A
*tabele derivate dinamic, nu sunt chiar diagrame		
N/A		**Diagrama de componente (Component diagram)**
N/A		**Diagrama de comunicare (Communication diagram)**

UML vs SySML (4)

DIAGRAMA SYSML	SCOP	DIAGRAMA UML
N/A		**Diagrama de desfasurare (Deployment diagram)**
N/A		**Diagrama de privire generala asupra interactiunilor (Interaction overview diagram)**
N/A		**Diagrama de obiecte (Object diagram)**
N/A		**Diagrama de timp (Timing diagram)**

Concluzii – relatia SysML / UML

- **UML** este un "General Purpose Modeling Language (GPML)", in timp ce **SysML** este un "Domain-Specific Modeling Language (DSML)" fiind definit ca o personalizare a lui UML 2.0

- **Avantaje**:
  - reutilizarea semanticii si notatiilor din UML 2.0
  - SysML personalizeaza mai bine semantica din ingineria sistemelor (prin adaugarea de doua noi diagrame: *Requirement Diagrams* si *Parametric Diagrams*)
  - SysML este "mai mic" dpdv al numarului de diagrame (9 vs. 13)

- **Dezavantaj**:
  - mosteneste multe din problemele din UML (d.e. Complexitatea notatiilor si semantica)