Diagrame pentru managementul modelelor (Packages)

Prof. univ. dr. ing. Florica Moldoveanu

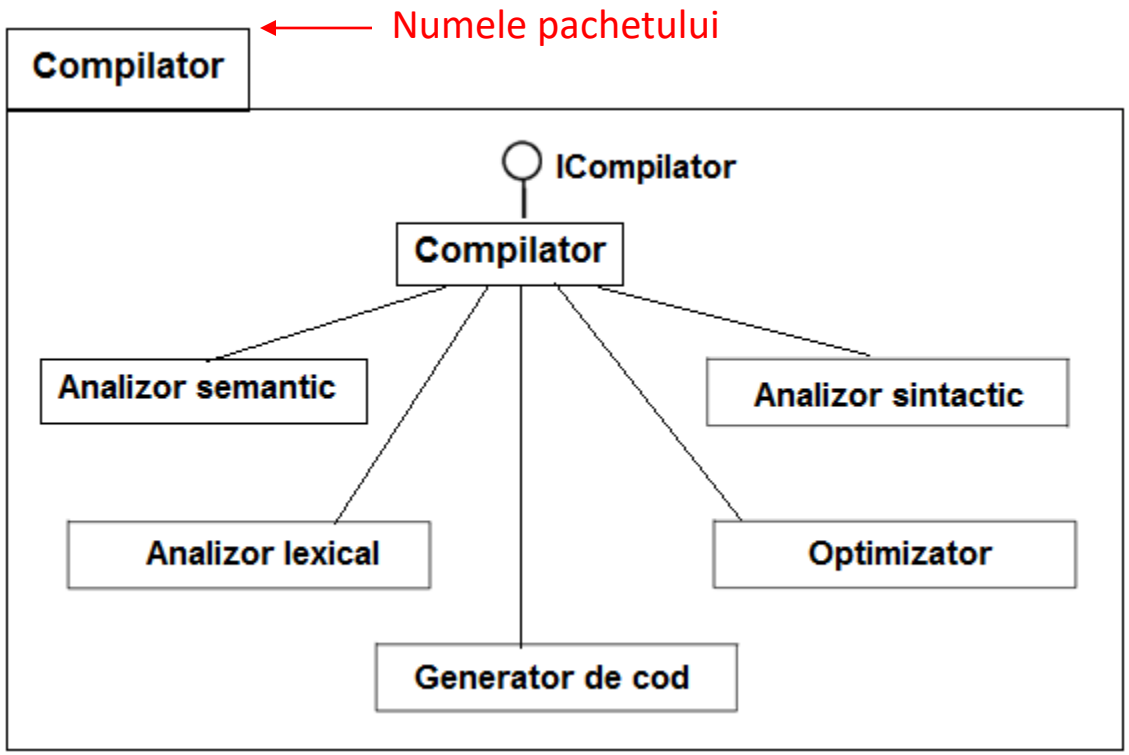
Curs *Ingineria programelor* – UPB, Automatică şi Calculatoare
2020-2021

Pachete - utilizari

- Se folosesc pentru organizarea artefactelor rezultate în procesul de dezvoltare al unui sistem software.
- Un pachet are exact aceeasi functionalitate ca si un director pe disc (folder).
- Un pachet contine elemente de modelare corelate logic:
  - cazuri de utilizare, diagrame de interactiune asociate cazurilor de utilizare, diagrame de clase si diagrame de componente care implementeaza cazurile de utilizare, diagrame de activitate
  - clasele care formeaza un subsistem, etc.
- Un pachet poate contine alte pachete, deci se poate crea o ierarhie de pachete.
  - Instrumentele de modelare UML afiseaza grafic gruparea elementelor de modelare în pachete printr-un arbore similar cu cel afisat de Windows Explorer.
- Pachetele sunt containere. Daca se şterge sau muta un pachet, toate elementele din interiorul său sunt şterse/mutate.

