

# Modellalkotás UML-ben

# Modellalkotás UML-ben

- A Unified Modeling Language (UML) egy grafikus modellező nyelv, amely
  - lehetőséget nyújt egy megoldandó probléma **specifikációjának** leírására absztrakt szinten,
  - Módszereket, eszközöket nyújt a specifikáció **megvalósítására** és
  - a megoldás **dokumentálására**.
- Az UML-re vonatkozó alapismereteket előző tanulmányaiban megszereztek.
  - Ismereteiket az ajánlott szakirodalom alapján felfrissíthatik.
- Az UML2 az UML olyan kiterjesztése ahol 13 alapvető diagramtípushat definiáltak, amelyek két nagy csoportba sorolhatók.
  - Strukturális modellezést támogató diagramok
  - Viselkedést támogató diagramok
  - Ismereteiket elmélyíthetik az ajánlott szakirodalom alapján.

# Első csoport

Az UML2 strukturális diagramjai

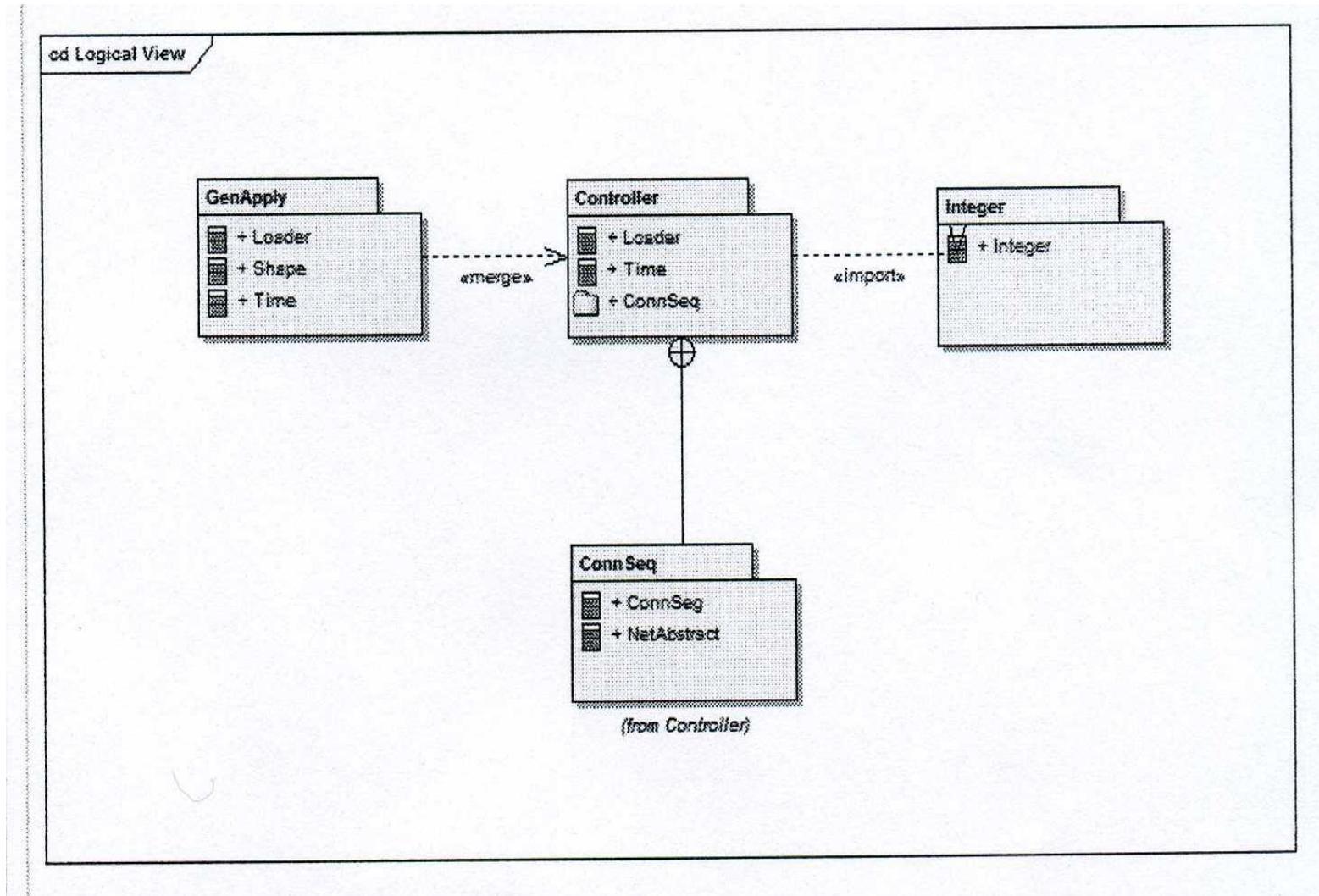
# Az UML2 strukturális modellező diagramjai

- Egy modell statikus architektúráját strukturális diagramok segítségével definiálhatjuk.
- Egy modell alapvető összetevői
  - az osztályok, objektumok, interfészek és komponensek.
- A strukturális modellek definiálják a modell elemei közötti kapcsolatokat (relációkat) és a közöttük fennálló függőségeket is.
- A strukturális modell megalkotásához hatféle diagramot használhatunk:
  - Csomag (package) diagramok
  - Osztály (class) diagramok
  - Objektum diagramok
  - Összetett (composite) strukturális diagramok
  - Komponens diagramok
  - Telepítési (deployment) diagramok

# UML2: Csomag (Package) diagramok

- Ezen diagramok segítségével írhatjuk le a csomagok szervezését és elemeit. A használati eset (use case) diagramok és az osztálydiagramok létrehozásában is fontos szerepük van. A csomagban lévő elemek névtere azonos.
- A csomagok között úgynevezett konnektorokat definiálhatunk
  - Package Merge
    - Ez a konnektor két csomag között implicit általánosítást definiál. A forrás csomag elemeinek definícióját kiterjeszthetjük a cél csomag elemeinek definícióival.
  - Package Import
    - Ez a konnektor azt jelzi, hogy a cél csomag elemei a forrás csomag neveit minősítés nélkül használják, amikor azokra hivatkoznak.
  - Nesting connectors
    - Ez a konnektor azt jelzi, hogy a forrás csomagot a cél csomag tartalmazza
- A következő dián példát mutatunk csomag diagramra

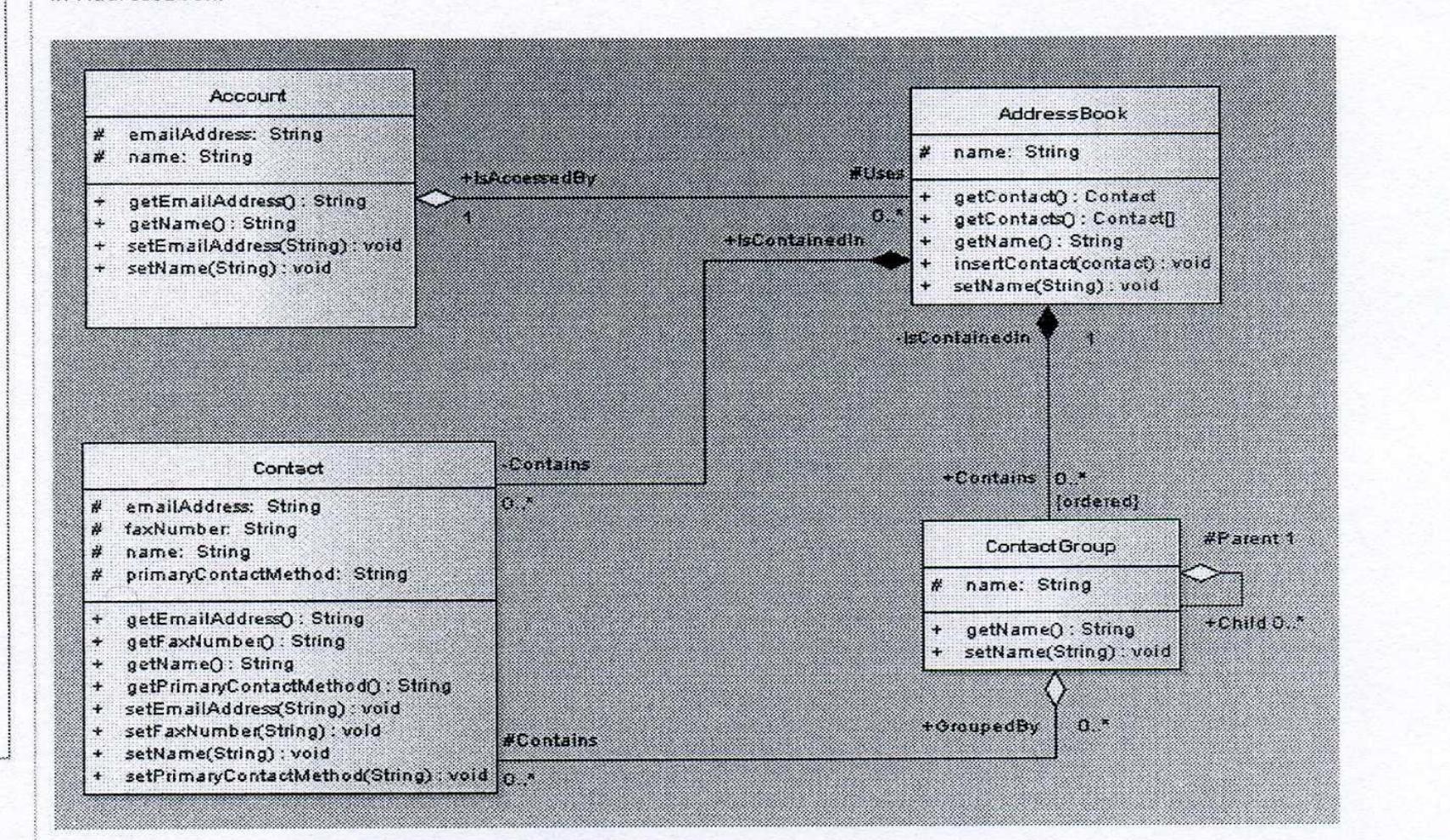
# UML package diagramra példa :



# UML2: Osztálydiagramok

- Az osztálydiagramokkal a rendszer statikus felépítését definiálhatjuk.
- A rendszert alkotó osztályok és interfészeik közötti kapcsolatok (relációk) leírását teszik lehetővé.
- Az osztályok között a következő relációk állhatnak fenn:
  - általánosítás (generalization),
  - aggregáció (aggregation),
  - asszociáció (association),
  - öröklődés (inheritance),
  - kompozíció (composition).
- Az osztálydiagram tehát a problématerben a megoldás szerkezetét leíró egyszeresen összefüggő gráf, amelyeknek csomópontjaihoz osztályokat, éleihez pedig relációkat rendelünk.
  - Sike Sándor, Varga László: Szoftvertechnológia és UML. ELTE Eötvös kiadó, 2003.
  - <http://uml.org/#UML2.0>

# Osztály- és objektumdiagramra példa



# Jelölések

- A világos fejű (lighter) aggregáció azt jelzi, hogy az „Account” osztály használja az AddresBook osztályt, de nem feltétlen tartalmazza annak egy példányát.
- A sötét fejű (strong, composite) aggregáció azt jelzi, hogy a célosztály birtokolja a forrásosztályt.
  - Például az előző ábrán az AddresBook tartalmazza a Contact és Contactgroup értékeit.

# Osztályok (classes) jelölése

cd Class Details

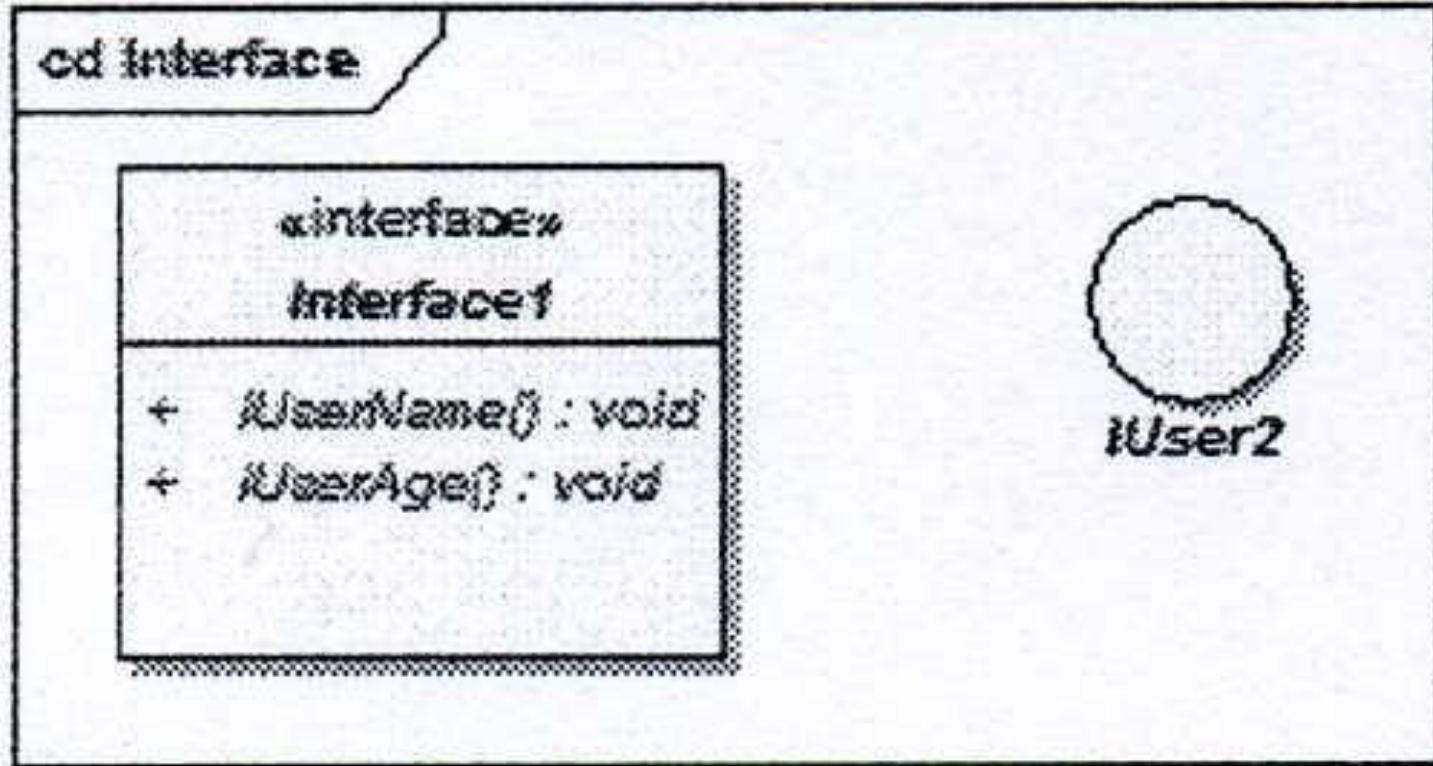
**Rectangle**

- length: double
- width: double
- center: Point = {10,10}

- + display(): void
- + remove(): void
- + setWidth(newWidth): void
- + setLength(newLength): void
- + setPosition(pos: Point): void

# Osztály interfész (class interface)



# Táblák (tables)

Nem része az alap UML-nek.

<<column>> attribútum; <<PK>> primary key; <<FK>> foreign key

cd Table

Customer

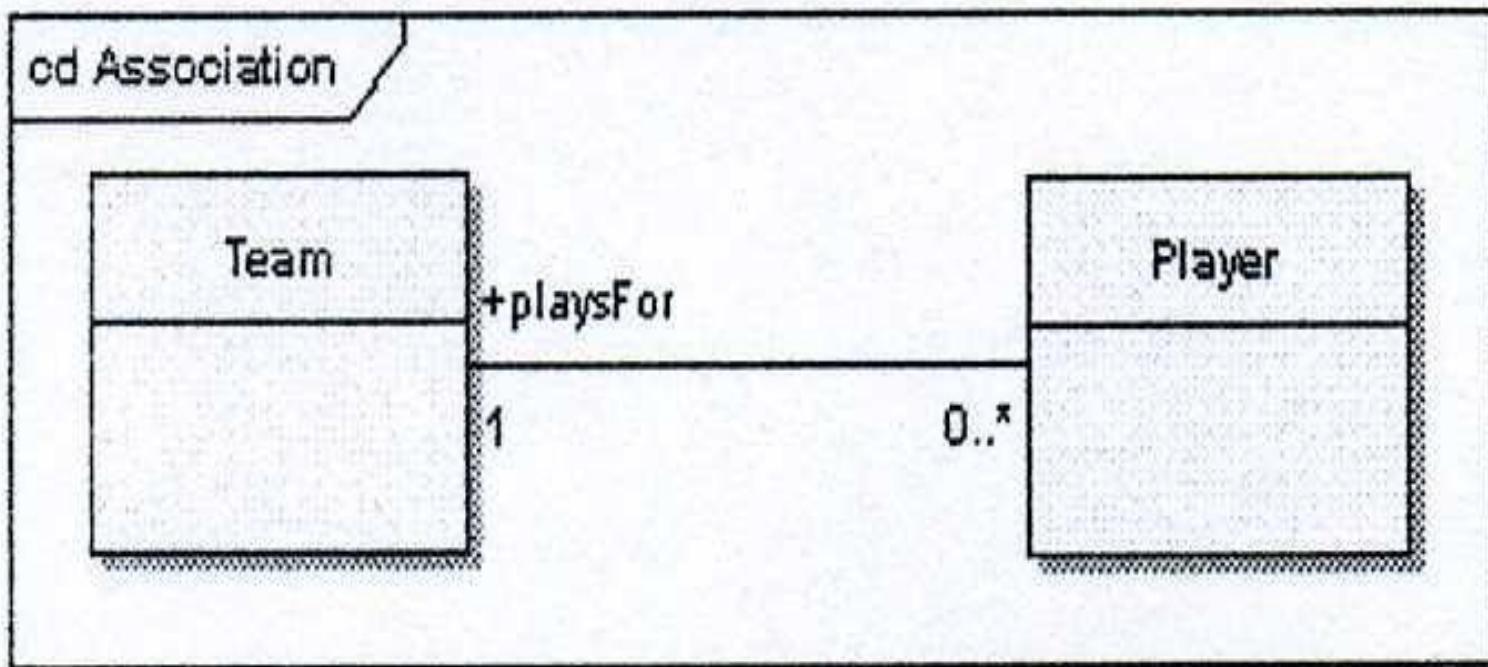


\*PK «column» CustomerID: Long  
«column» LastName: Text(50)  
«column» FirstName: Text(50)  
«column» Address: Memo

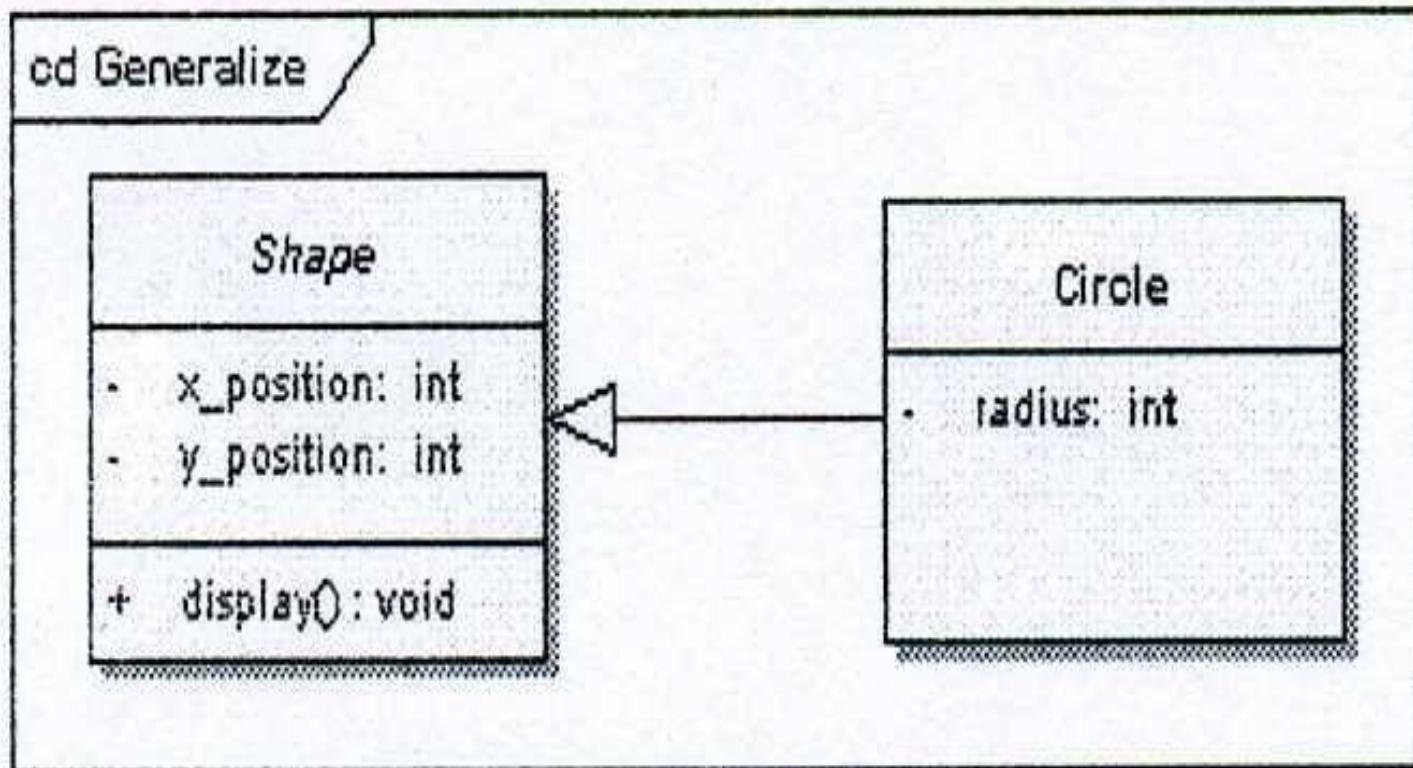
+ «PK» PK\_Customer(Long)

# Asszociációk (associations)

"playsFor" will become an instance variable in the "Player" class.