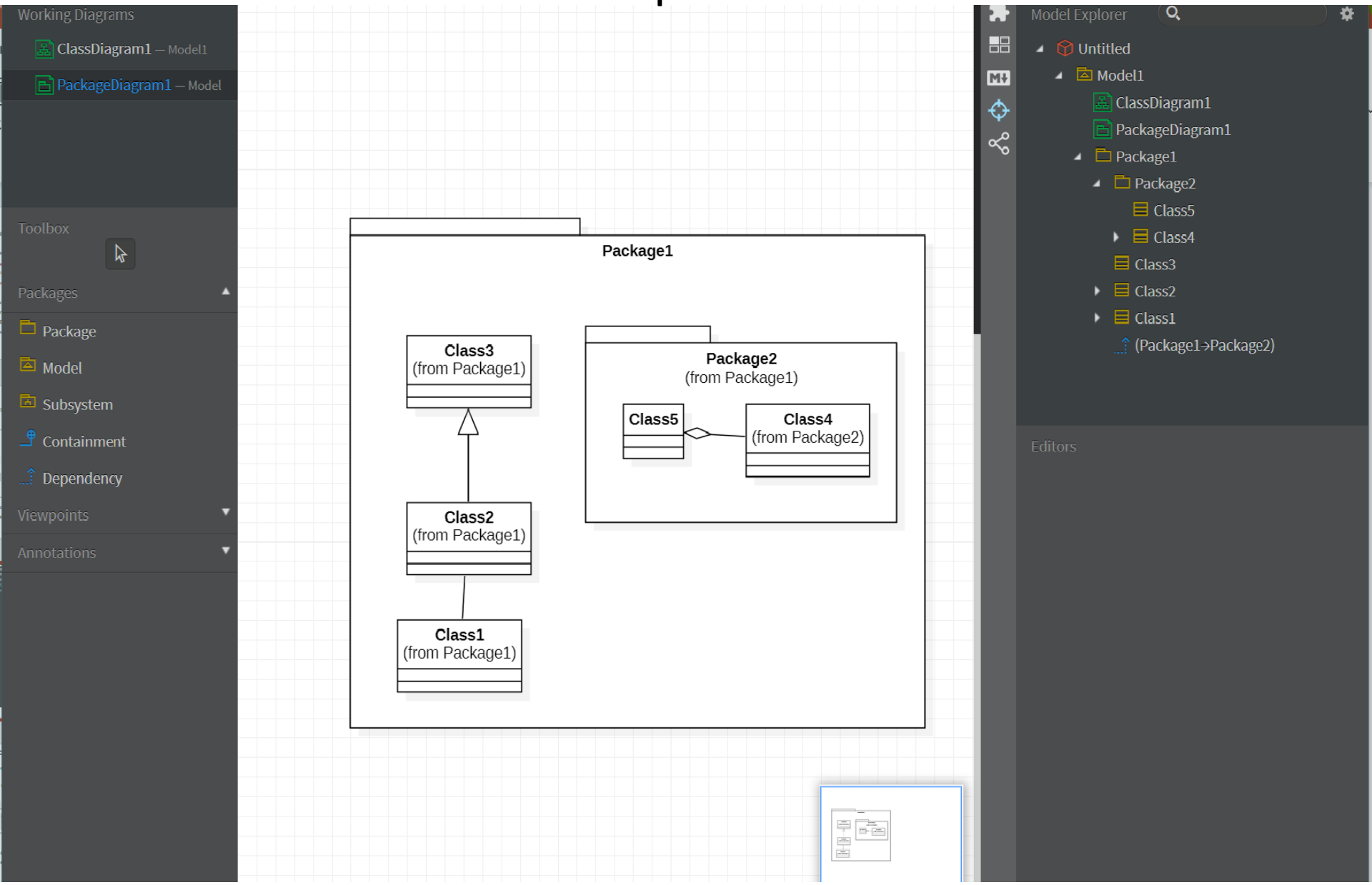
Exemplu de pachet UML

Gruparea elementelor de modelare ale subsistemului Compilator, într-un pachet UML.



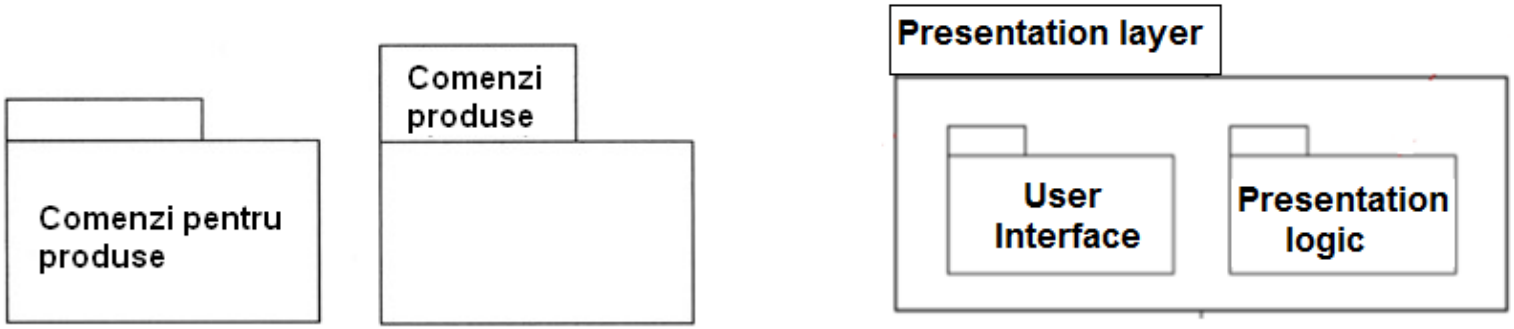
- Pachetul cuprinde: clasele subsistemului, clasa “fațadă” Compilator şi interfața **Icompilator**.
- Pot fi adaugate: cazurile de utilizare implementate de subsistem, diagrame de activitate, s.a

Ierarhie de pachete



Pachete – reprezentare grafica

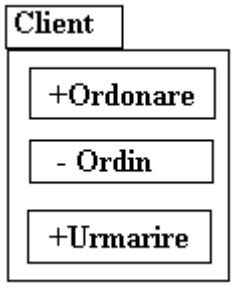
- Un pachet este reprezentat printr-un dreptunghi cu o mica “toartă” ataşată dreptunghiului in partea de stanga sus.
- Numele pachetului poate fi amplasat în centrul dreptunghiului sau pe toartă:



- Un pachet are asociat un „spatiu de nume” (namespace): elementele amplasate în acelasi pachet trebuie sa aiba nume unice; elemente din pachete diferite pot avea acelasi nume.

Vizibilitatea elementelor dintr-un pachet

- Elementele unui pachet pot fi marcate cu atributele de vizibilitate **public (+)** sau **private (-):**



➢ Elementele publice ale unui pachet sunt întotdeauna accesibile din afara pachetului folosind ***nume calificate***, de forma:

Nume_pachet::nume_element

Nume_model::nume_pachet::nume_clasa::nume_operatie

- Un pachet poate **importa** fie membrii individuali ai altor pachete fie alte pachete în întregime.
- **Spatiul de nume al pachetului care importa este extins cu cel al pachetului important.**

➢ Elementele pachetului Prezentare pot fi referite in pachetul Aplicatie Web folosind nume necalificate.

Relatii intre pachete (1)
Extindere şi import

- Un pachet poate fi extins cu un alt pachet.
- Relaţiile de import şi extindere (merge) sunt relaţii de dependenţă între pachete.

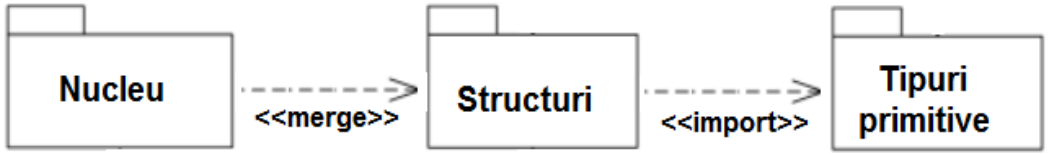
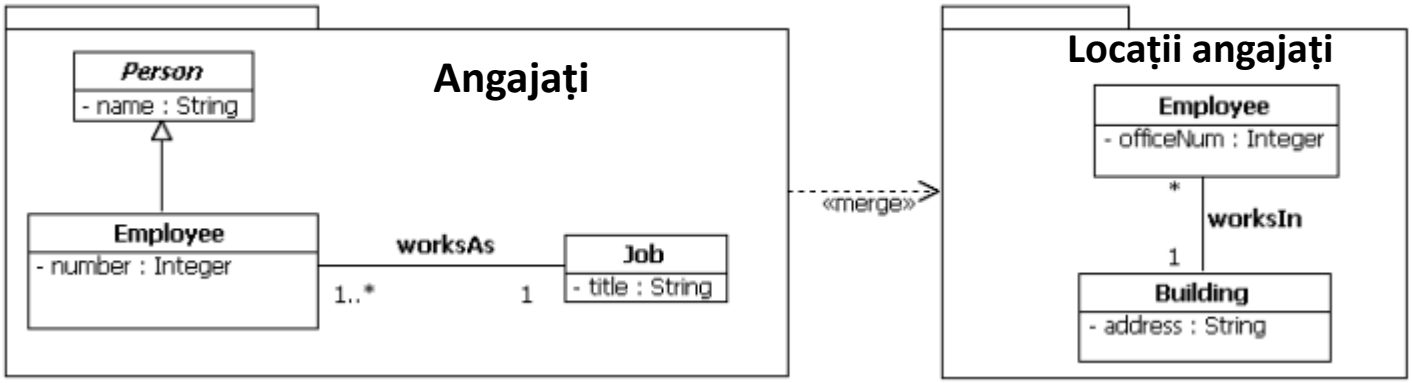