# Általánosítás (generalizations)



# Általánosítás ekvivalens ábrázolása

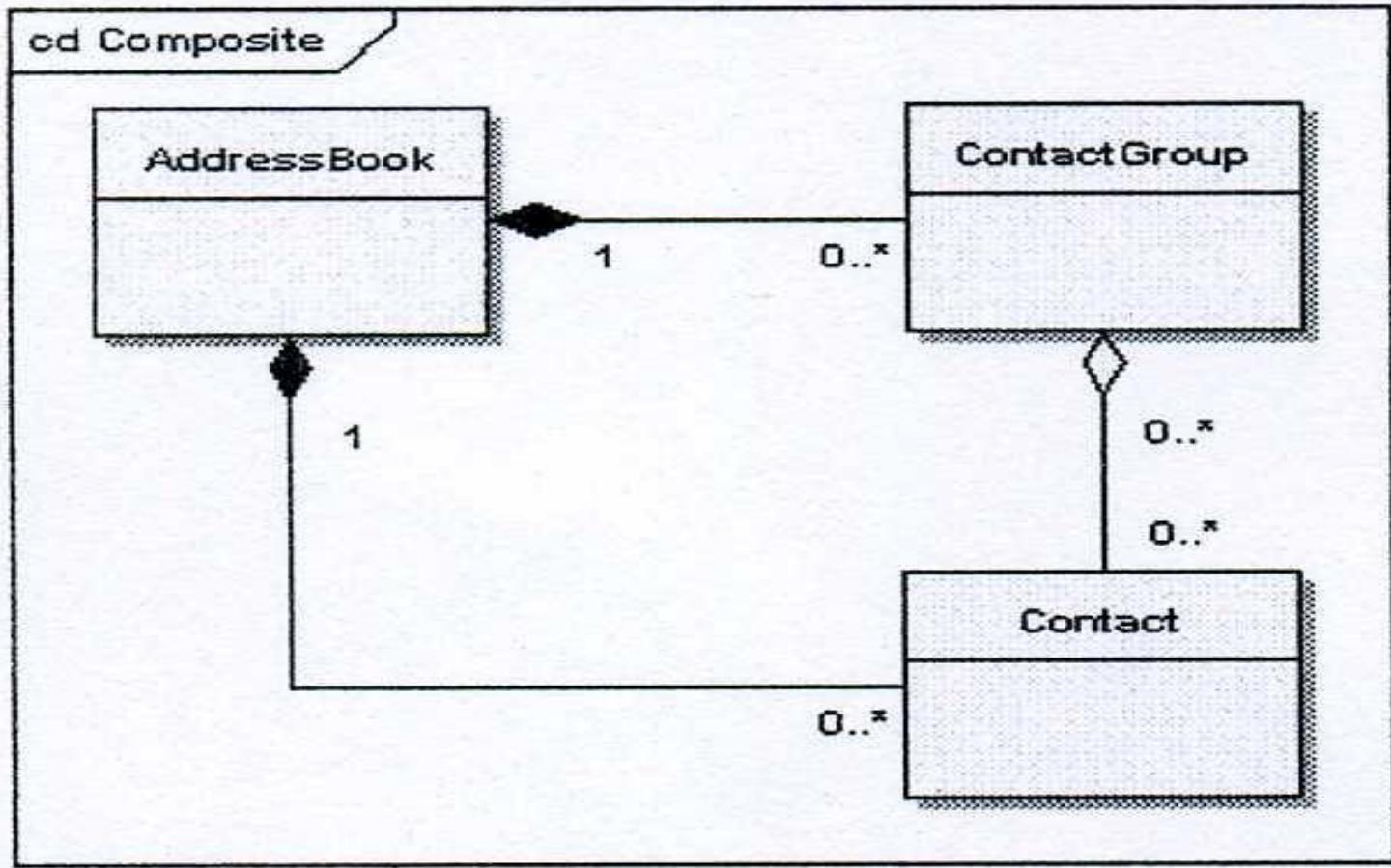
```
cd Generalize
```

*Shape*

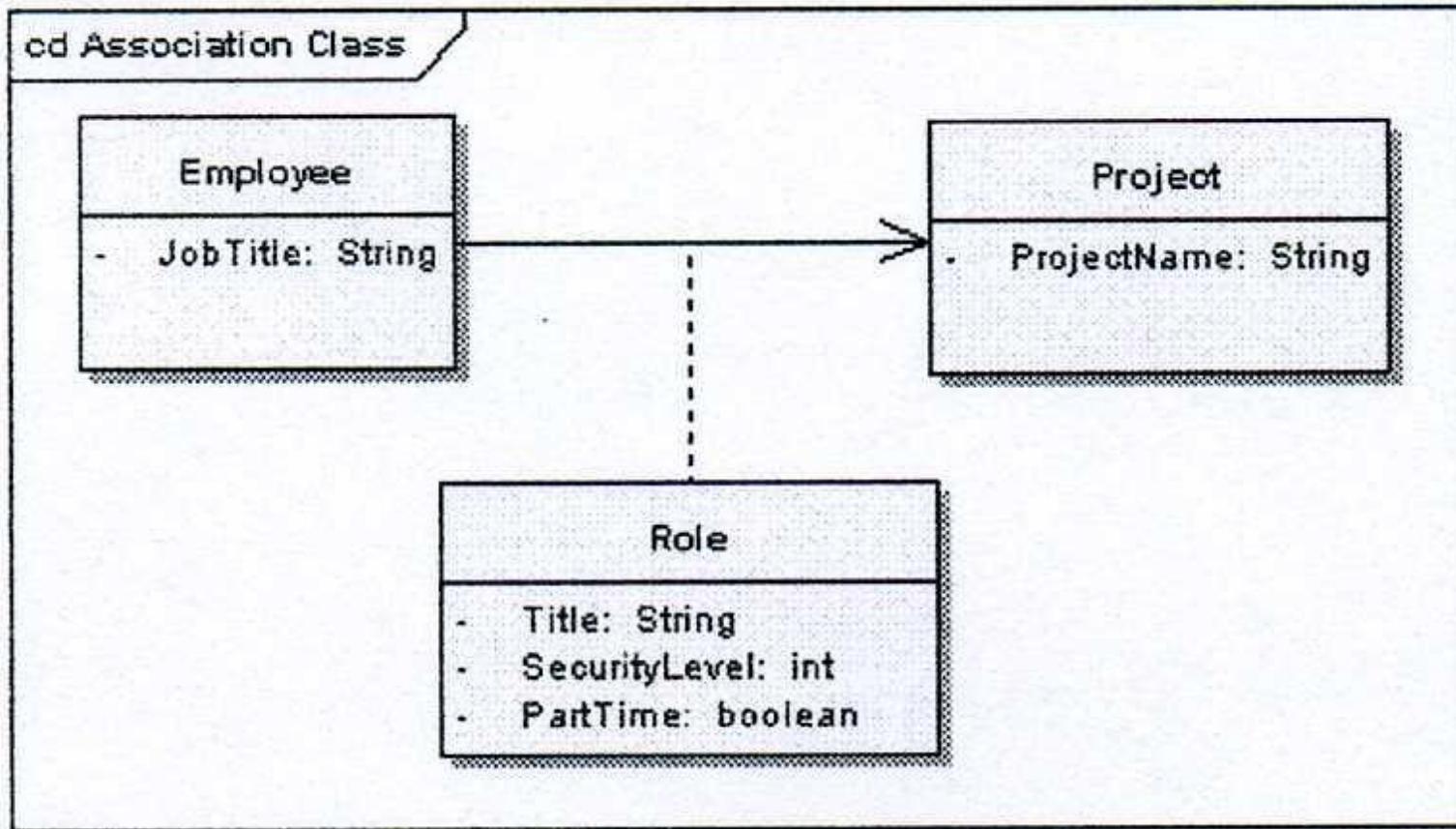
*Circle*

```
- radius: int
::Shape
- x_position: int
- y_position: int
::Shape
+ display(): void
```

# Aggregáció (aggregations)



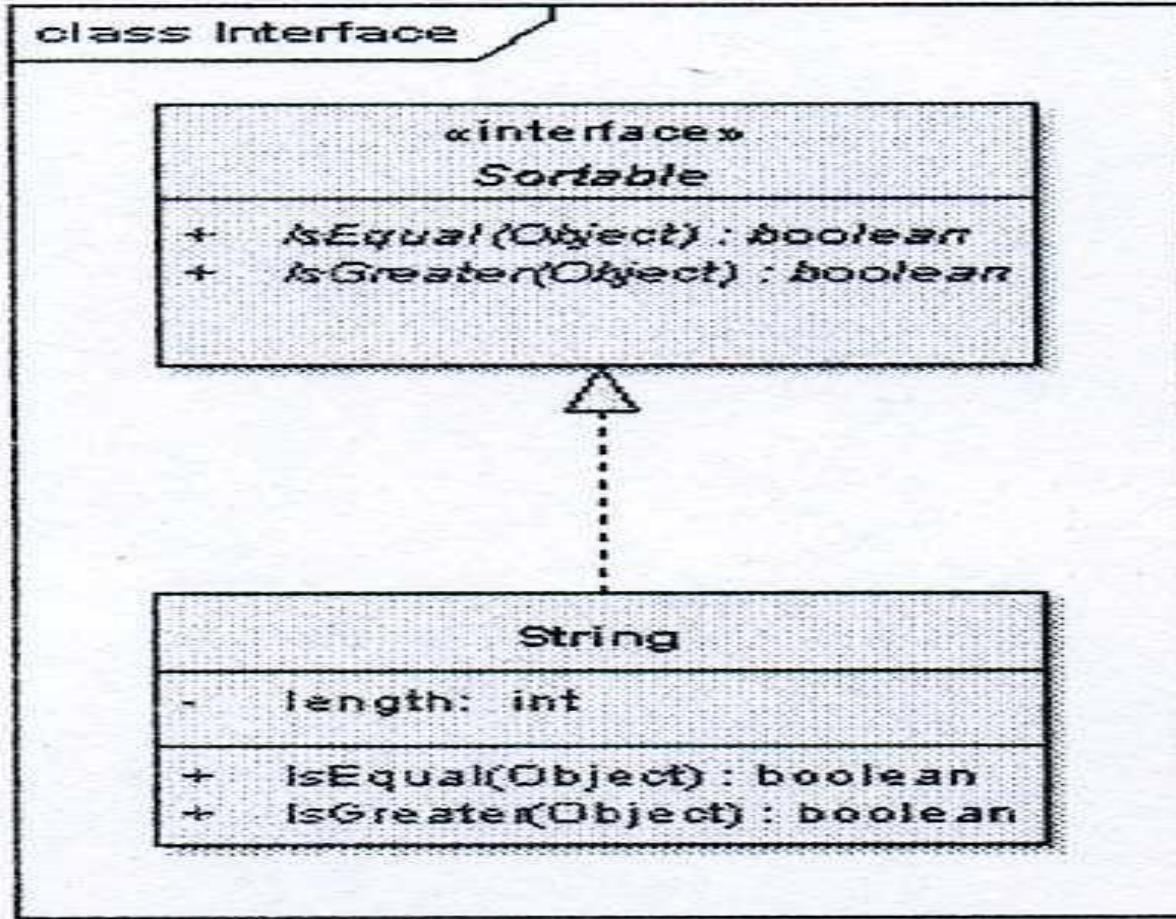
# Osztály asszociáció (association class)



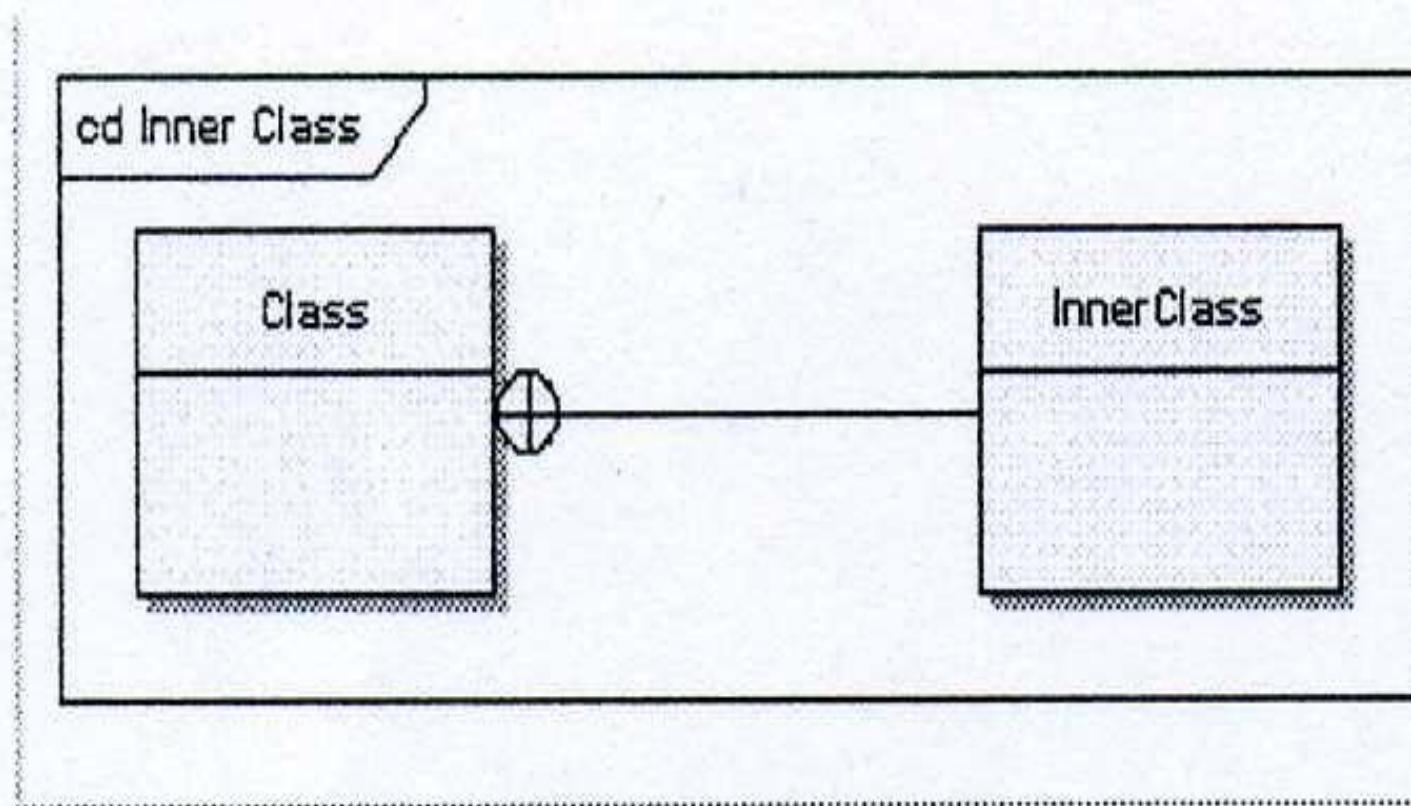
# Realizációk (realizations)

A forrásobjektum megvalósítja a célobjektumot

http://www.cs.vt.edu/~shankar/teach/CS501/lectures/lec10.pdf



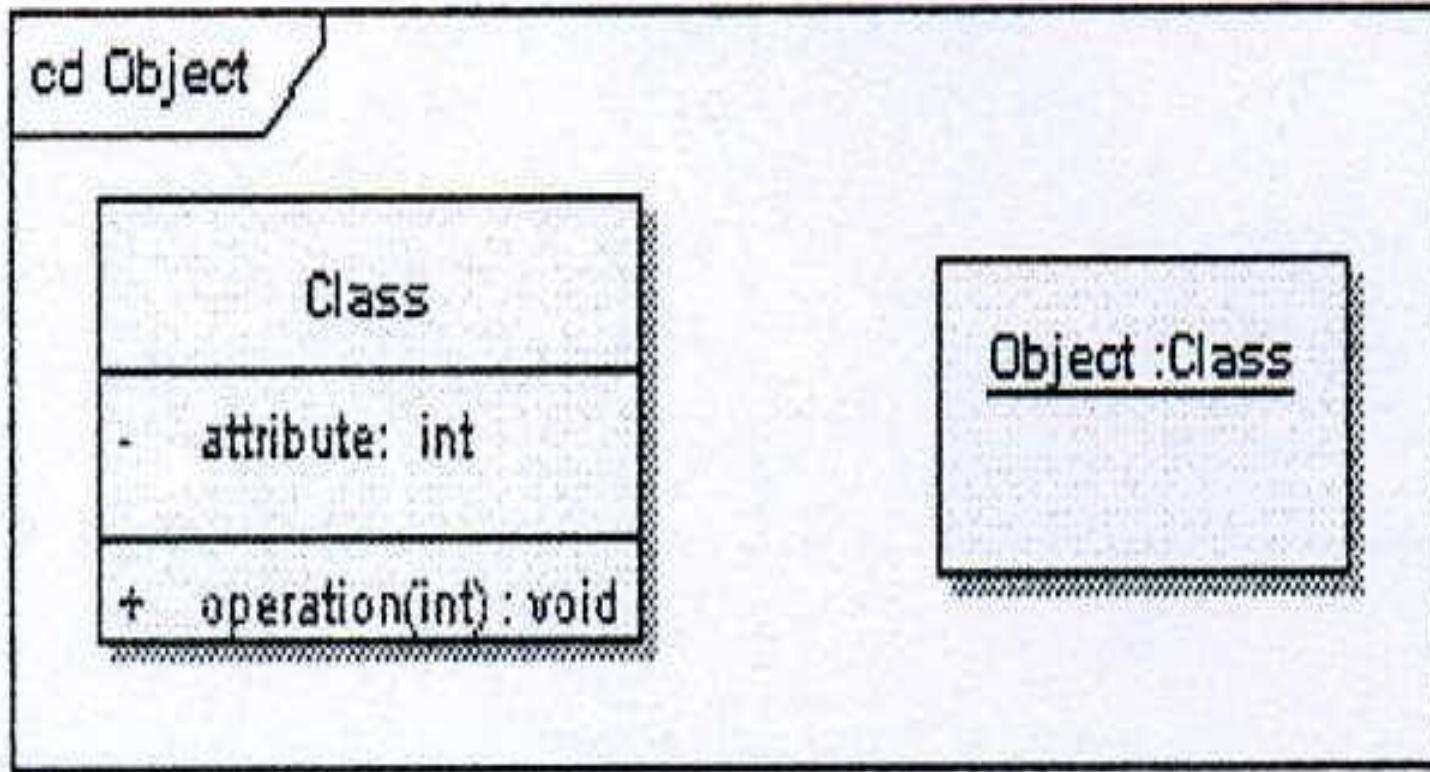
# Beágyazások (nestings)



# UML2: Objektumdiagramok

- Az osztálydiagramok speciális esetei.
- Az osztályok példányai közötti, adott pillanatokban érvényes kapcsolatokat hangsúlyozhatjuk ki.
- Az objektumdiagramok és az osztálydiagramok közötti különbségekre világít rá a következő példa.
  - Az osztályoknak három fő része van: név, attribútumok, műveletek.
  - Az objektumnak csak a nevét adjuk meg és azt, hogy melyik osztály példánya.

# Osztály és objektumelemek



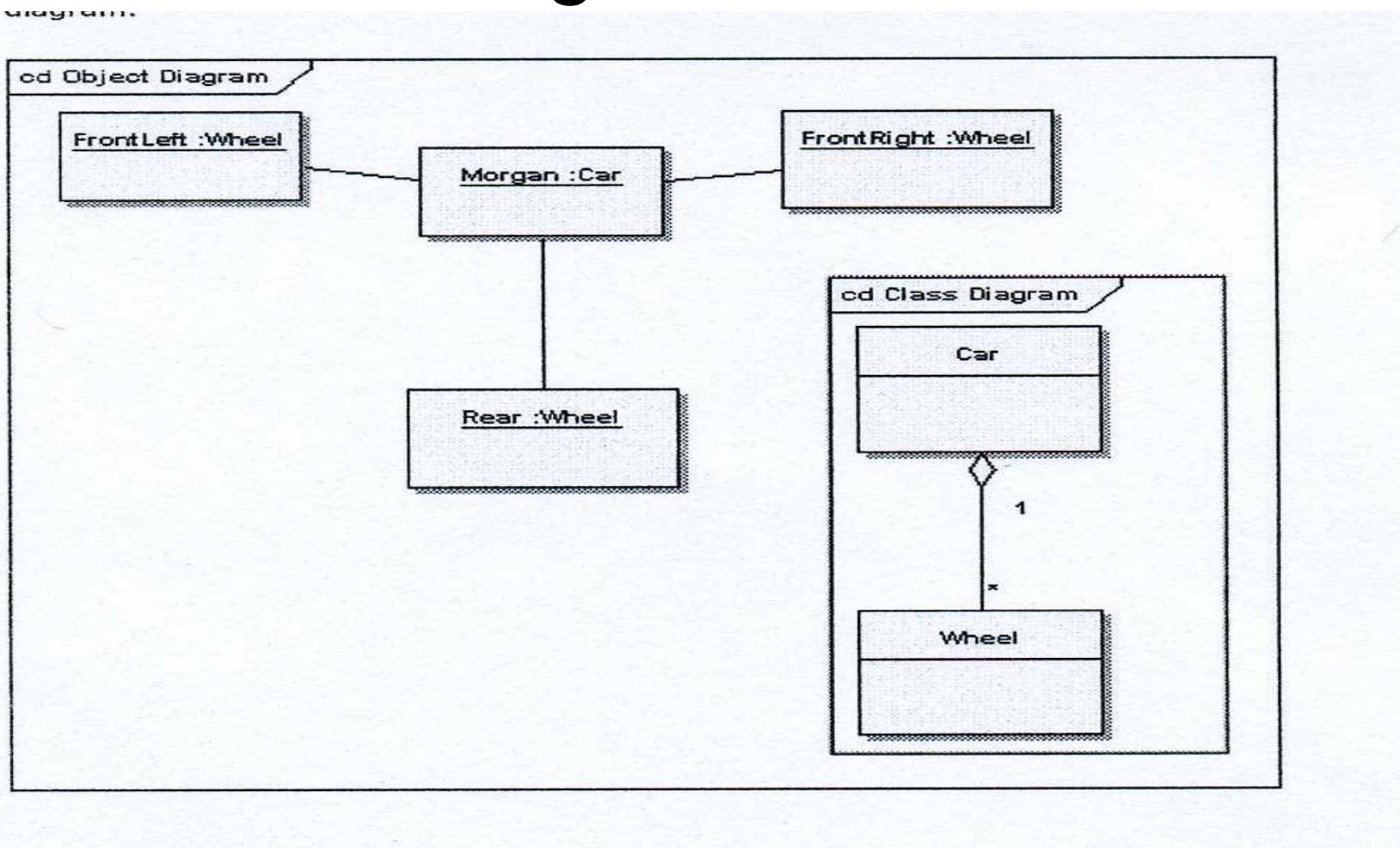
# Futás idejű állapot (run time state)

```
cd Run Time State
```

Manager :Employee

```
last_name = "Smith"  
first_name = "John"  
age = 42
```

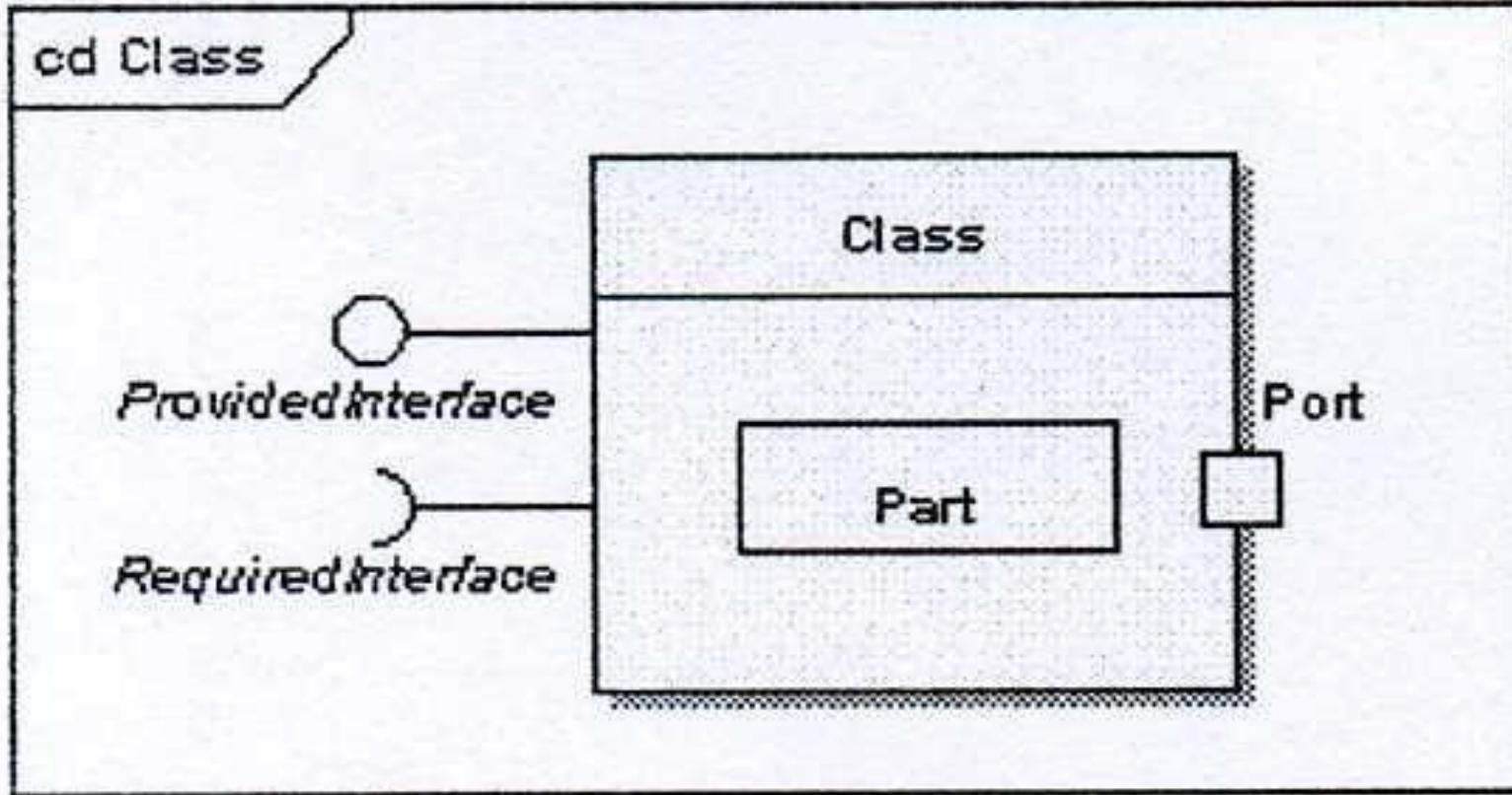
# Példa osztály és objektum diagramokra



# UML2: Összetett (composite) diagramok

- Ezen diagramokkal egy osztály belső struktúráját definiálhatjuk és azokat a pontokat, ahol a rendszer más elemeivel kapcsolatba léphetnek.
- Ezeket a diagramokat részletesebben tárgyaljuk továbbiakban.
- A composite diagram egy osztályának definiálására szolgáló általános sémát a következő ábrán mutatjuk be.