Diagrama de pachete

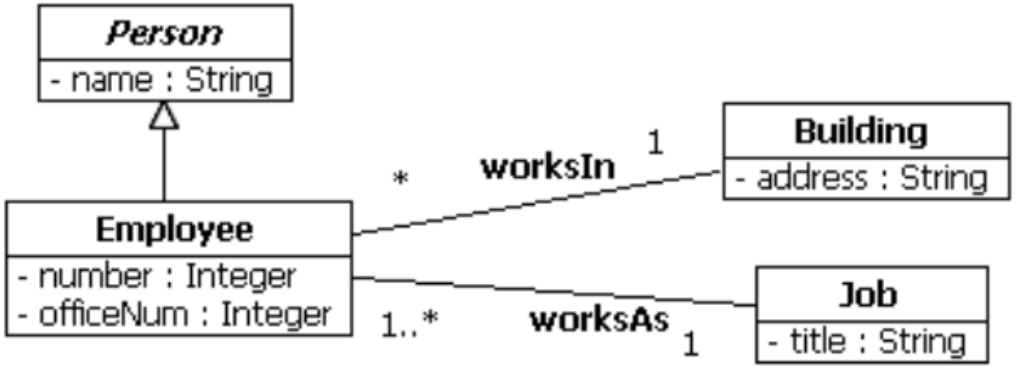
- Pachetul „Nucleu” este extins (combinat) cu pachetul „Structuri”.
- Pachetul “Tipuri primitive” este importat de pachetul “Structuri”, ceea ce permite referirea la elementele pachetului “Tipuri primitive” din pachetul “Structuri” utilizând nume necalificate: spaţiul de nume al pachetului “Tipuri primitive” este adăugat la spaţiul de nume al pachetului “Structuri”.

Relatii intre pachete (2)

Extindere



- Elementele din **Locații angajați** care au acelasi nume cu elemente din **Angajați** sunt combinate (cu attribute si relatii) iar cele care nu se regasesc in pachetul **Angajați** sunt copiate în pachetul rezultat.



Continutul pachetului **Angajați** extins cu **Locatii angajati**

Relatii intre pachete (3)

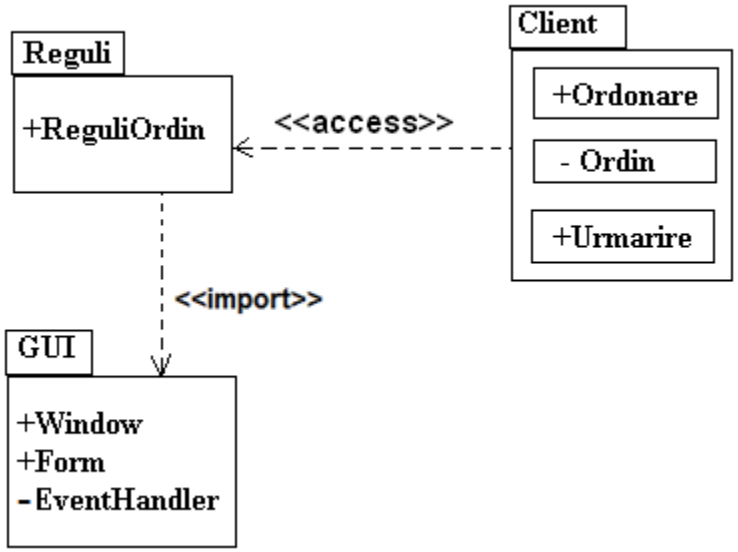
Import public şi privat

- Importul elementelor unui pachet poate fi public sau privat:

  - public - elementele din pachetul importat isi pastreaza atributul de vizibilitate în cadrul spaţiului de nume al pachetului care importă;

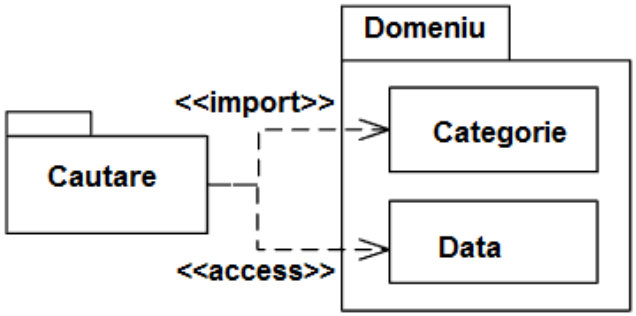
  - privat - elementele pachetului importat sunt accesibile numai în pachetul care-l importă.

- Vizibilitatea importului este definită prin stereotipul ataşat relaţiei de import: **<<import>>** pentru public, respectiv **<<access>>** pentru privat.



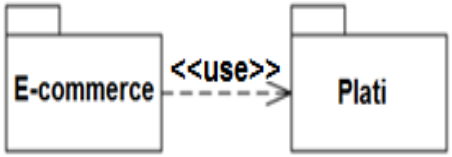
Relatii intre pachete (4)
Import membri individuali

- **Importul unor membri individuali** ai unui pachet se reprezintă printr-o săgeată care punctează către membrii importaţi.
- Membrii importaţi sunt adăugaţi la spaţiul de nume al pachetului care importă.



Vizibilitatea importului unui membru al unui pachet poate fi de tip **public** sau **privat**, cu aceeasi semnificaţie ca în cazul importului întregului pachet.

Relaţia de dependenţă intre pachete



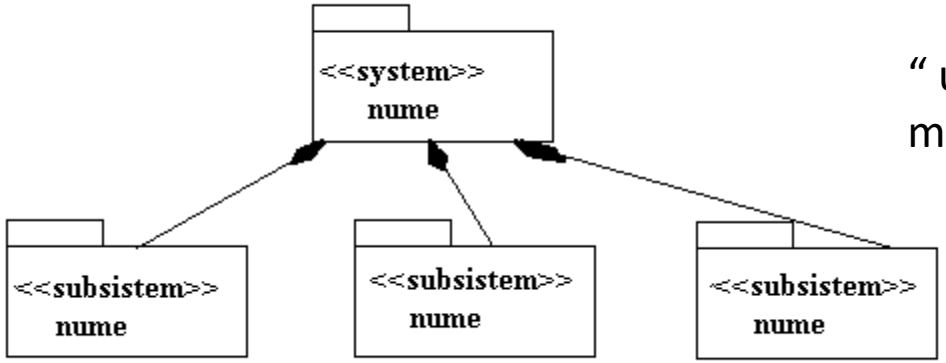
Relaţia <<use>> nu precizează cum elementul client (E-commerce) utilizează furnizorul (Plati). Astfel, pachetul „Plăţi” ar putea fi utilizat în definiţia sau în implementarea pachetului „E-commerce”.

Sisteme şi Modele(1)

- Un sistem este o grupare de elemente organizată pentru realizarea unui scop.
- Sistemele pot fi descompuse în subsisteme separate, fiecare subsistem putand fi vazut la randul sau ca un sistem, la un nivel de detaliu mai coborat.
- Subsistemele sunt grupări de elemente care pot fi dezvoltate relativ independent. Descompunerea are loc în etapa de proiectare arhitecturală, când se decide cum vor fi realizate cerințele.
- Relațiile sistem-subsisteme pot fi reprezentate:
  - In UML 1.x, printr-o diagramă de pachete
  - In UML 2.x, printr-o diagramă de componente

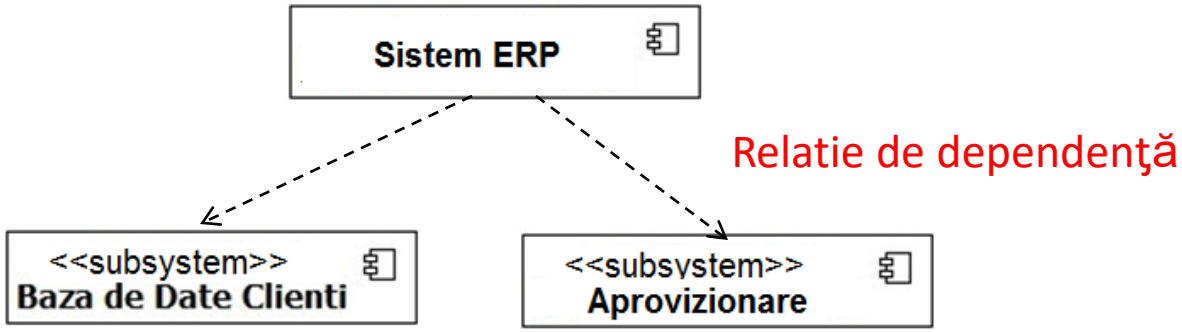
Sisteme şi Modele (2)

- In UML 1.x, un subsistem este un tip particular de pachet. Se reprezinta ca un pachet cu stereotipul <<subsistem>>.



“ un sistem este compus din mai multe subsisteme”

- In UML 2.x, un subsistem este un tip particular de componentă

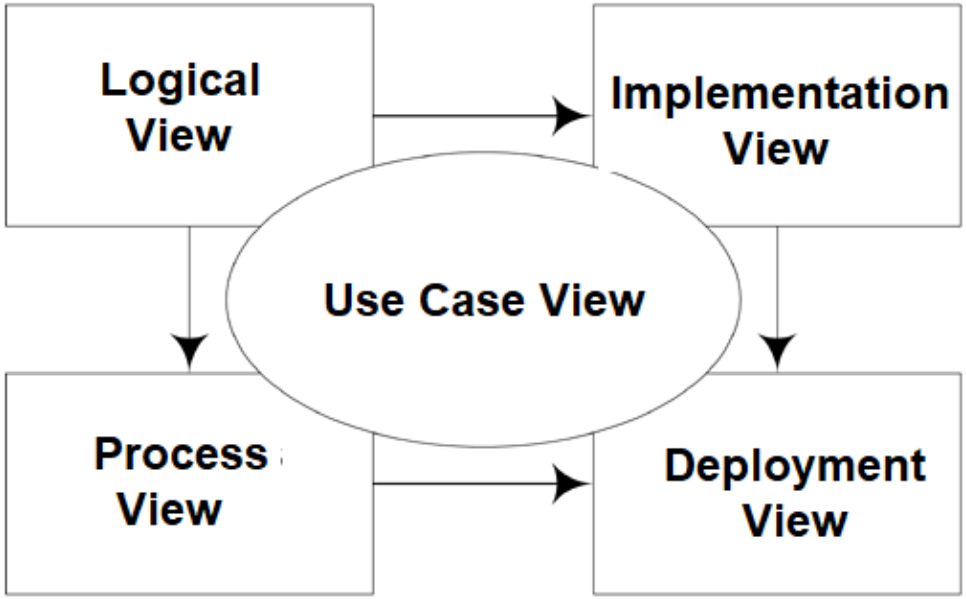


Relatie de dependenţă

Sisteme şi Modele (3)

- Modelul unui sistem este o simplificare a realității, o abstracție a sistemului, creata pentru a înțelege mai bine sistemul şi a-l specifica.
- O **vedere** este o proiecție a unui sistem dintr-un punct de vedere în care sunt omise aspectele nerelevante din punctul de vedere respectiv.

- Exemplu: cele 5 vederi ale unei arhitecturi software



Sisteme şi Modele (4)

Use Case View cuprinde:

- Cazurile de utilizare care descriu comportarea sistemului vazută de utilizatori, analisti şi testeri
- Diagrame de cazuri de utilizare
- Diagrame de interacțiune, de stări şi de activități

Logical View: vedere externă asupra sistemului, independentă de implementare

Cuprinde:

- Clasele si diagramele de clase din modelul de analiza - pentru modelarea aspectelor statice
- Diagrame de interactiune, de stari si de activitate - pentru modelarea aspectelor dinamice

Sisteme şi Modele (5)

Process View: procesele si firele de executie care formeaza mecanismele de concurență si de sincronizare ale sistemului.