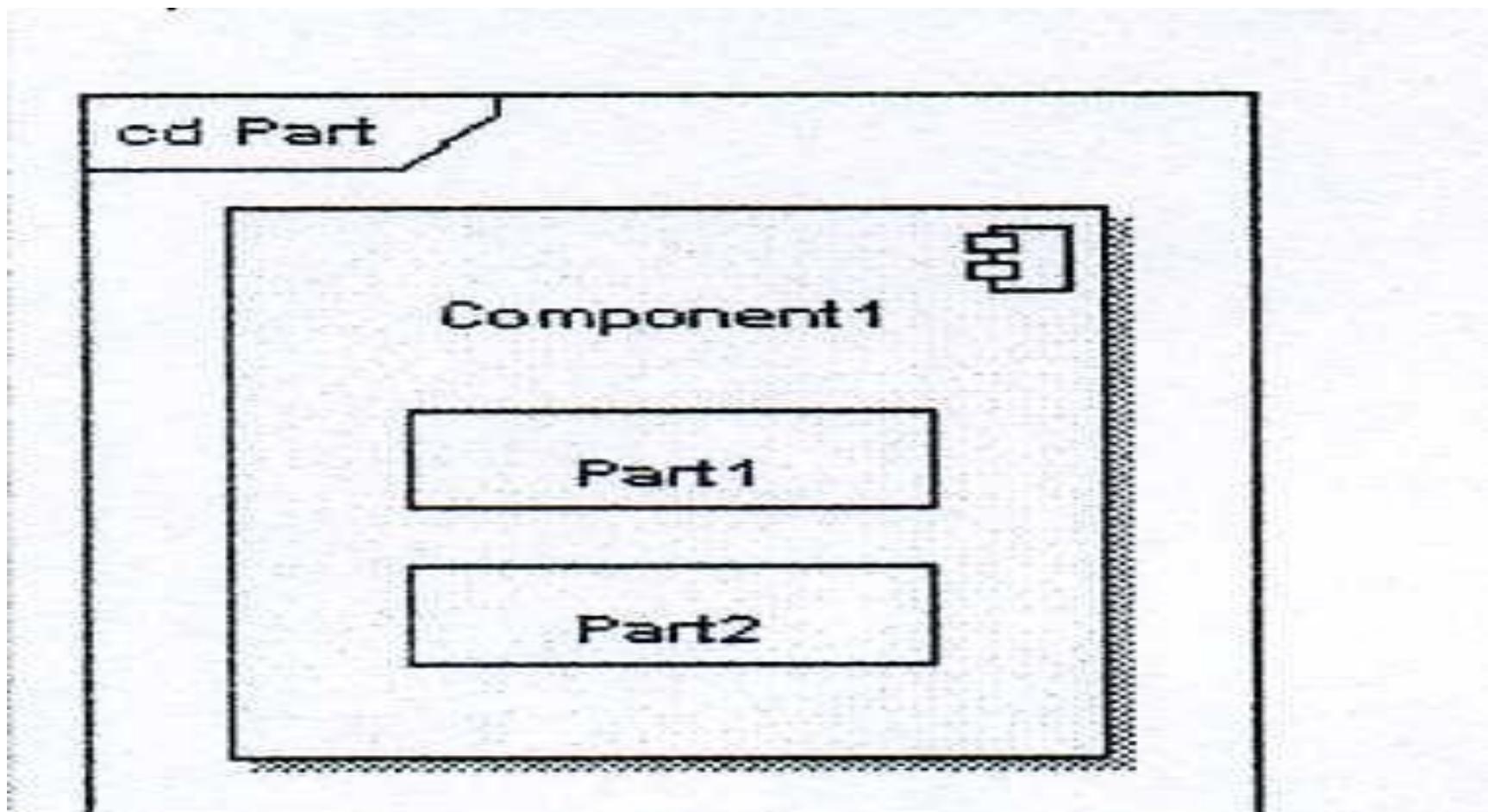
# UML2 Composite diagramjának egy osztálya



# Rész (part)

- Egy rész (part) egy osztály egy olyan eleme, amely egy vagy több olyan példányt reprezentál, amelyeket az adott osztály példánya birtokol.
- Egy rész grafikus megjelenítése a következő ábrán látható.

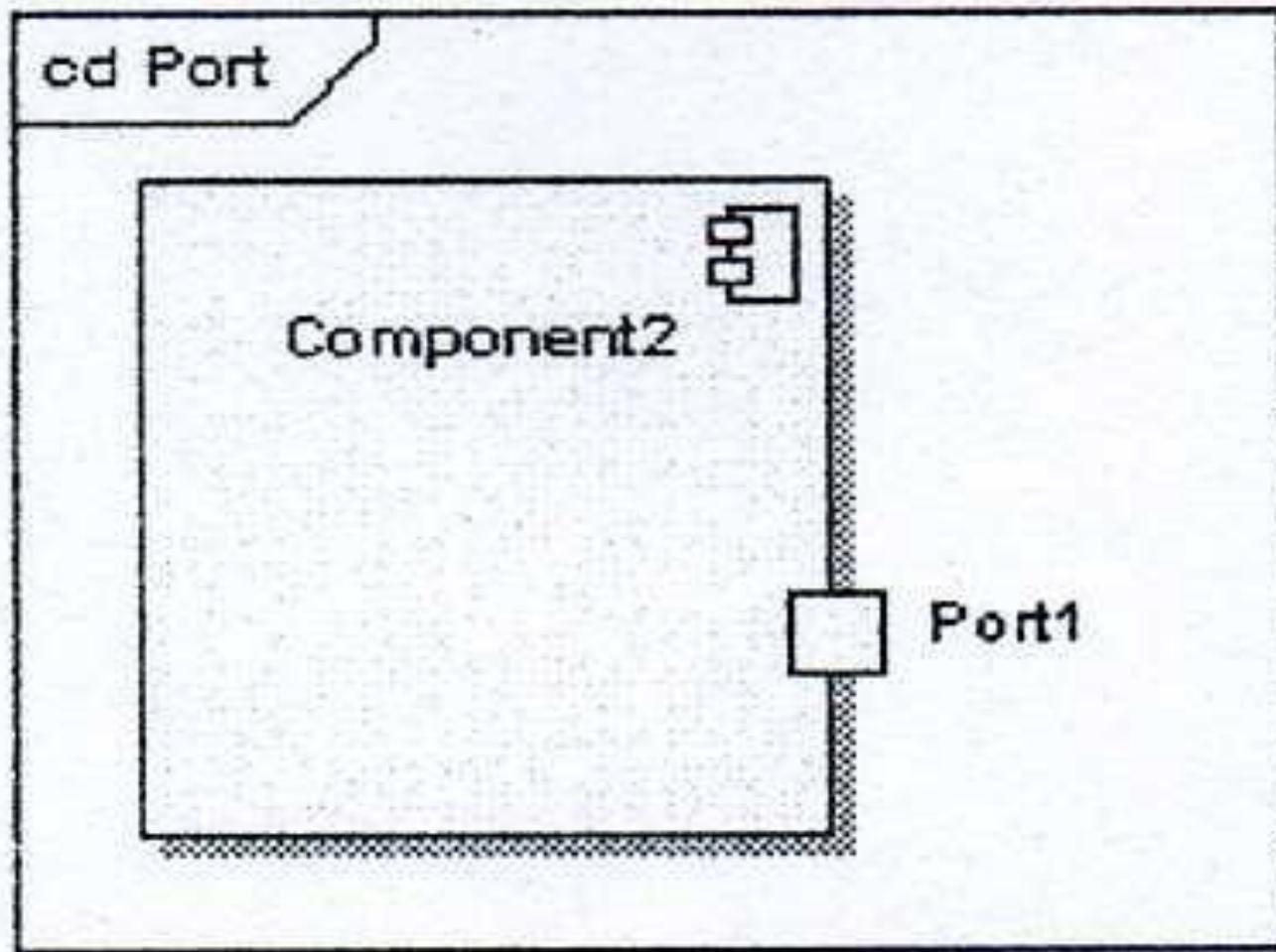
# Egy rész grafikus ábrázolása



# Kapu (port)

- Egy kapu egy típussal rendelkező elem, amely egy kívülről látható részét reprezentálja egy osztálypéldánynak.
- Egy kapu minden szolgáltatásokat specifikálhatja, amelyeket egy osztály szolgáltat a környezete számára és amelyeket elvár a környezetétől.

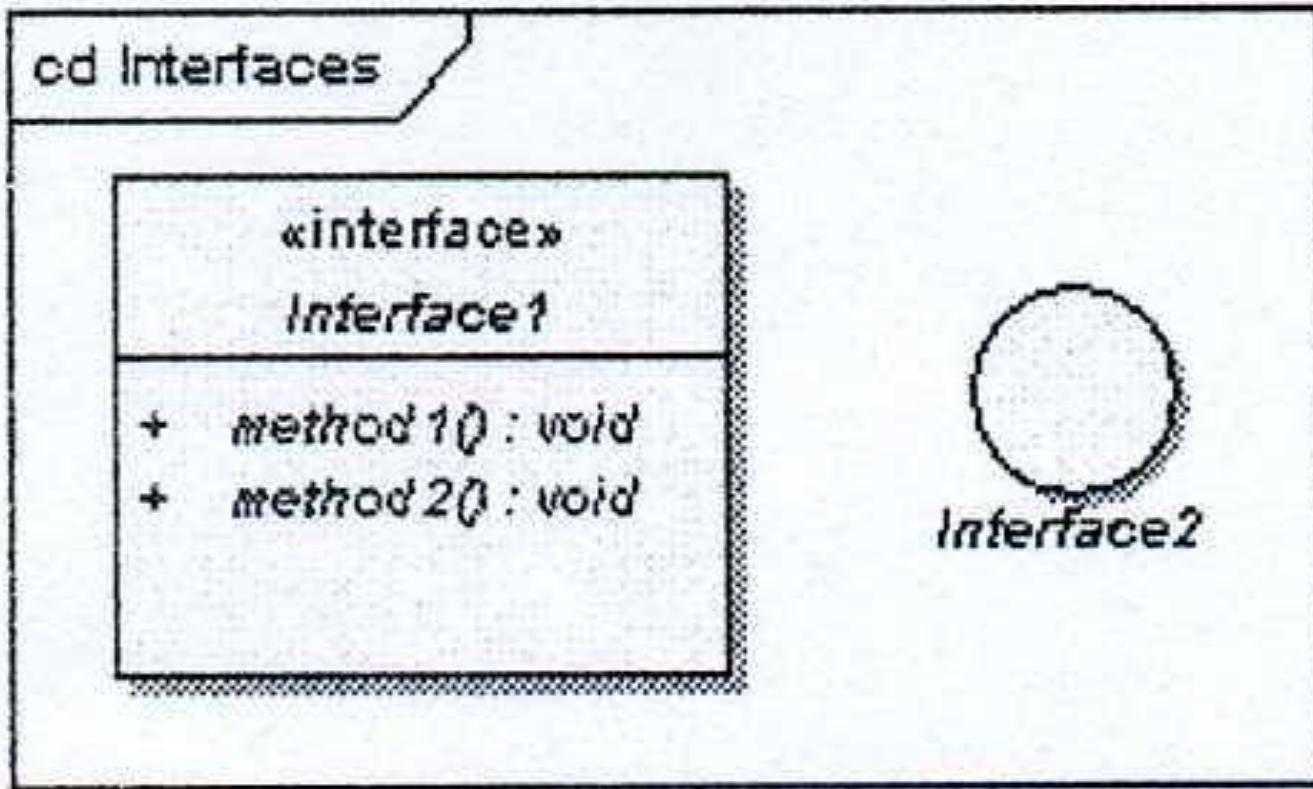
# A kapu grafikus megjelenítése



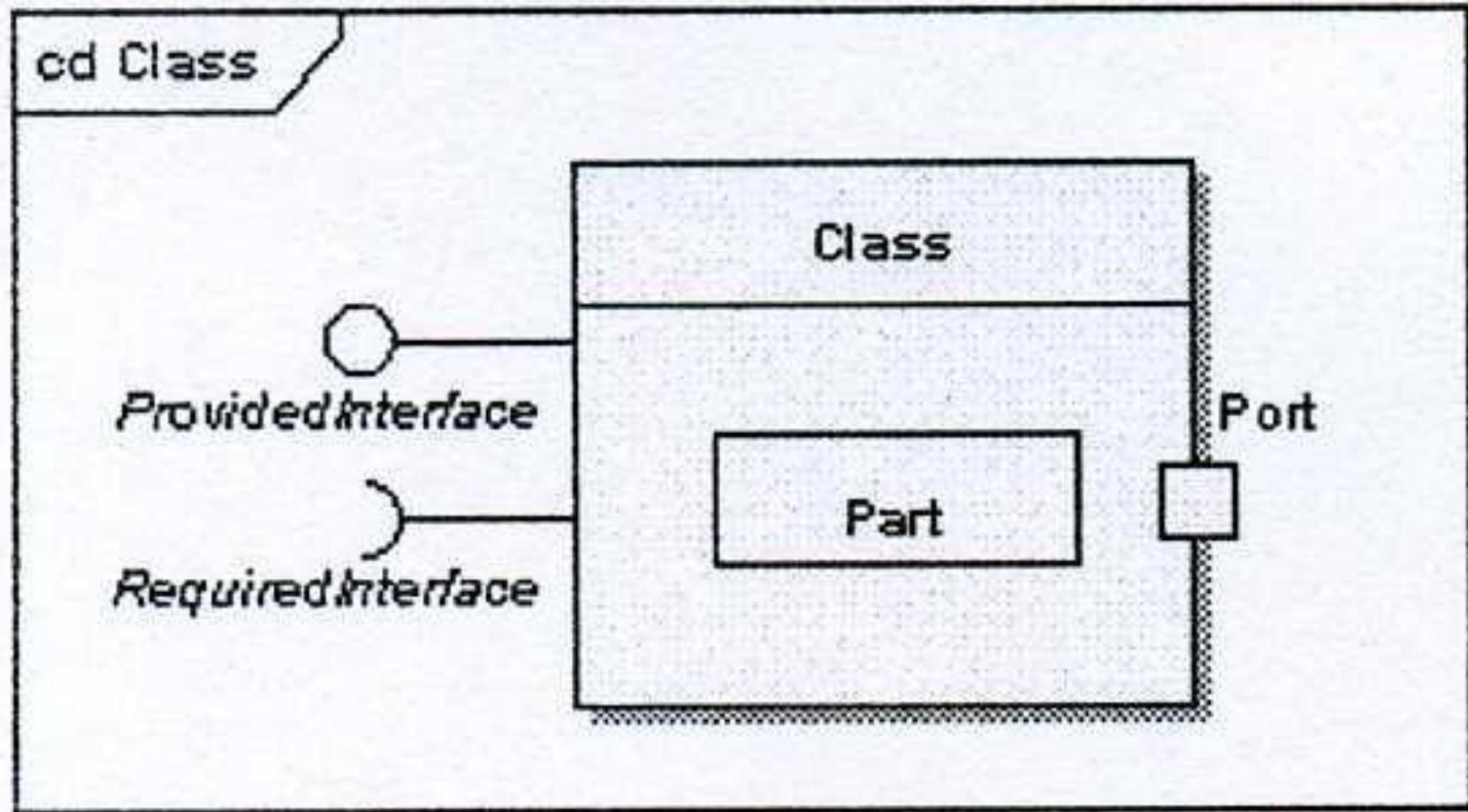
# Interfészek

- Egy interfész egy osztályhoz hasonló, de számos megszorítást tartalmaz.
  - minden interfész művelet nyilvános és absztrakt, de nincs alapértelmezett megvalósítása.
  - Az interfész minden attribútuma konstans.
  - Ha egy osztály csak egyetlen szuperosztálytól örökölhet, akkor megvalósíthat többszörös interfészt.
- Az interfész grafikus megjelenítése kétféle módon történhet, ahogy azt a következő ábrán láthatjuk.

# Interfészek grafikus ábrázolása



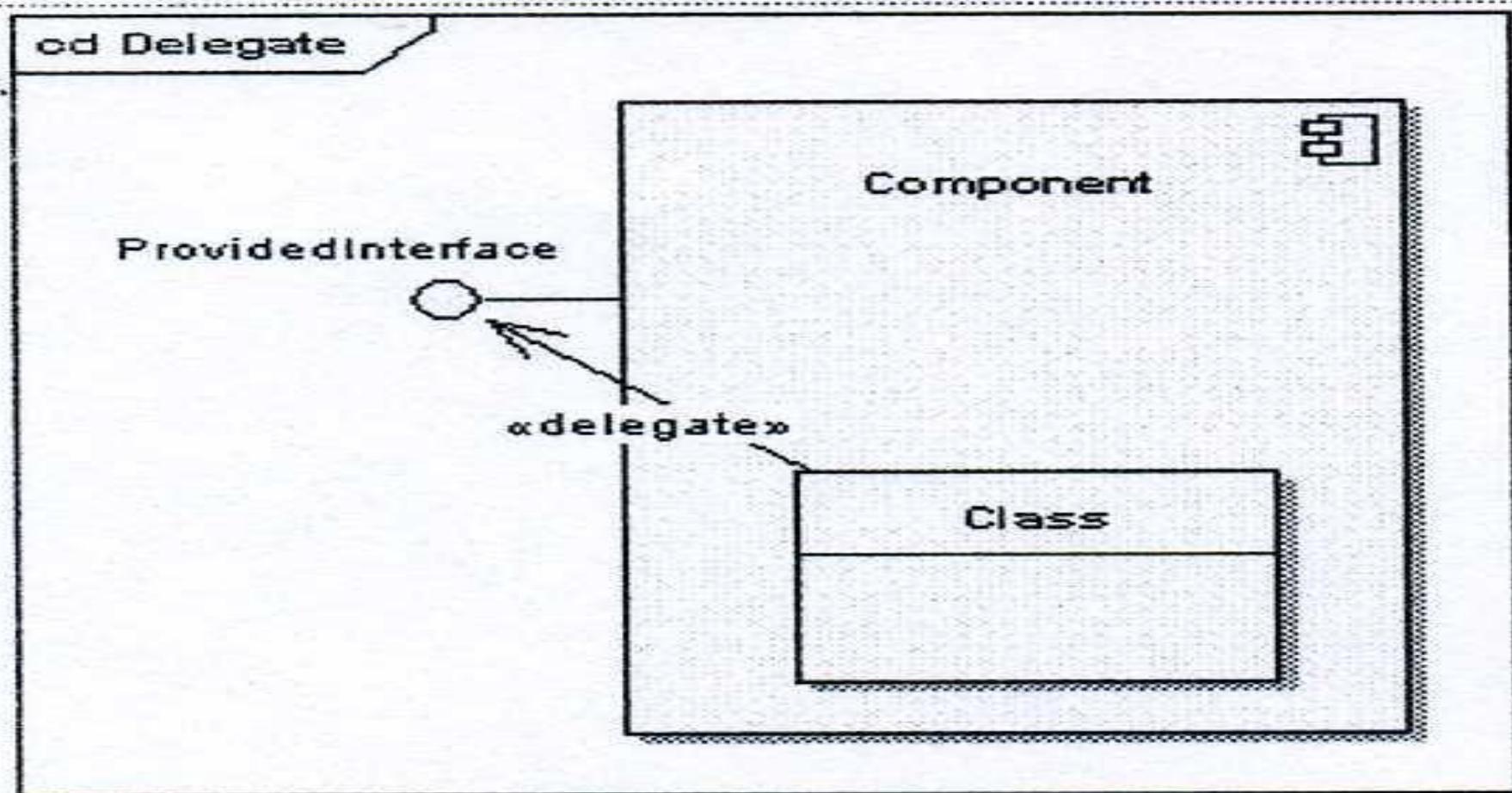
# A szolgáltatott és elvárt interfések grafikus megjelenítése



# Delegáció (delegate)

- A delegáció egy olyan konnektor, amellyel egy komponens külső kapuinak és interfészeinek belső működését definiálhatjuk.
- A delegáció grafikus megjelenítését a következő ábra mutatja.

# Delegáció megjelenítése

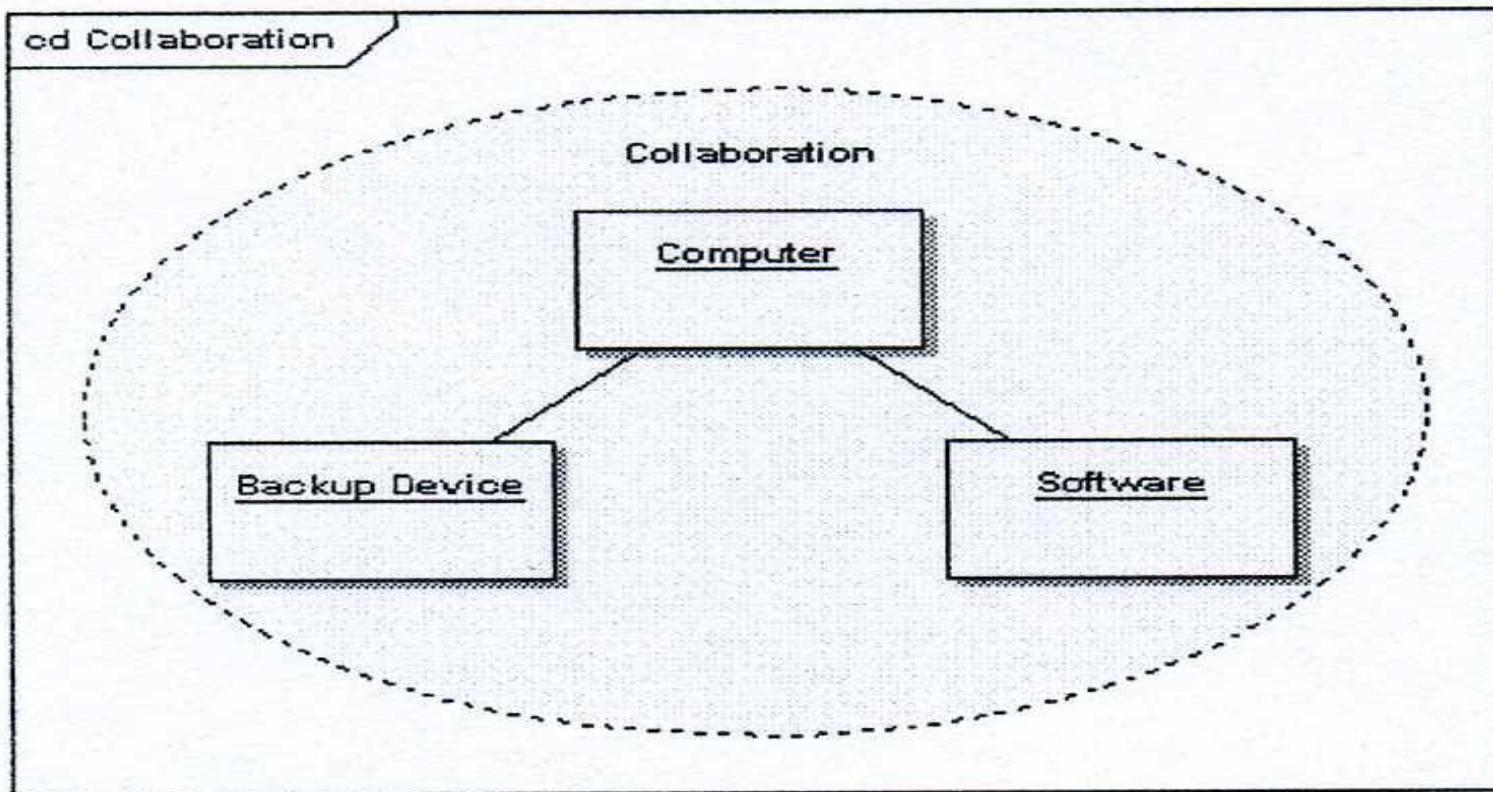


# Együttműködés (collaboration)

- Egy együttműködés kooperáló szerepek egy halmazát definiálja, amelyeket együttesen arra használunk, hogy egy specifikus funkcionálitást illusztráljanak.
- Egy együttműködés gyakran egy mintát valósít meg.
- Az együttműködés csak azon szerepeket és attribútumokat mutatja meg, amelyek a kívánt funkcionálitás végrehajtásához szükségesek.
- Az együttműködést egy ellipszissel ábrázoljuk, ahogy azt a következő ábra mutatja.

# Együttműködés megjelenítése

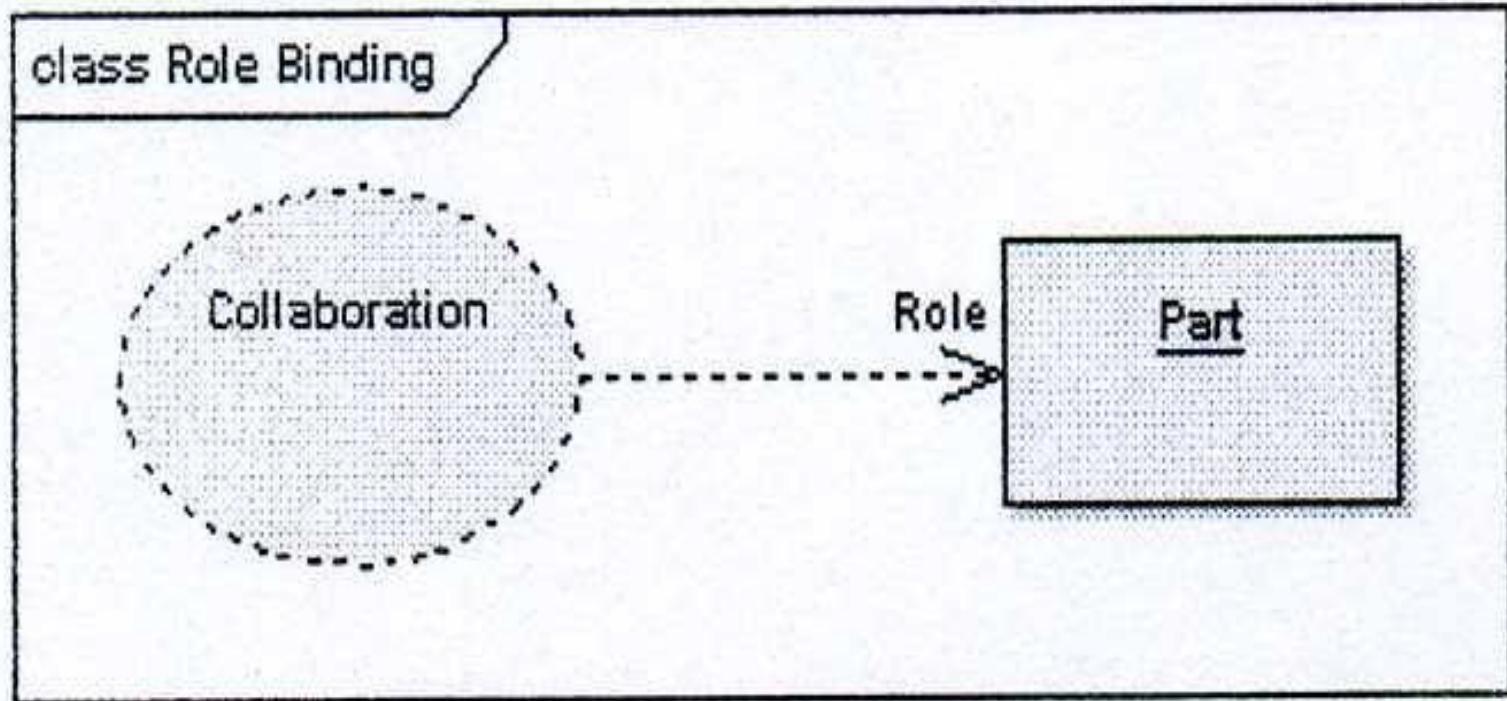
A collaboration element is shown as an ellipse.



# Szerepkötés (role binding)

- Egy olyan konnektor, amely egy együttműködés és egy osztály vagy annak része között teremt kapcsolatot. Az osztály teljesíti, valósítja meg az adott szerepet.
- A szerepkötés grafikus megjelenítését a következő ábra mutatja be.

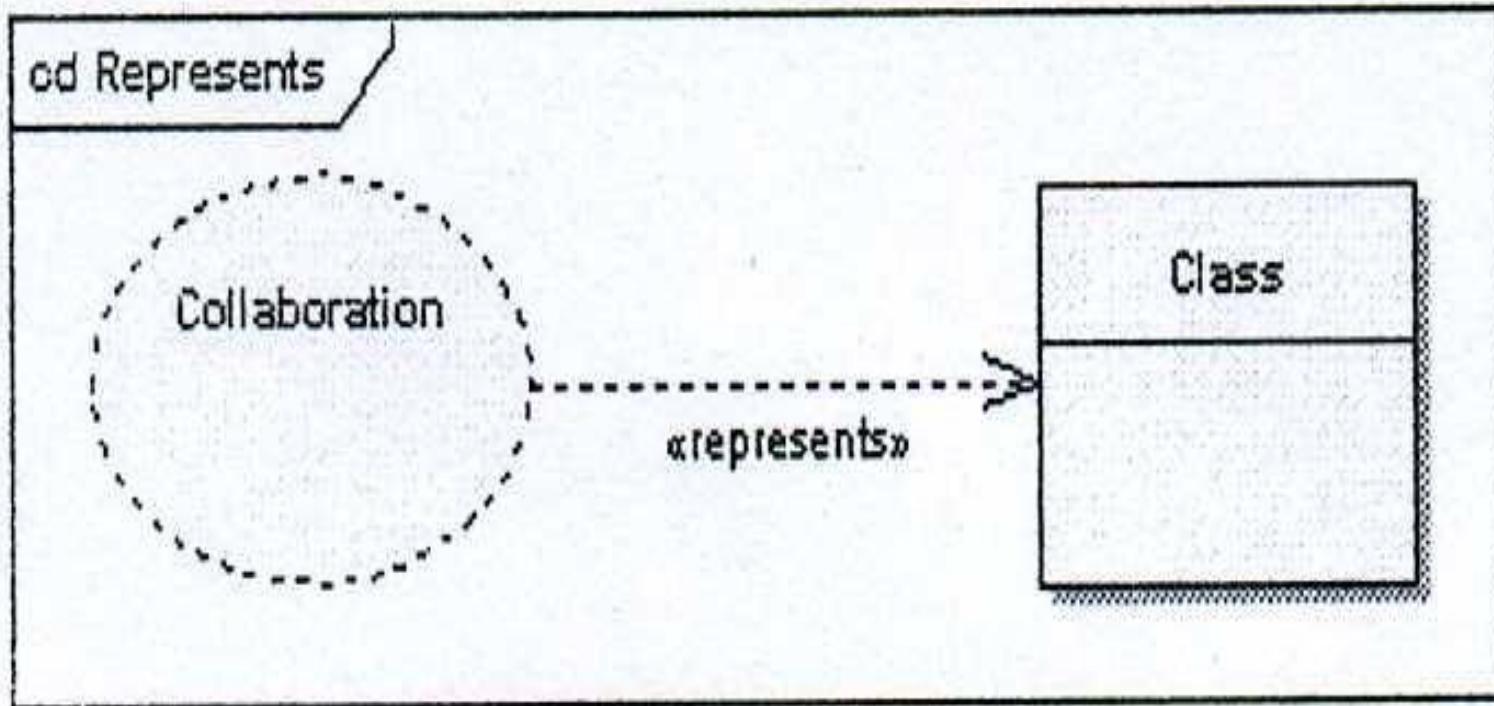
# Szerepkötés megjelenítése



# Reprezentáció (represents)

- A reprezentáció egy olyan speciális konnektor, amely egy együttműködés és egy osztály között hoz létre kapcsolatot, ahol az együttműködést az adott osztályon belül használjuk.
- A reprezentáció grafikus ábrázolását mutatja be a következő ábra.

# A reprezentáció grafikus megjelenítése

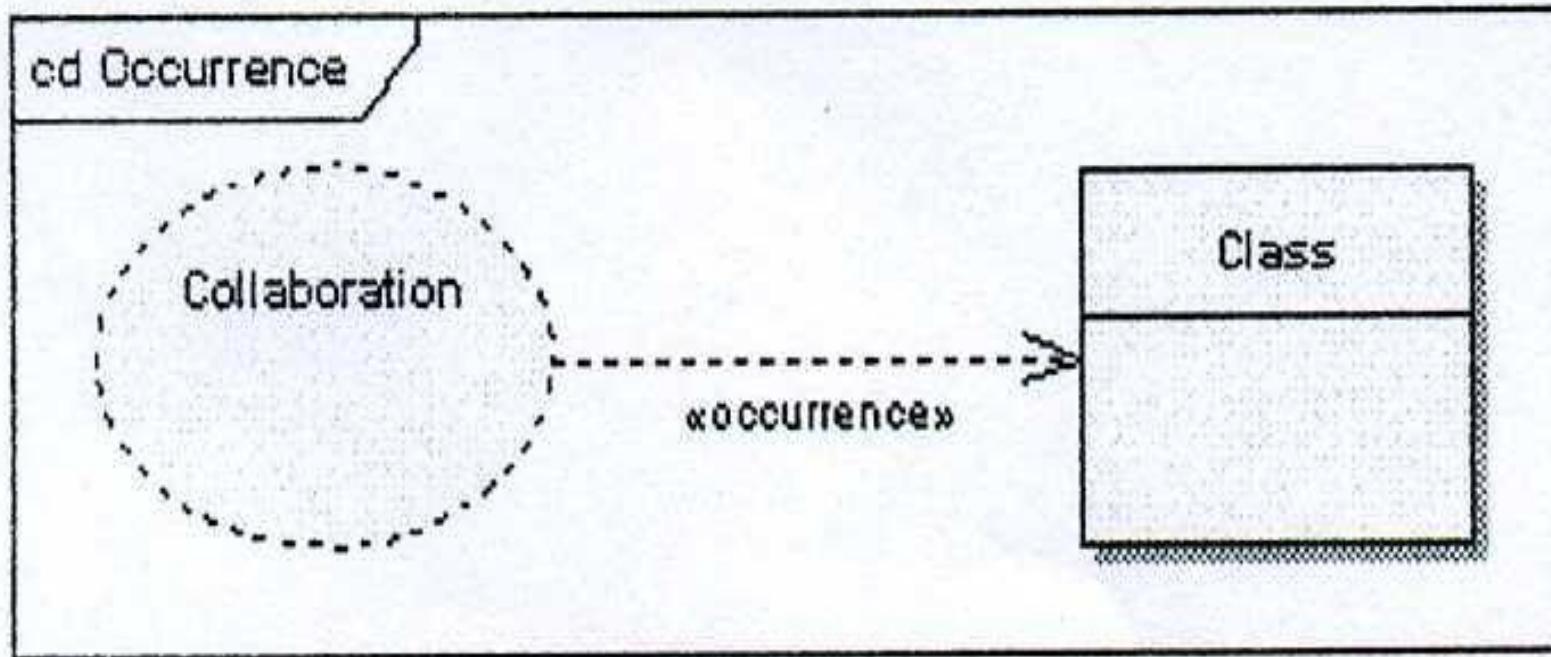


# Előfordulás (occurrence)

- Egy olyan konnektor, amely egy együttműködés és egy osztály között hoz létre kapcsolatot, ahol az együttműködés reprezentálja az adott osztályt.
- Az előfordulás megjelenítést a következő ábrán mutatjuk be.

# Előfordulás grafikus megjelenítése

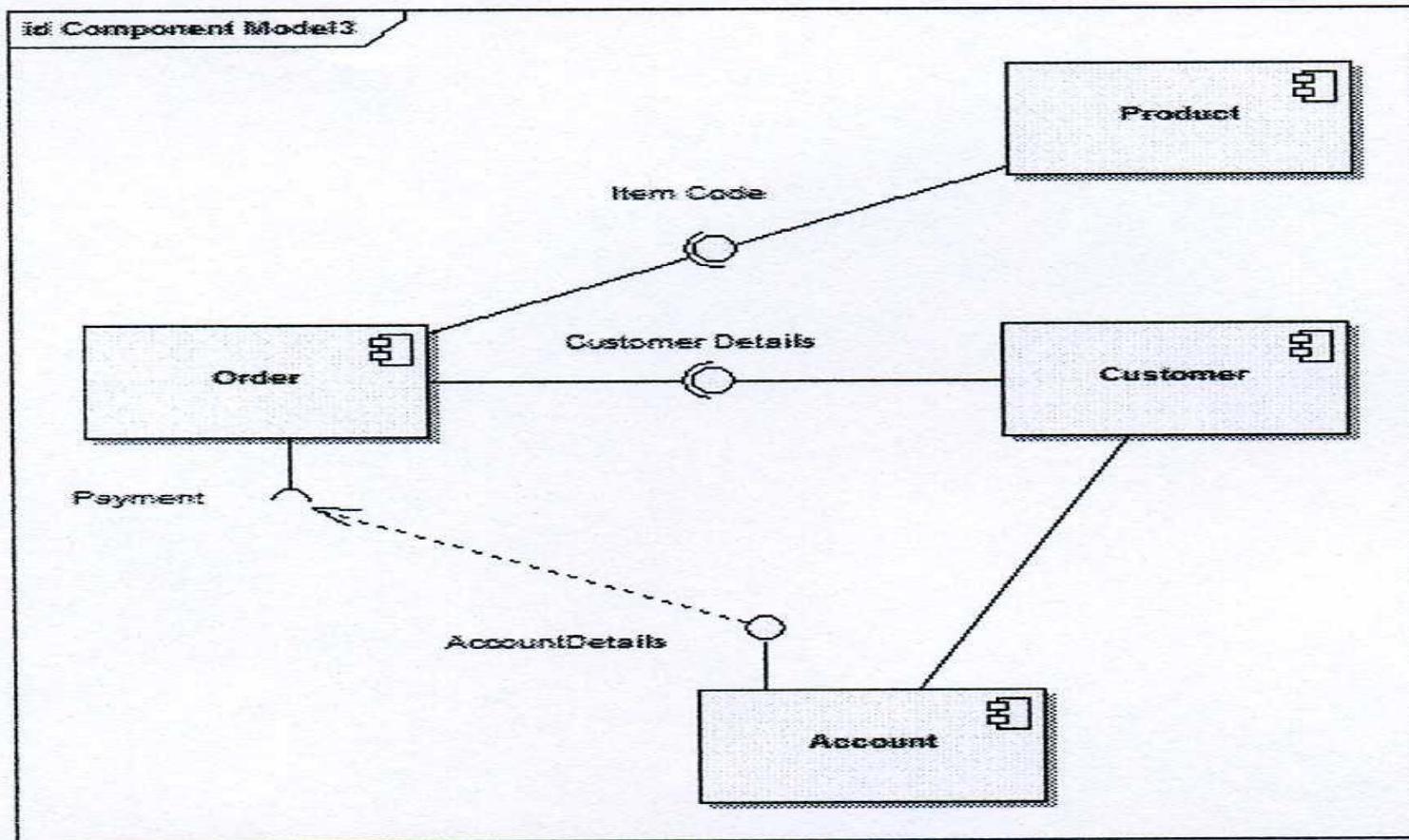
IS SHOWN AS A UML LINE WITH AN OCCURRENCE AND THE KEYWORD OCCURRENCE



# UML2: komponens diagramok

- A komponens diagramok a szoftver egy-egy szeletét mutatják be, amelyekből az egész rendszer összeáll.
- Egy komponens diagram absztrakciós szintje magasabb, mint egy osztálydiagramé. Egy komponens általában több osztállyal valósítható meg futási időben.
- Egy komponens diagram több komponenst és azok kapcsolatait jelenítheti meg, ahogy azt a következő példában láthatjuk.

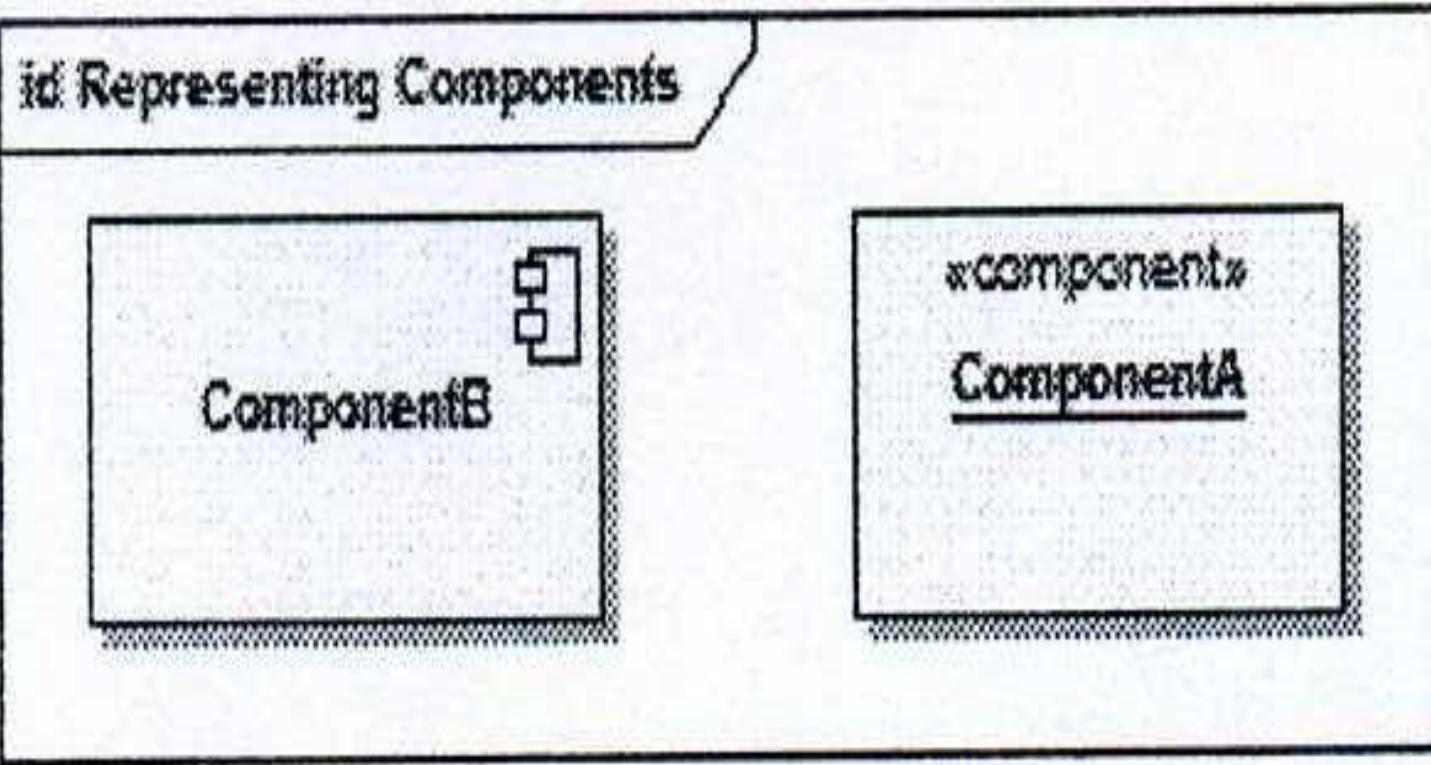
# Példa komponens diagramra



# Component Model3 példa magyarázata

- A „Product” és „Customer” komponensek szolgáltatják az „Order” komponens számára az interfészeket.
- Egy függőségi (dependency) relációra látunk példát az „Account” és az „Order” komponensek között.
  - A függőségi reláció képezi le az ügyfélhez (Customer) tartozó számla (Account) részleteit a megrendelés „Payment” névvel azonosított elvárt interfészére.