- Diagrame de clase ale obiectelor care participa la fire si procese de executie.
- Diagrame de interactiune, de activitate si de stări

Implementation View

Descrie componentele care alcatuiesc sistemul fizic final. Se folosesc:

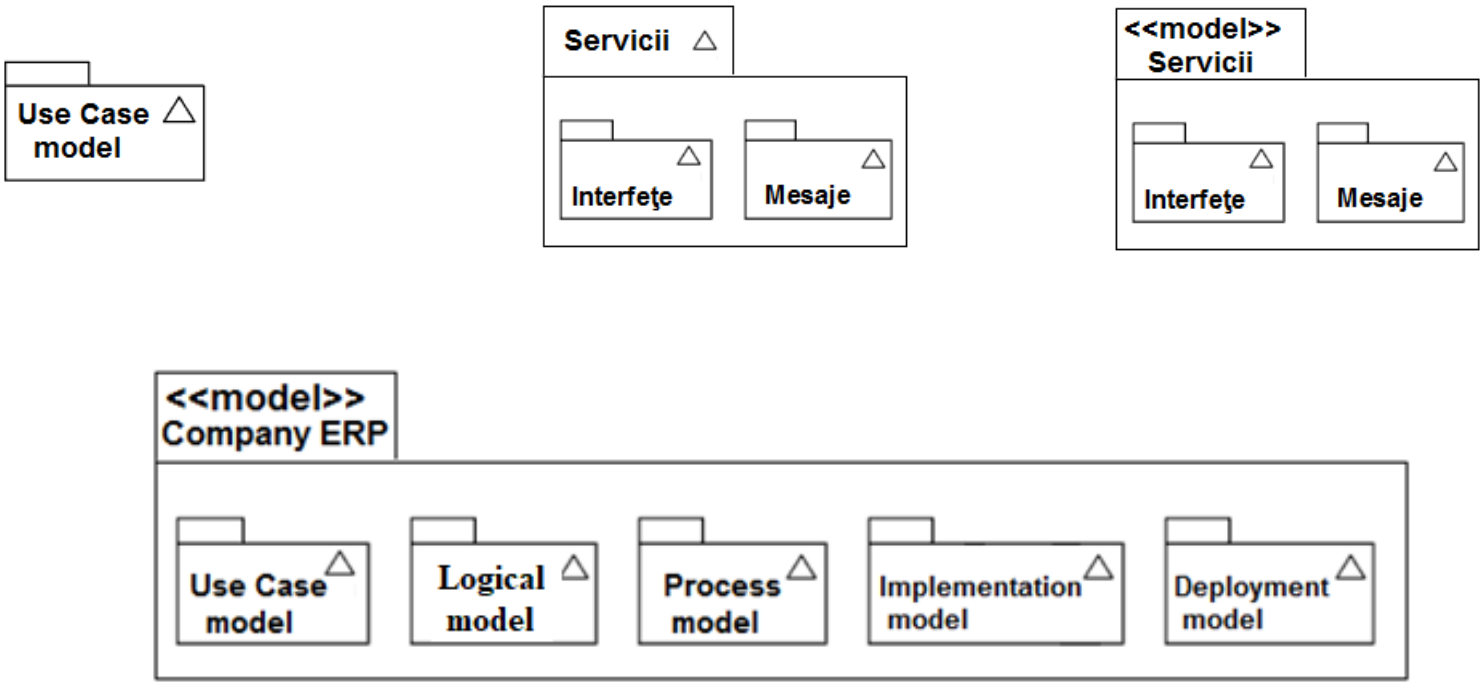
- Diagrame de clase si de componente pentru a modela aspectele statice
- Diagrame de interactiune, de stari si activitate, pentru modelarea aspectelor dinamice.

Deployment View: nodurile care formeaza topologia hardware a sistemului si repartitia elementelor software fizice pe noduri. Se folosesc:

- Diagrame de distributie, pentru modelarea aspectelor statice
- Diagrame de interactiune, de stari si de activitate, pentru modelarea aspectelor dinamice.

Sisteme şi Modele (6)

- **Elementele de modelare care constituie o vedere asupra unui sistem pot fi grupate intr-un *model*.**
- Un model se reprezinta printr-un pachet marcat cu stereotipul <<model>> sau unul care include simbolul triunghi:



Lecturi suplimentare - pachete

https://www.uml-diagrams.org/package-diagrams.html

https://www.uml-diagrams.org/package-diagrams-overview.html

https://www.uml-diagrams.org/package-diagrams-reference.html

https://www.win.tue.nl/~aserebre/2IW80/2013-2014/05%20-%20UML%20Structural%20Diagrams%20Other%20Diagrams.pdf