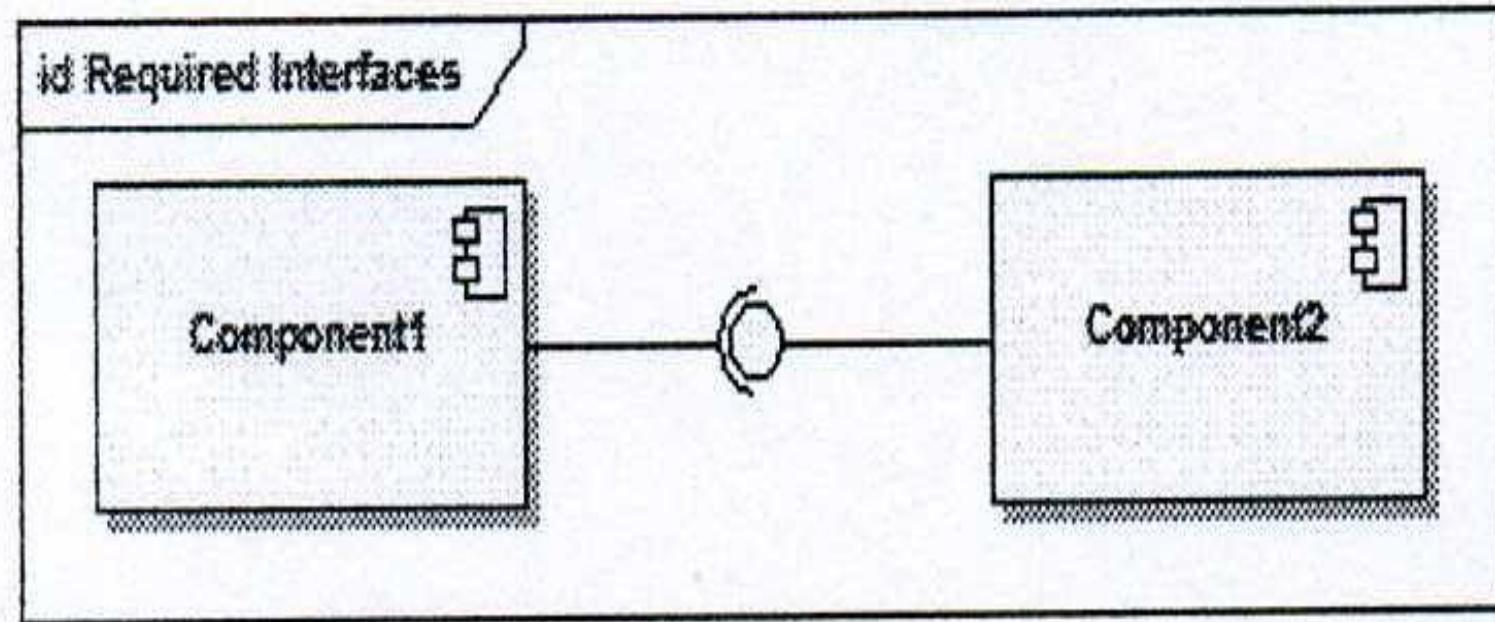
# Komponensek grafikus reprezentálása



# Összeszerelési konnektor (assembly connector)

Elvárt interfész: Component1; Szolgáltatott interfész: Component2

(Component2); this allows one component to provide the services that a

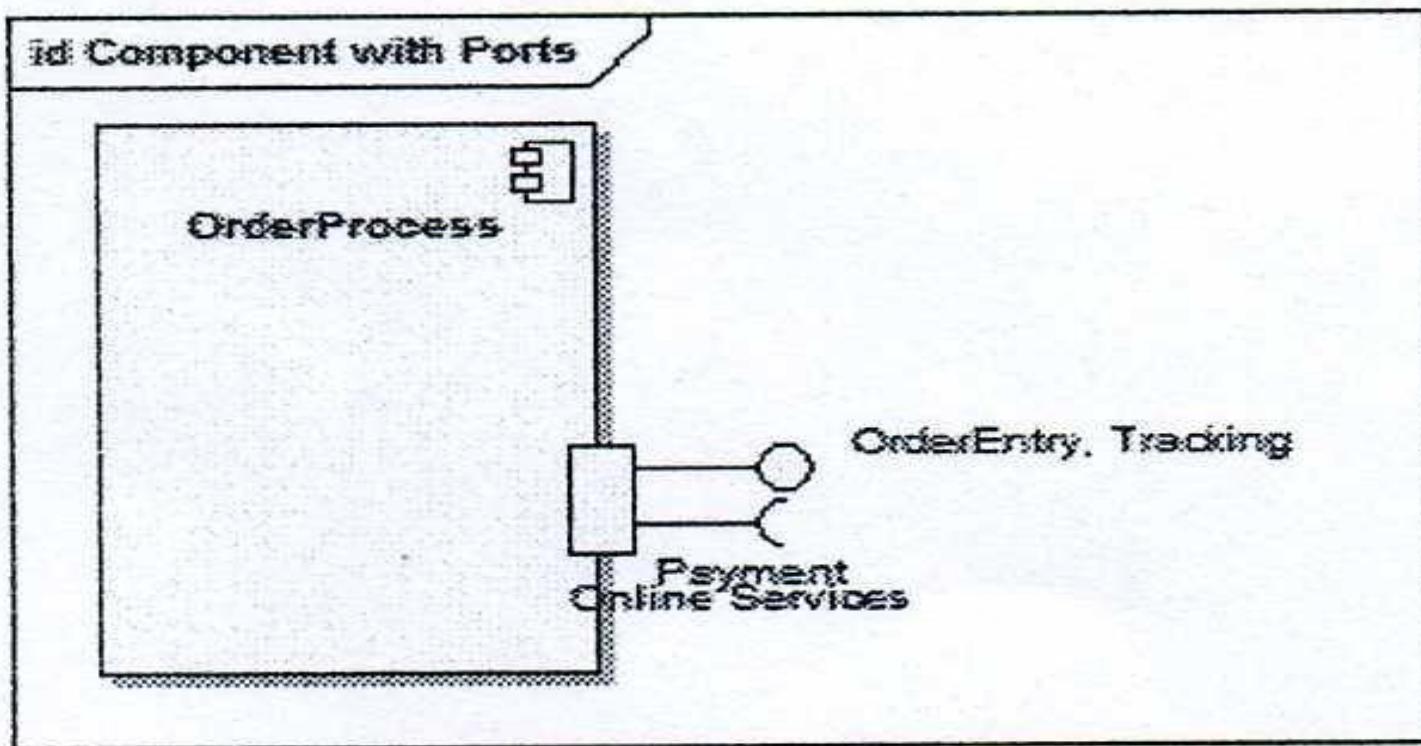


# Komponens kapukkal

- Egy komponens specifikációját kiegészíthetjük olyan kapukkal, ahol a környezet számára nyújtott szolgáltatásokat és a környezettől elvárt szolgáltatásokat egyaránt specifikálhatjuk.
- A következő ábrán szereplő kapuk közül az OrderEntry, Tracking a szolgáltatott interfészt, a payment az elvárt interfészt jeleníti meg az OrderProcess komponens számára, amelynek egy Online Service kapuja van.

# Példa

payment.



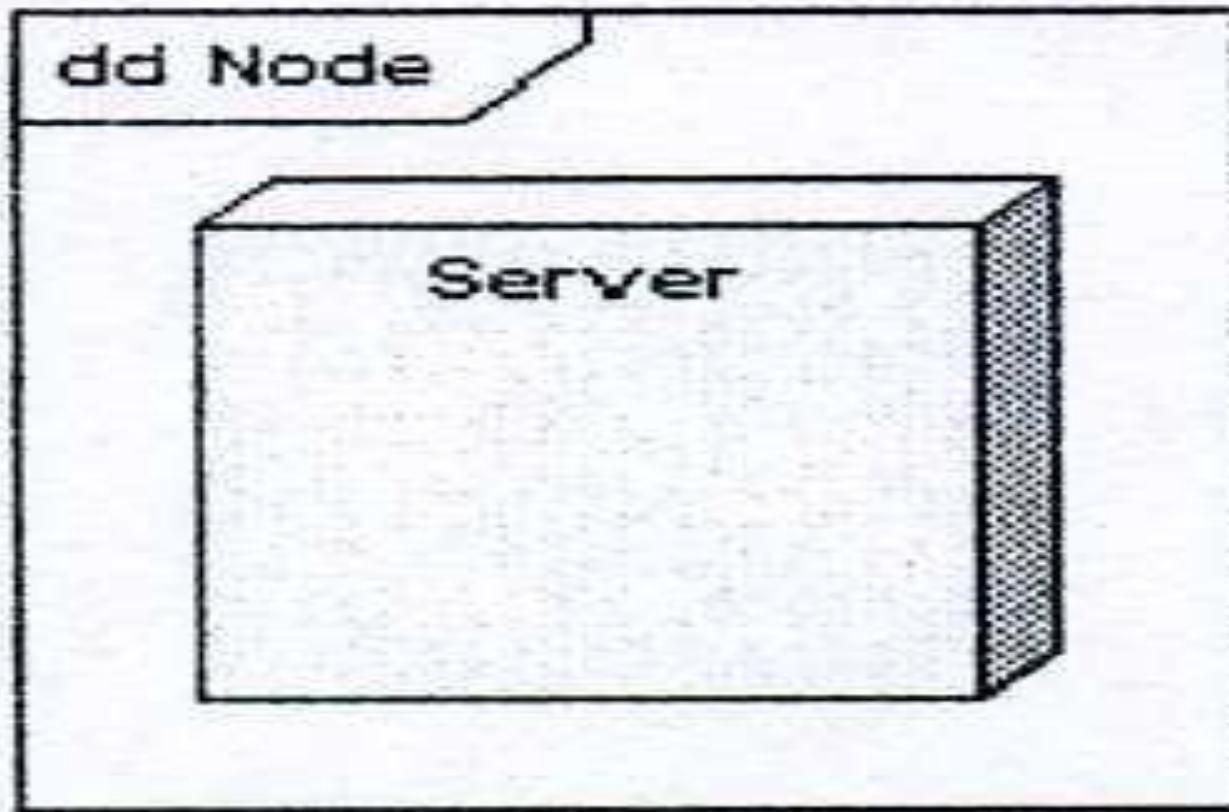
# UML2: telepítési diagramok

- Egy telepítési diagram (deployment diagram) egy rendszer futásidejű architektúráját modellezi.
- A diagram a hardverelemek (nodes) konfigurációját és a szoftverelemek leképezését a hardverre mutatja be.
- A telepítési diagram elemeit a következő ábrákon példák segítségével mutatjuk be.

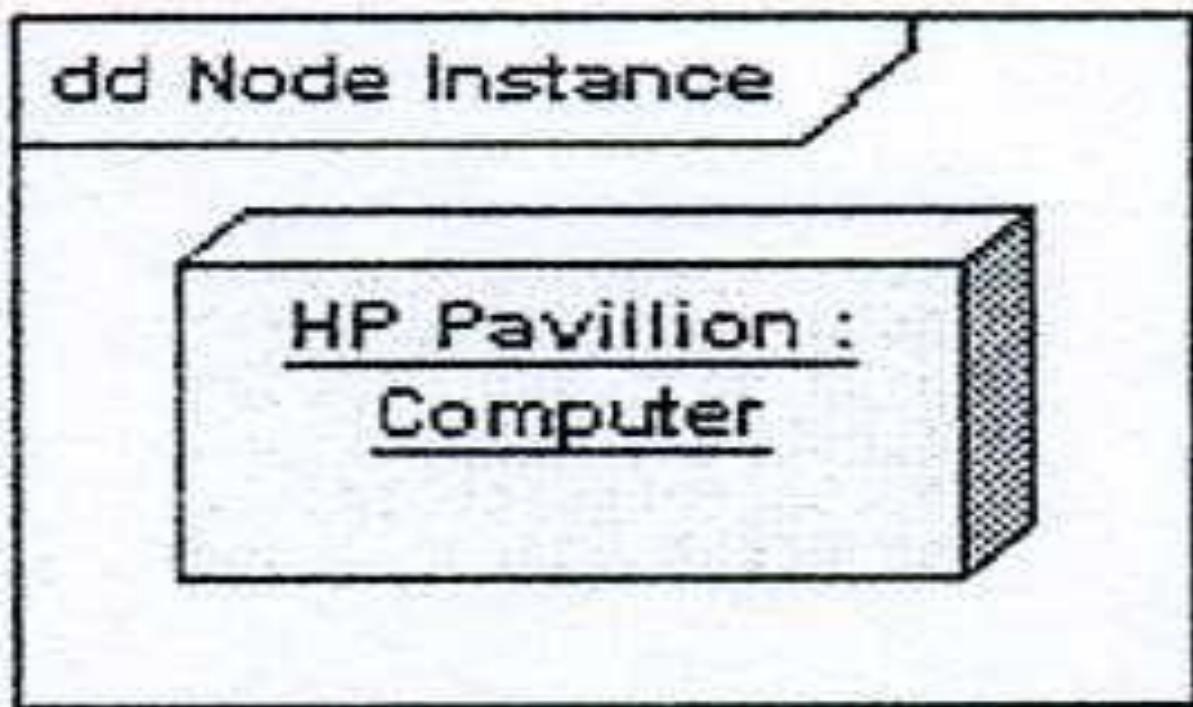
# Nodes

- Egy node egy hardver- vagy egy szoftverelement reprezentálhat, amelyet egy háromdimenziós dobozzal reprezentálhatunk.

# Egy node grafikus ábrázolása



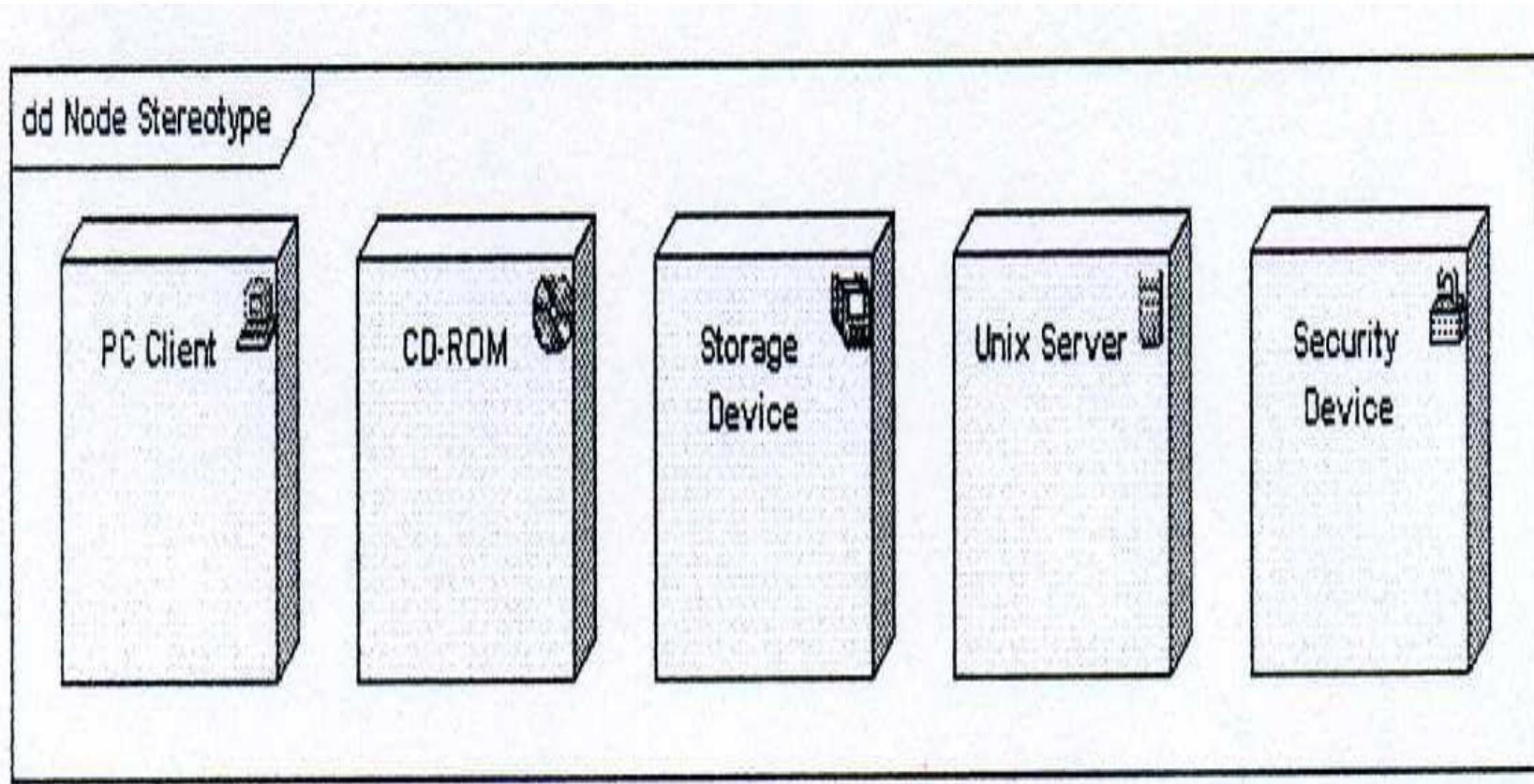
# Egy node példány ábrázolása



# Node sztereotípiák

- Számos standard sztereotípiát definiáltak egy node azonosítására: <server>, <storage>, <cdrom>, <computer>, <pc>, <unix server>, <pc server>, <pc client>, stb.
- Néhányat ezek közül bemutatunk a következő ábrán.

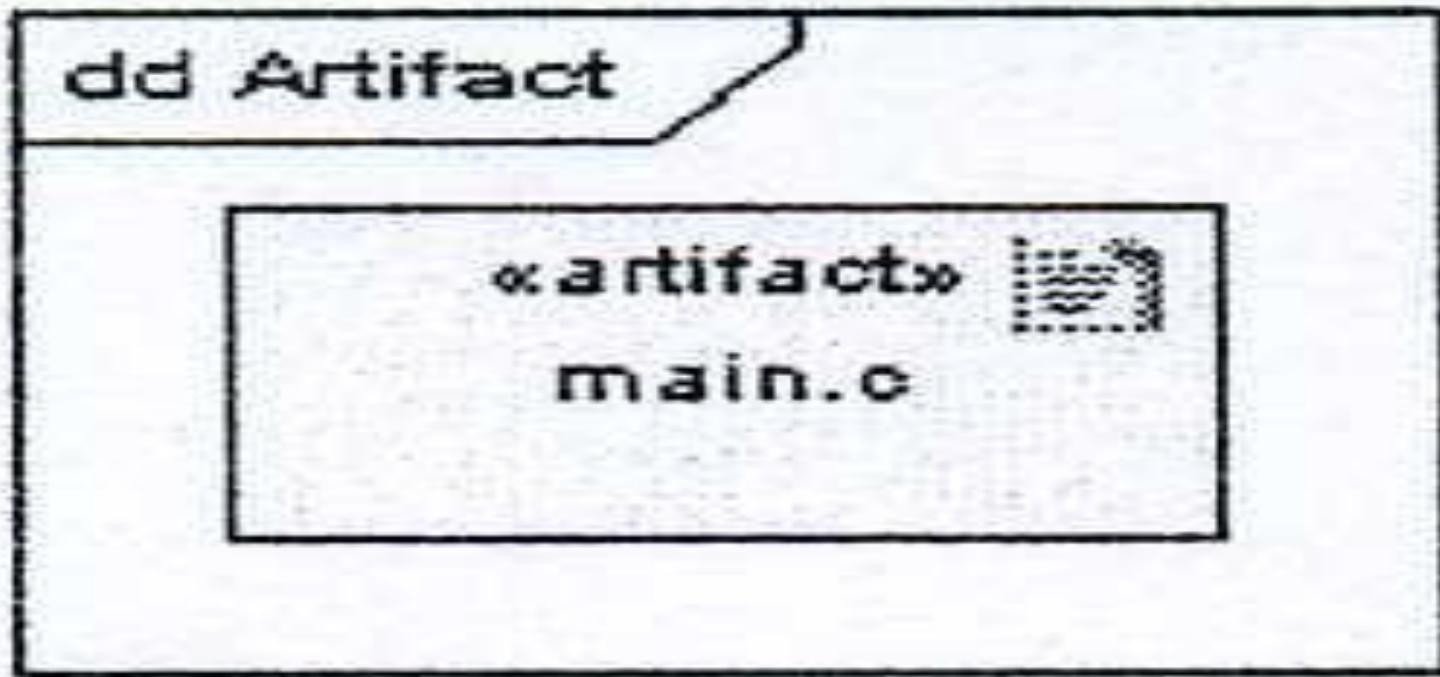
# Node sztereotípiák ábrázolása



# Termék (artifact)

- Egy termék egy szoftverfejlesztő folyamat eredményeként áll elő, amely
  - modelleket (használati modellek, tervezési modellek, stb.),
  - teszt eredményeket, prototípusokat és felhasználói kézikönyveket, forrás fájlokat, végrehajtható állományokat, stb. tartalmazhat.
- A termék grafikus megjelenítésére példát mutatunk a következő ábrán.

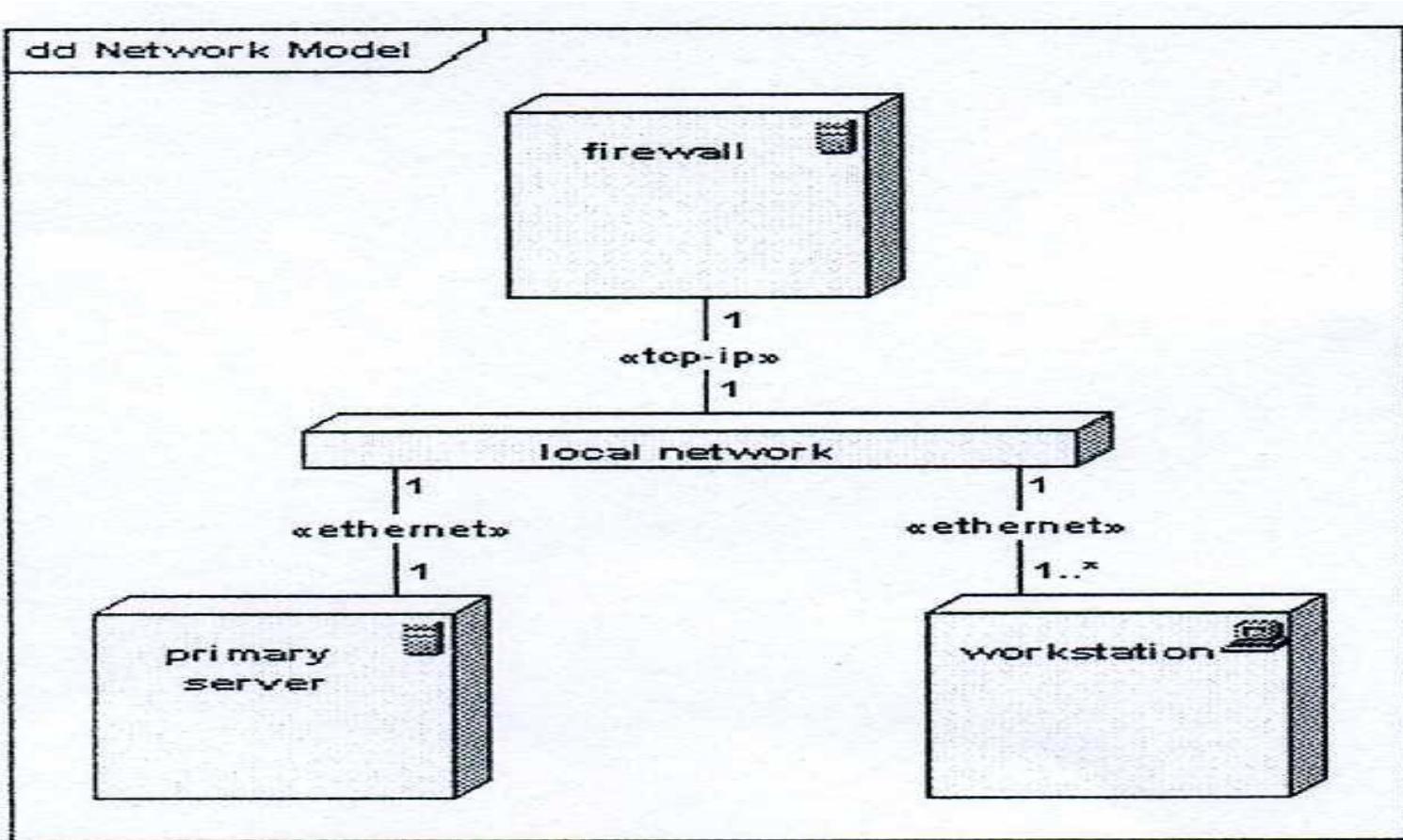
# Termék grafikus megjelenítése



# Telepítési asszociáció

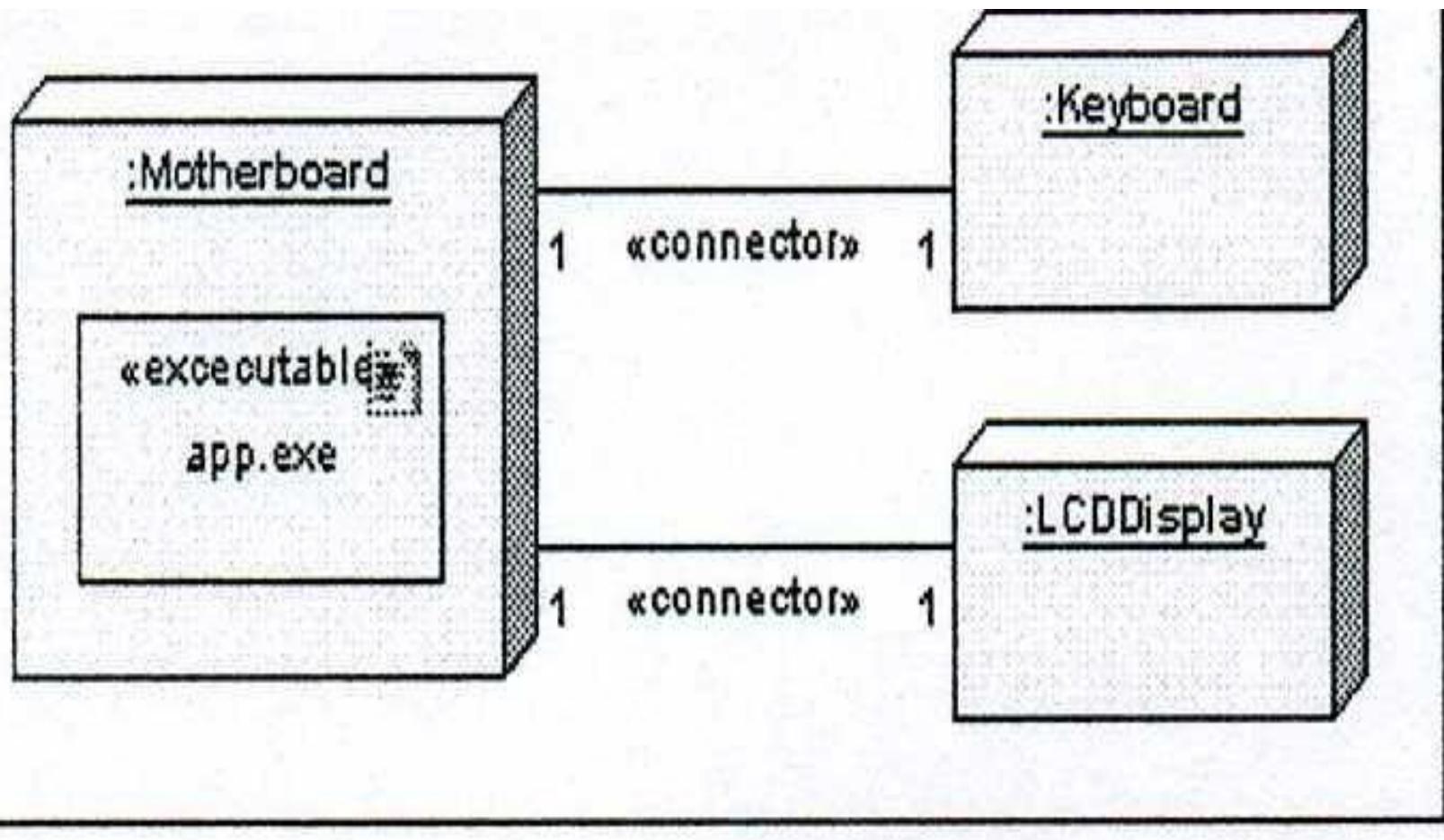
- A telepítési diagramok esetében az asszociációk (associations) a node-k közötti kommunikációs utakat reprezentálják.
- A következő ábrán egy példán mutatjuk be a telepítési asszociációt.

# Példa telepítési asszociációra



# Node mint konténer megjelenítése

dd Embedded Model



# Második csoport

Az UML2 viselkedési diagramjai

# UML2: viselkedést leíró diagramok

- Használati eset ( use case) diagramok
- Aktivitási (activity) diagramok
- Állapotgép (state machine) diagramok
- Kommunikációs (communication) diagramok
- Szekvencia (sequence) diagramok
- Időzítési (timing) diagramok
- Kölcsönhatás (Interaction overview) diagramok

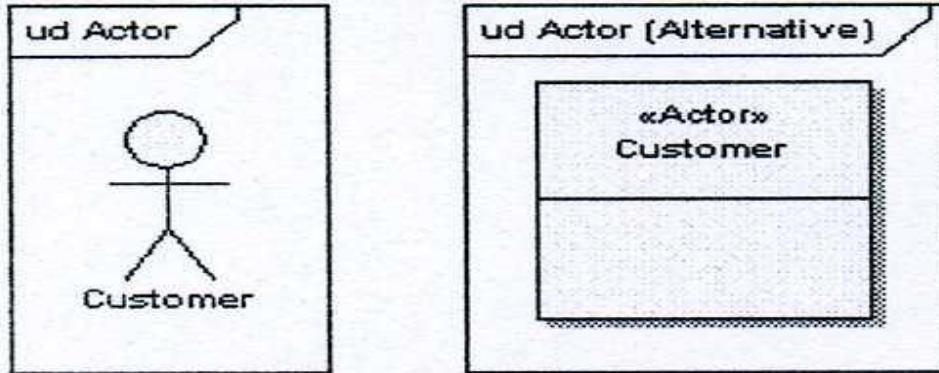
# UML2: használati eset diagramok

- A használati modell a rendszerrel kapcsolatos elvárásainkat a lehető legmagasabb absztrakciós szinten fogalmazza meg.
- A modell összetevői, elemei:
  - szereplők (actors), akik más szereplőket általnosíthatnak,
  - használati esetek, amelyek magas szinten írják le a külső megfigyelők számára a rendszer viselkedését,
  - használati eset definíció
    - név és leírás
    - követelmények
    - forgatókönyv
    - forgatókönyv diagramok
    - Egyéb információk.

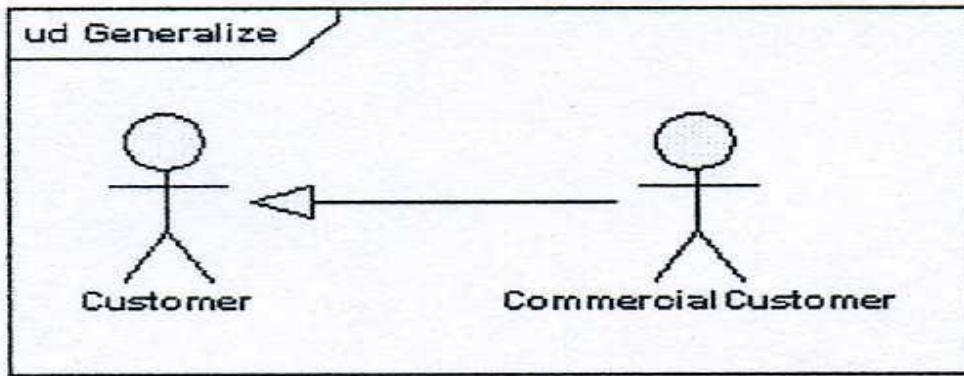
# A modell összetevői 2.

- További elemek:
  - Hivatkozott használati eset (including use cases)
  - Használati eset kiterjesztése (extending use cases)
  - Kiterjesztési pontok (extension points)
  - Rendszer határa (system boundary)

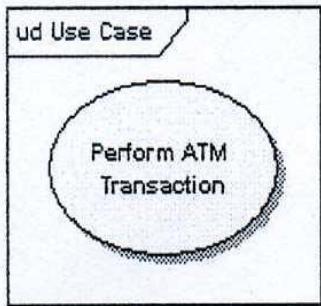
# Az elemek grafikus ábrázolása



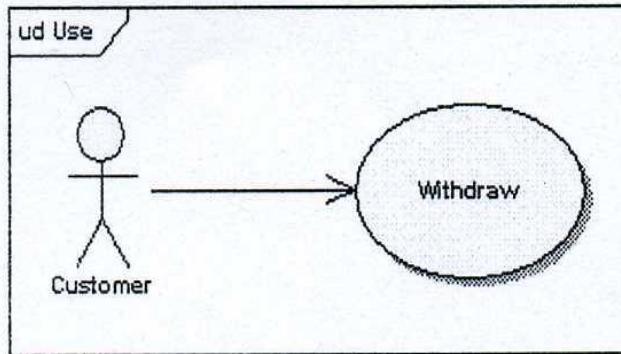
Actors can generalize other actors as detailed in the following diagram:



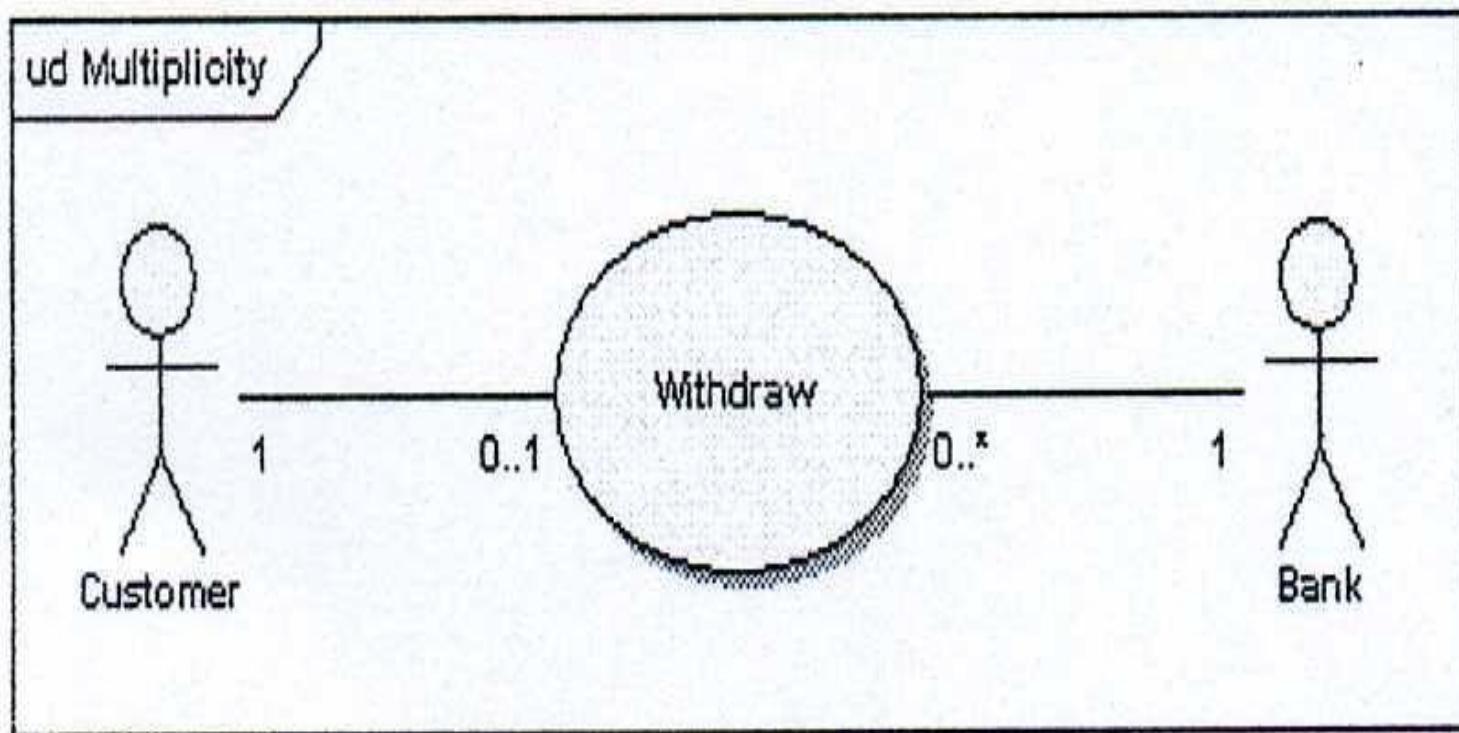
# Használati esetek jelölése



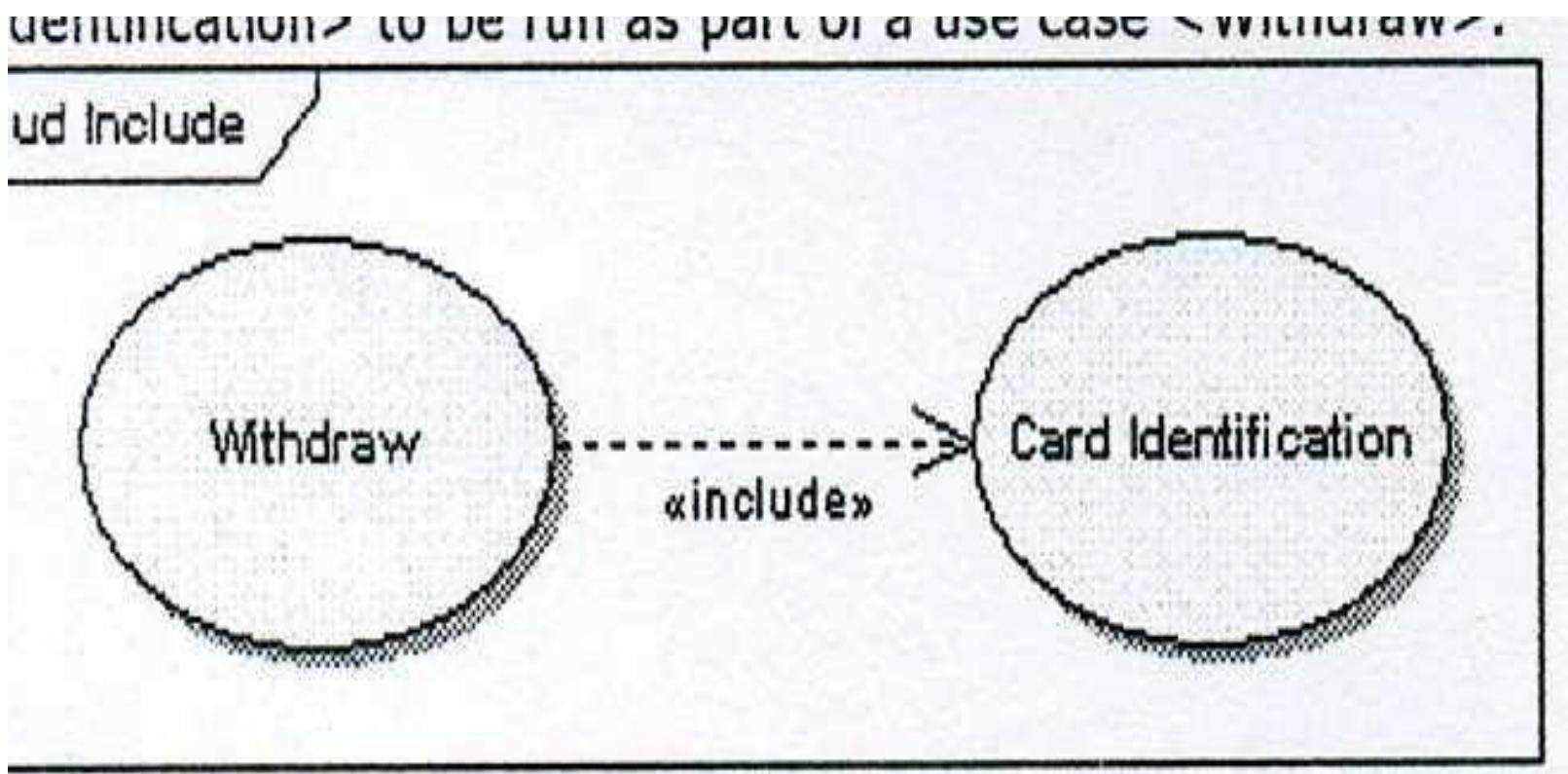
The notation for using a use case is a connecting line with an optional arrowhead showing the direction of control. The following diagram indicates that the actor "Customer" uses the "Withdraw" use case.



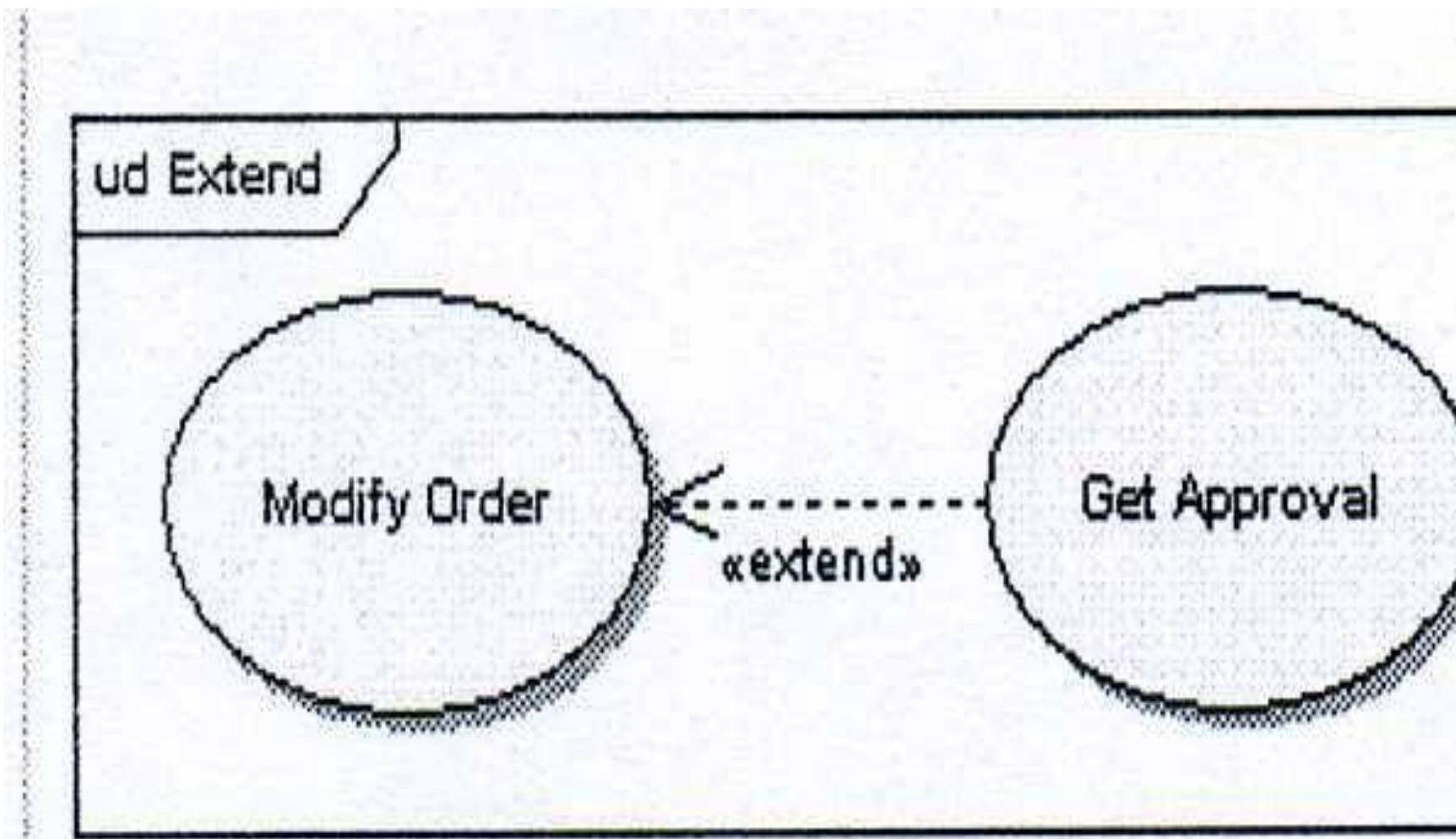
# Több szereplő



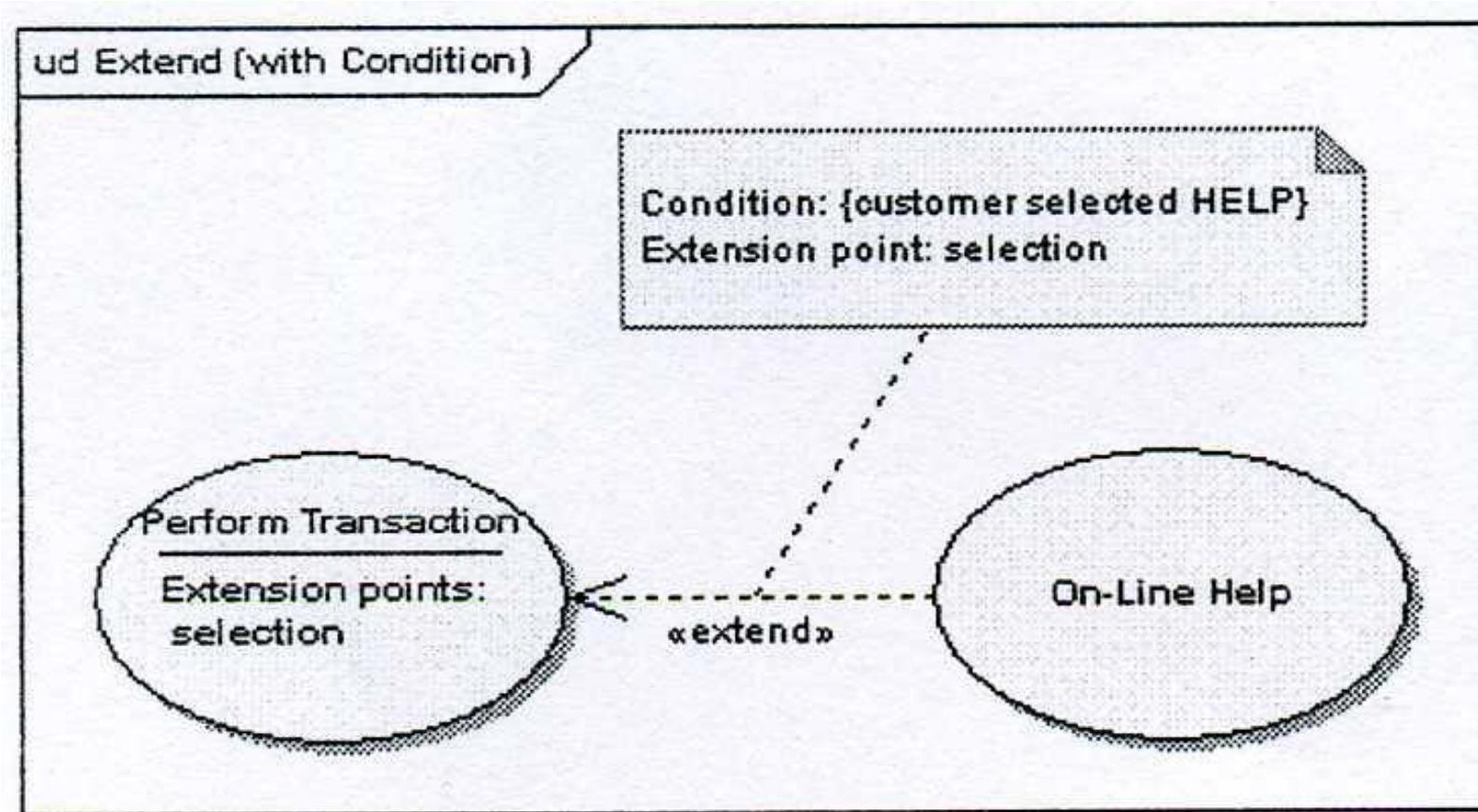
# Tartalmazás (including use case)



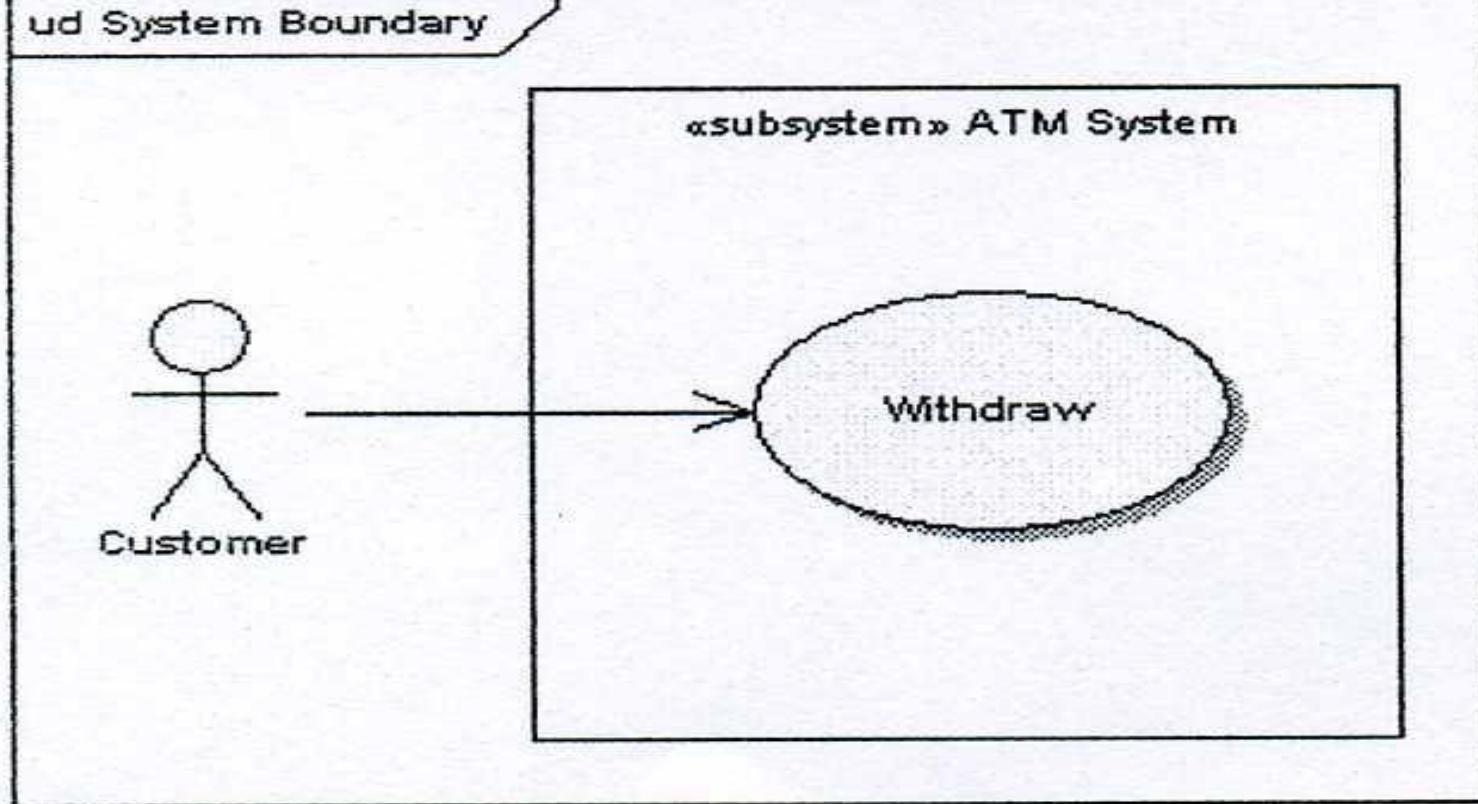
# Kiterjesztés (extending use case)



# Kiterjesztési pontok (extension points)



# Rendszer határa (system boundary)



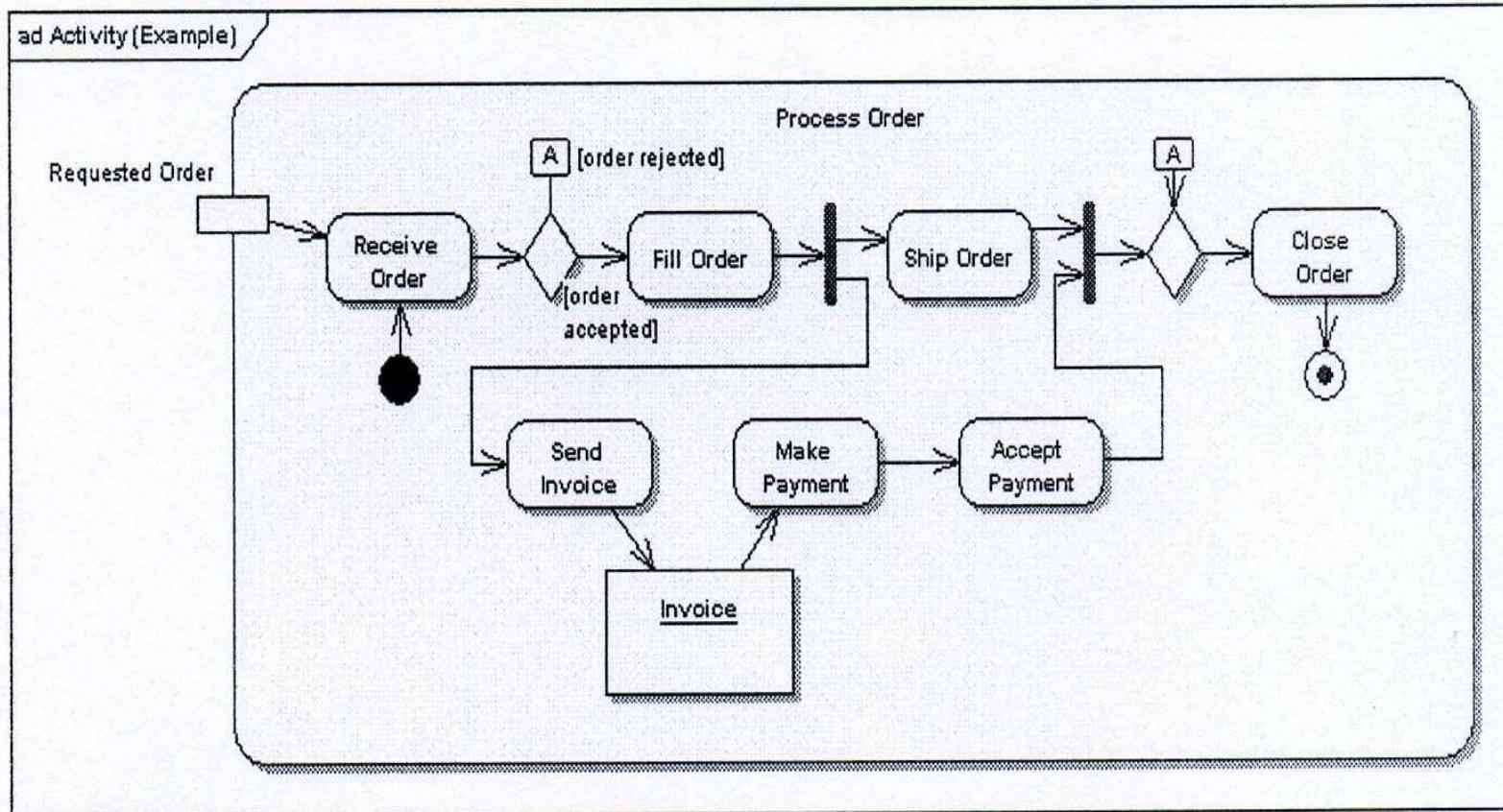
# UML2: aktivitás diagramok

- Aktivitások szekvenciájának megjelenítésére szolgálnak egy munkafolyamat kezdetétől annak befejezéséig.
- A diagramban a különböző döntési útvonalakat valamint a párhuzamos végrehajtást is megjeleníthetjük.
- Az aktivitási diagramokat leggyakrabban üzleti folyamatok modellezésére míg az állapotgép diagramokat egyetlen objektum viselkedésének modellezésére használják.

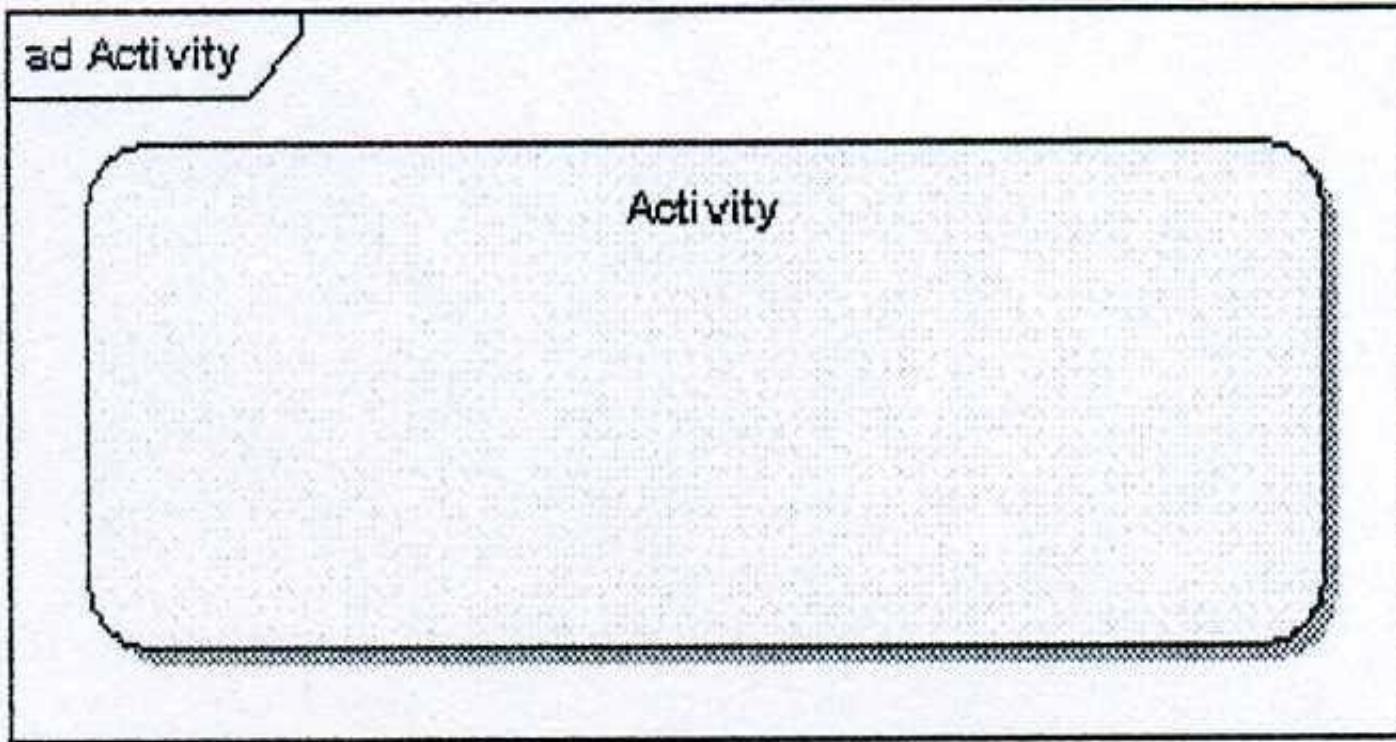
# UML2: aktivitás diagramok

- Elemei:
  - aktivitások (activity),
  - akciók (actions),
  - akciókra vonatkozó megszorítások (action constrains),
  - vezérlési folyam (control flow),
  - kezdő pont (initial node),
  - végpont (final node),
  - objektumok és objektum folyamok (objects and object flows),
  - elágazási és gyűjtő csomópontok (decision and merge nodes),
  - fork és join csomópontok (párhuzamos végrehajtás),
  - bővítési régió (expansion region),
  - kivétel kezelés (exception handlers),
  - megszakítási régió (interruptible activity region),
  - particionálás (partition).

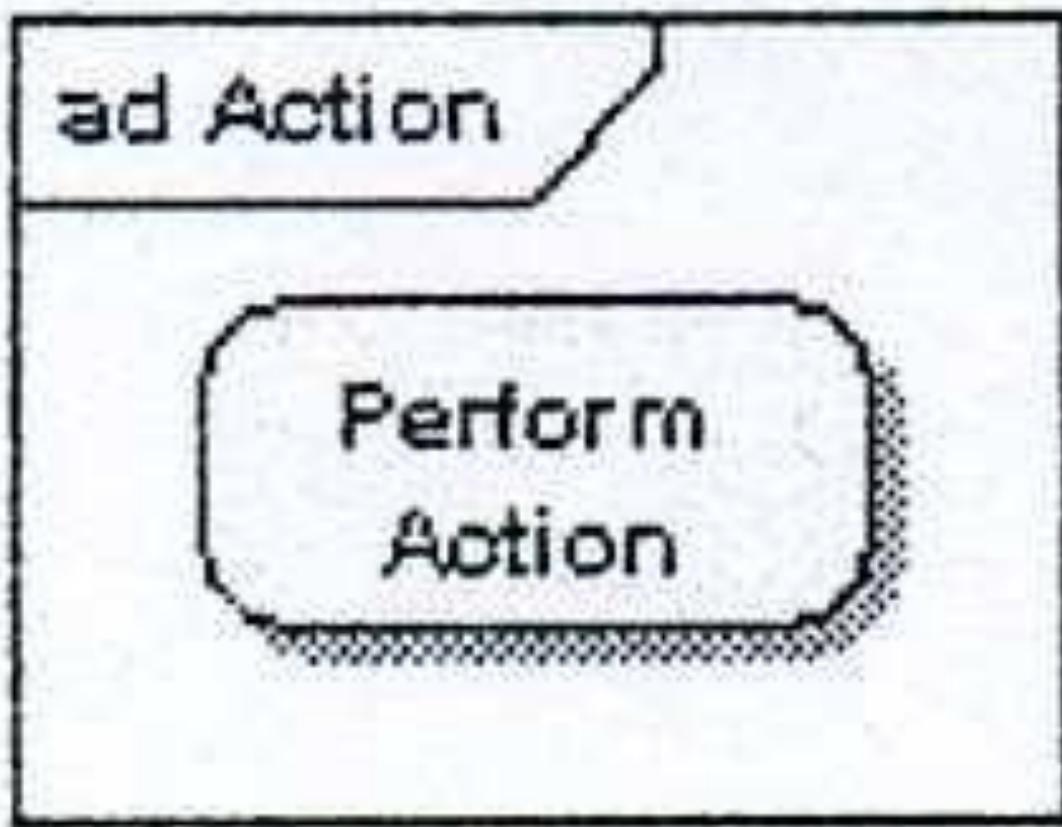
# Egy példa aktivitási diagramra



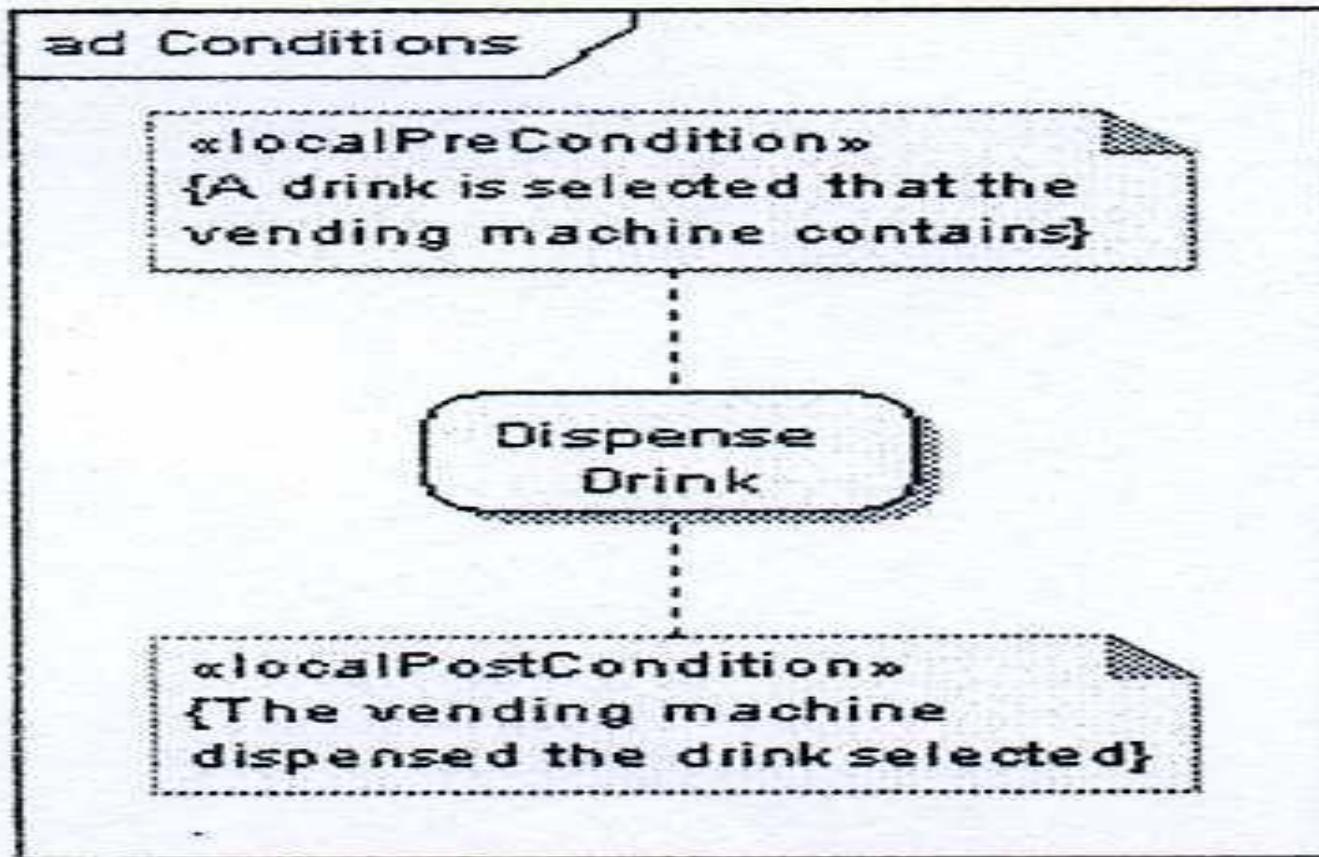
# Aktivitások (activity)



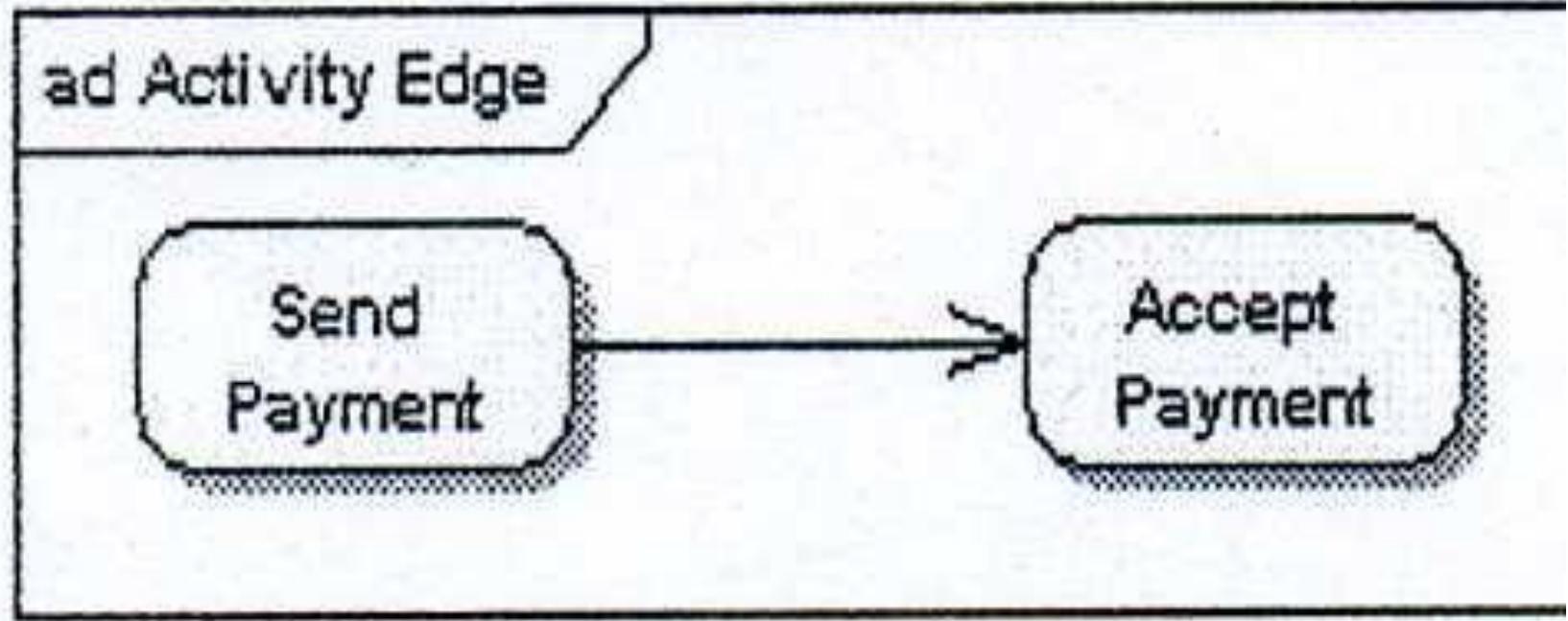
# Akciók (actions)



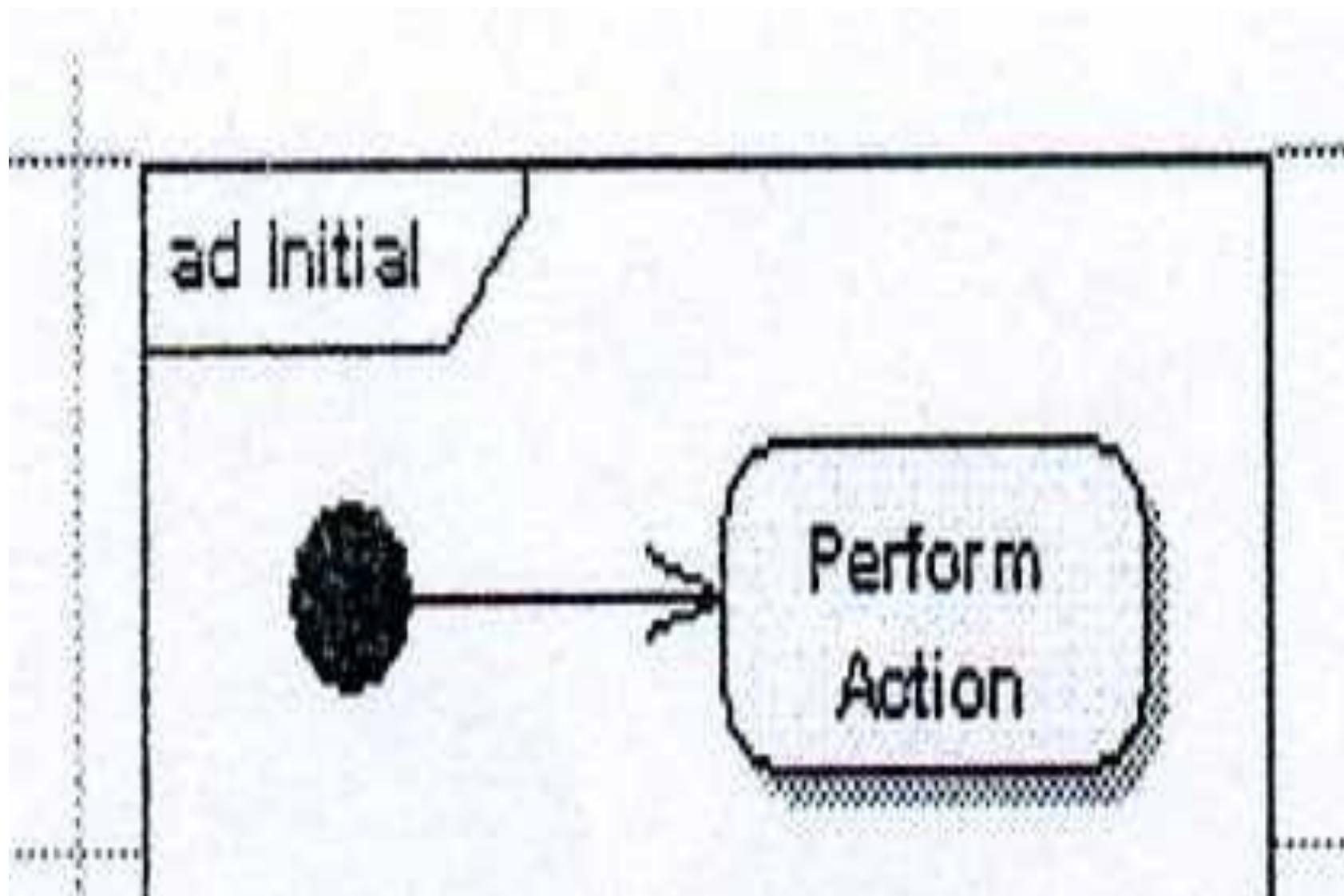
# Akciókra vonatkozó megszorítások (action constraints)



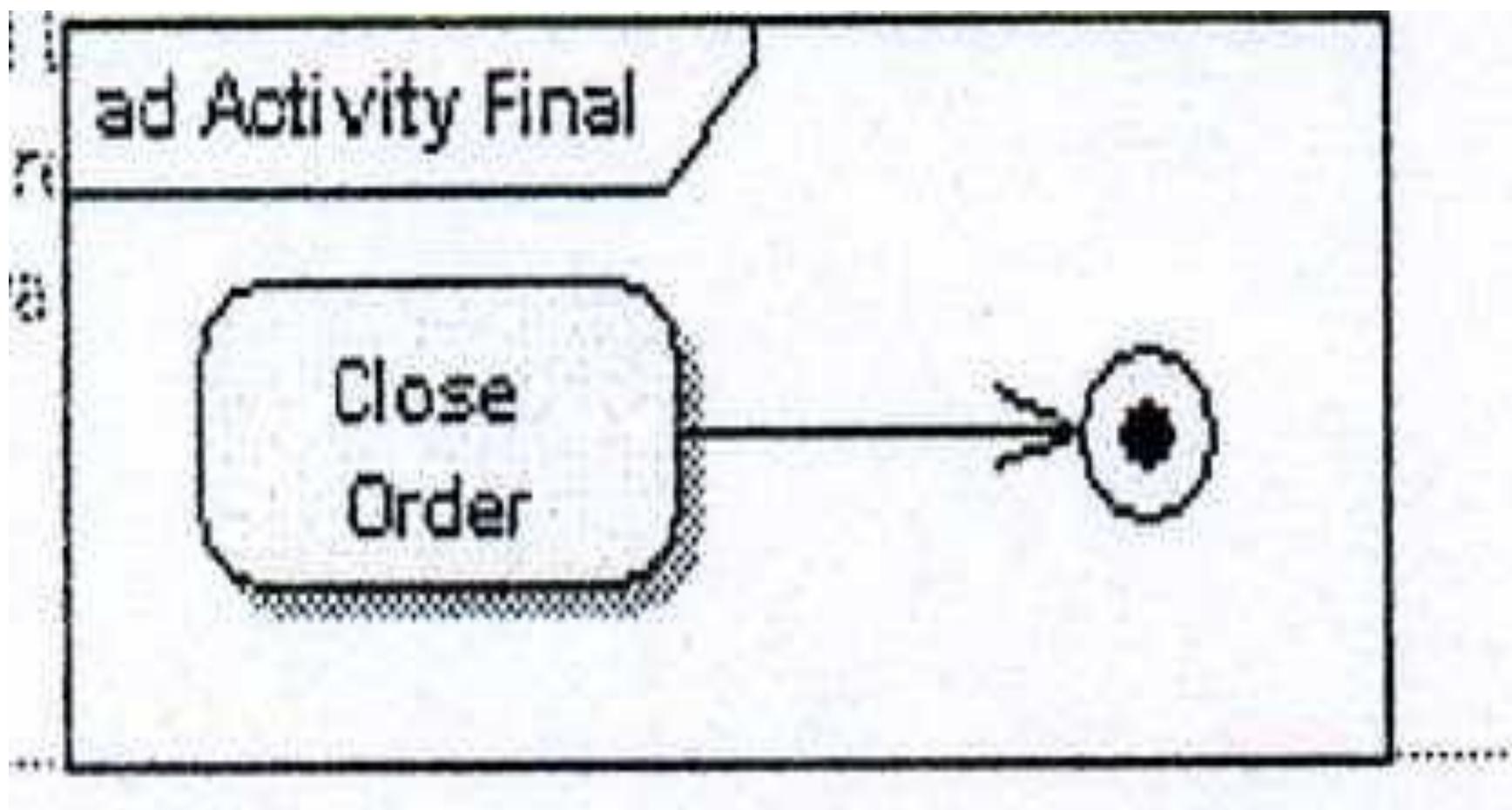
# Vezérlési folyam (control flow)



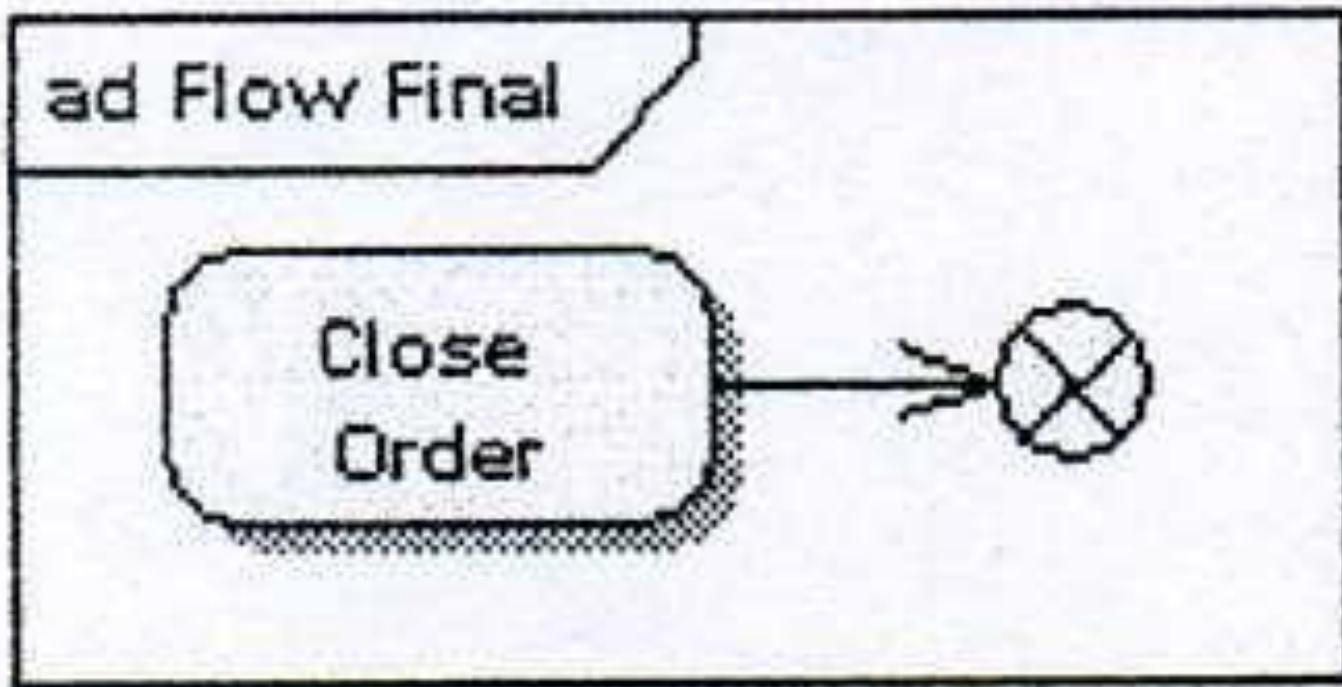
# Kezdő pont (initial node)



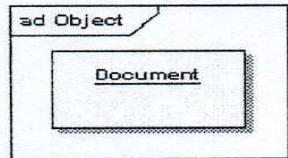
# Végpont (final node)



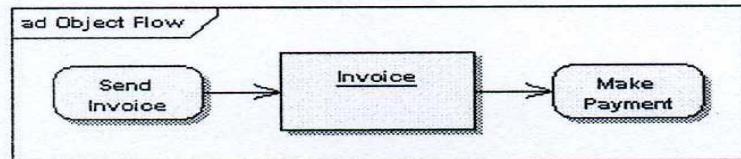
# Folyam végpontja



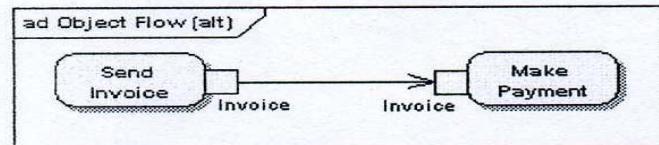
# Objektumok és objektum folyamok (objects and object flows),



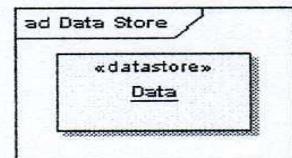
An object flow is shown as a connector with an arrowhead denoting the direction the object is being passed.



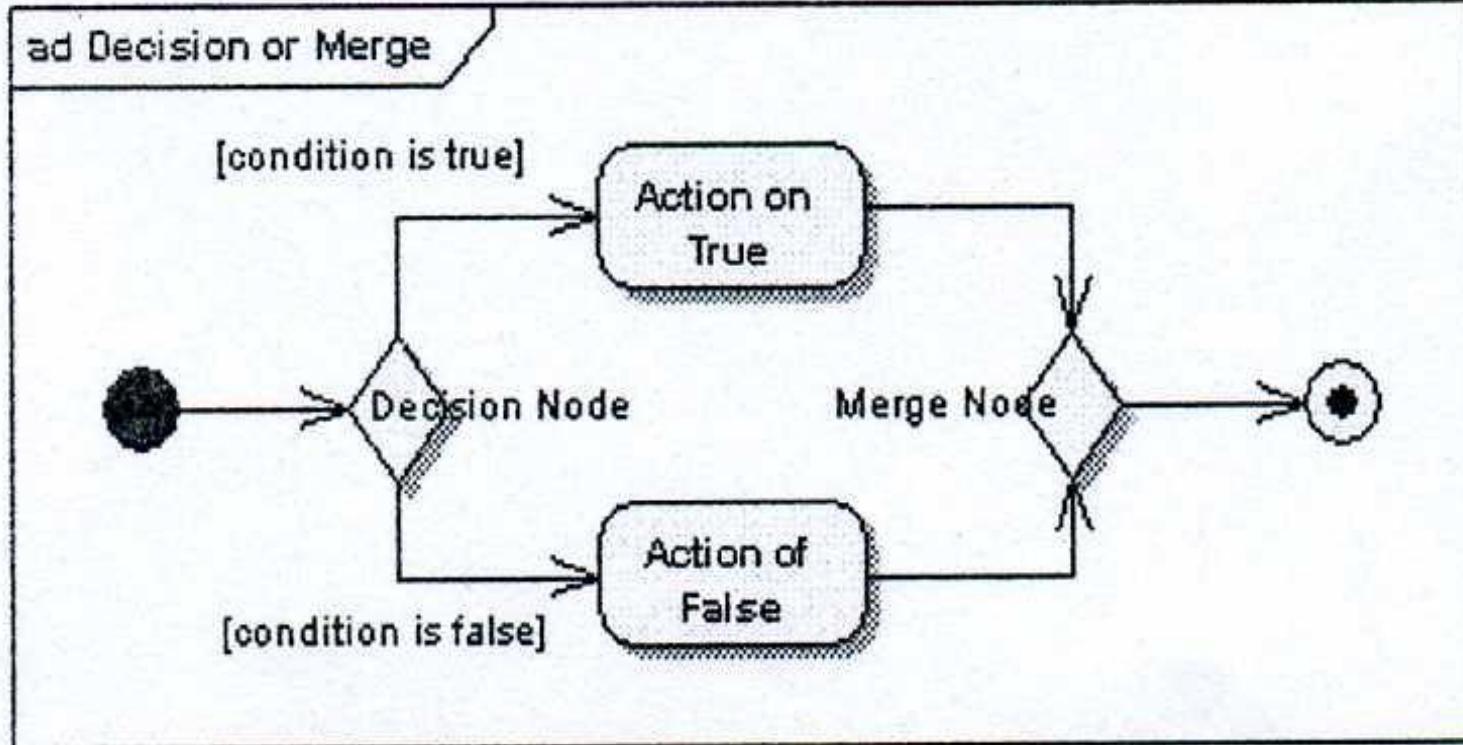
An object flow must have an object on at least one of its ends. A shorthand notation for the above diagram would be to use input and output pins.



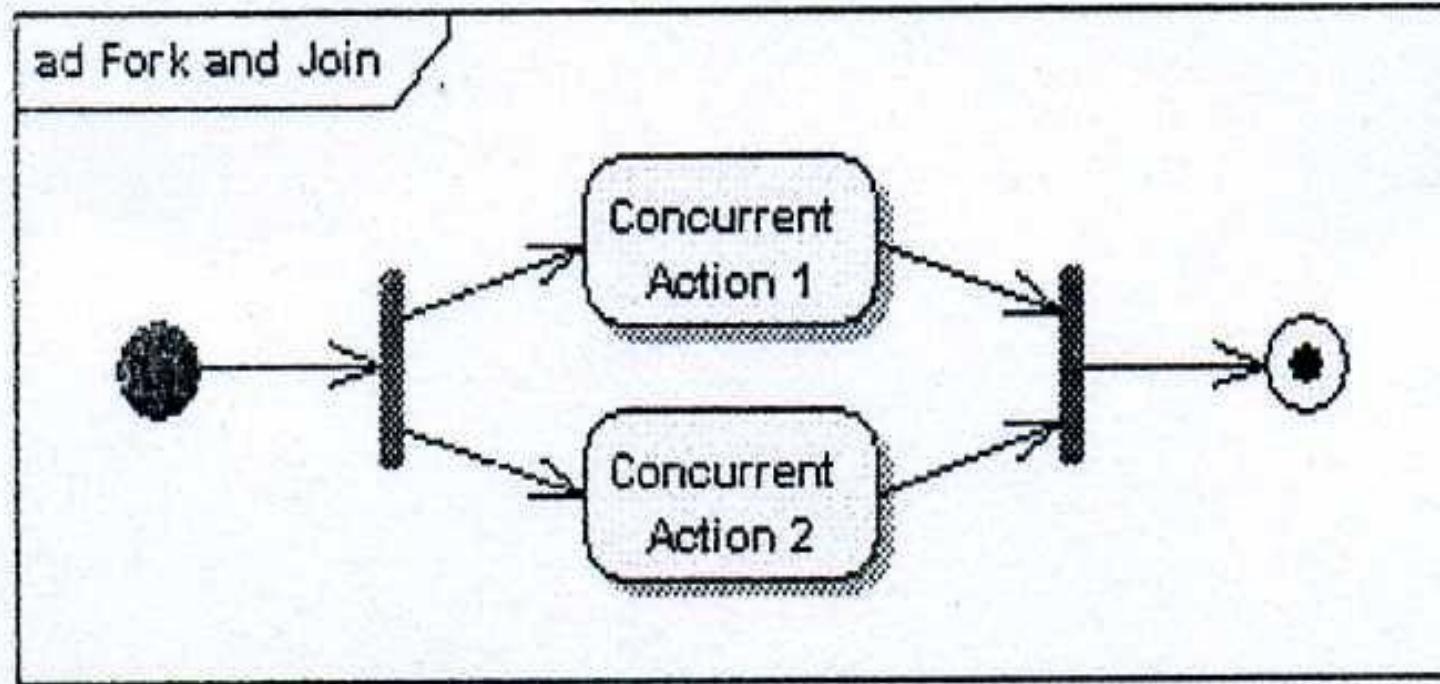
A data store is shown as an object with the «datastore» keyword.



# Elágazási és gyűjtő csomópontok (decision and merge nodes),

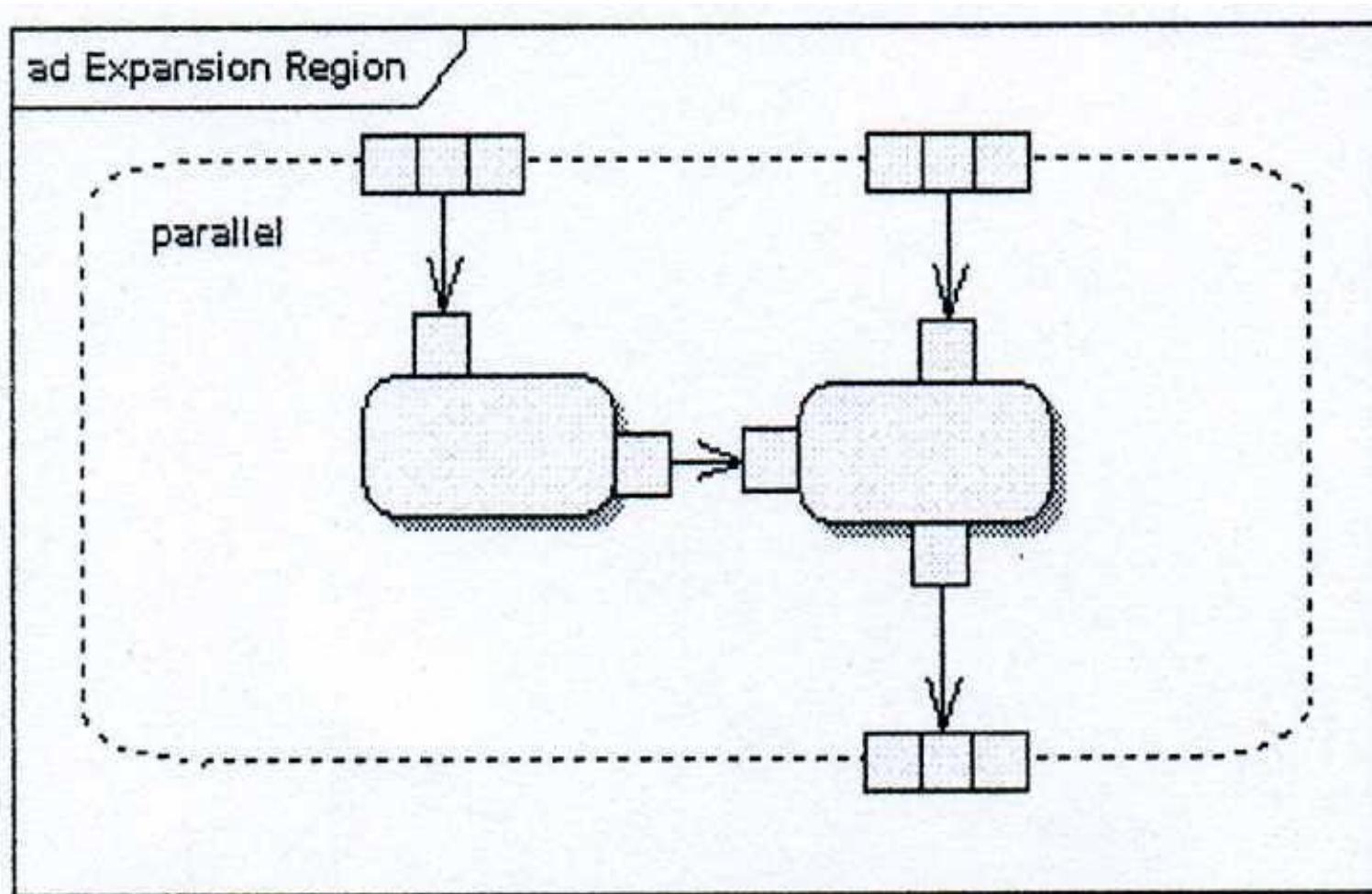


# fork és join csomópontok (párhuzamos végrehajtás),



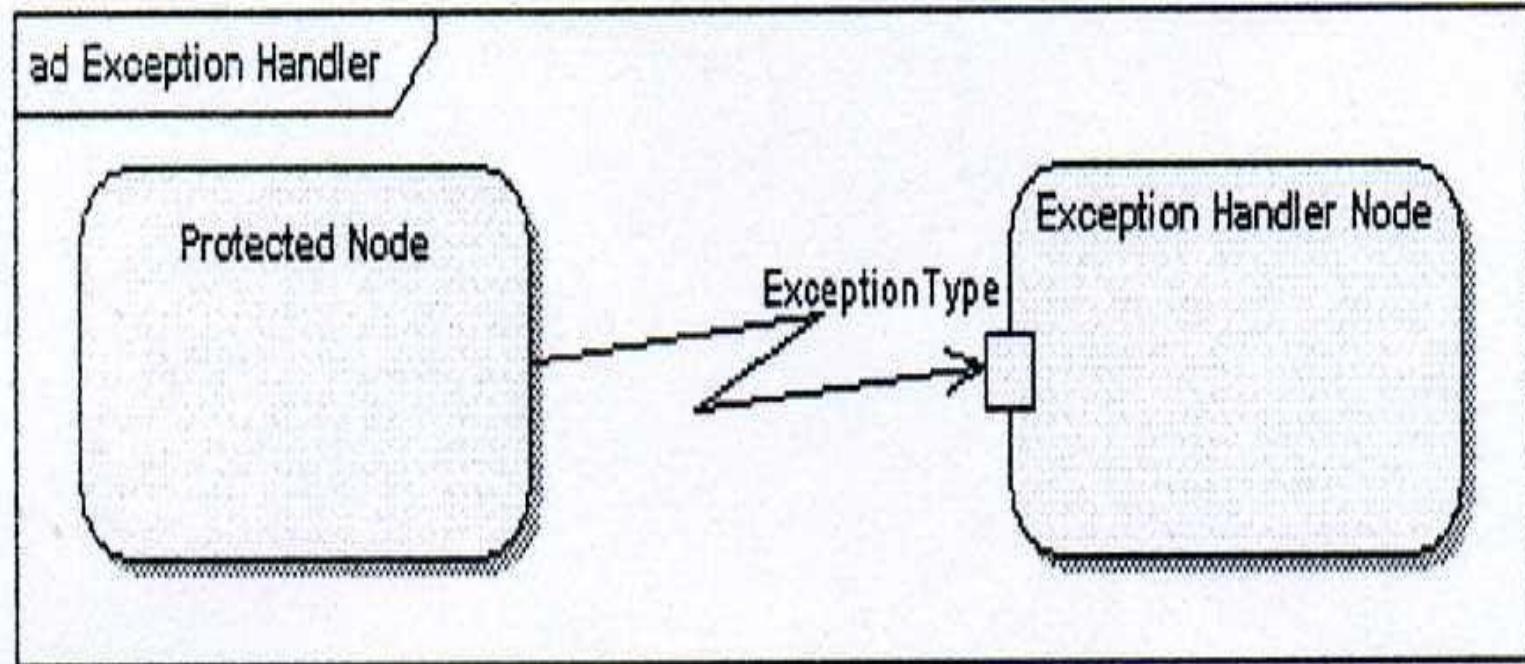
# Bővítési régió (expansion region)

egy olyan strukturált aktivitás, amelyet többször végrehajtunk,  
kulcsszavak: „iterative”, „parallel”, „stream”

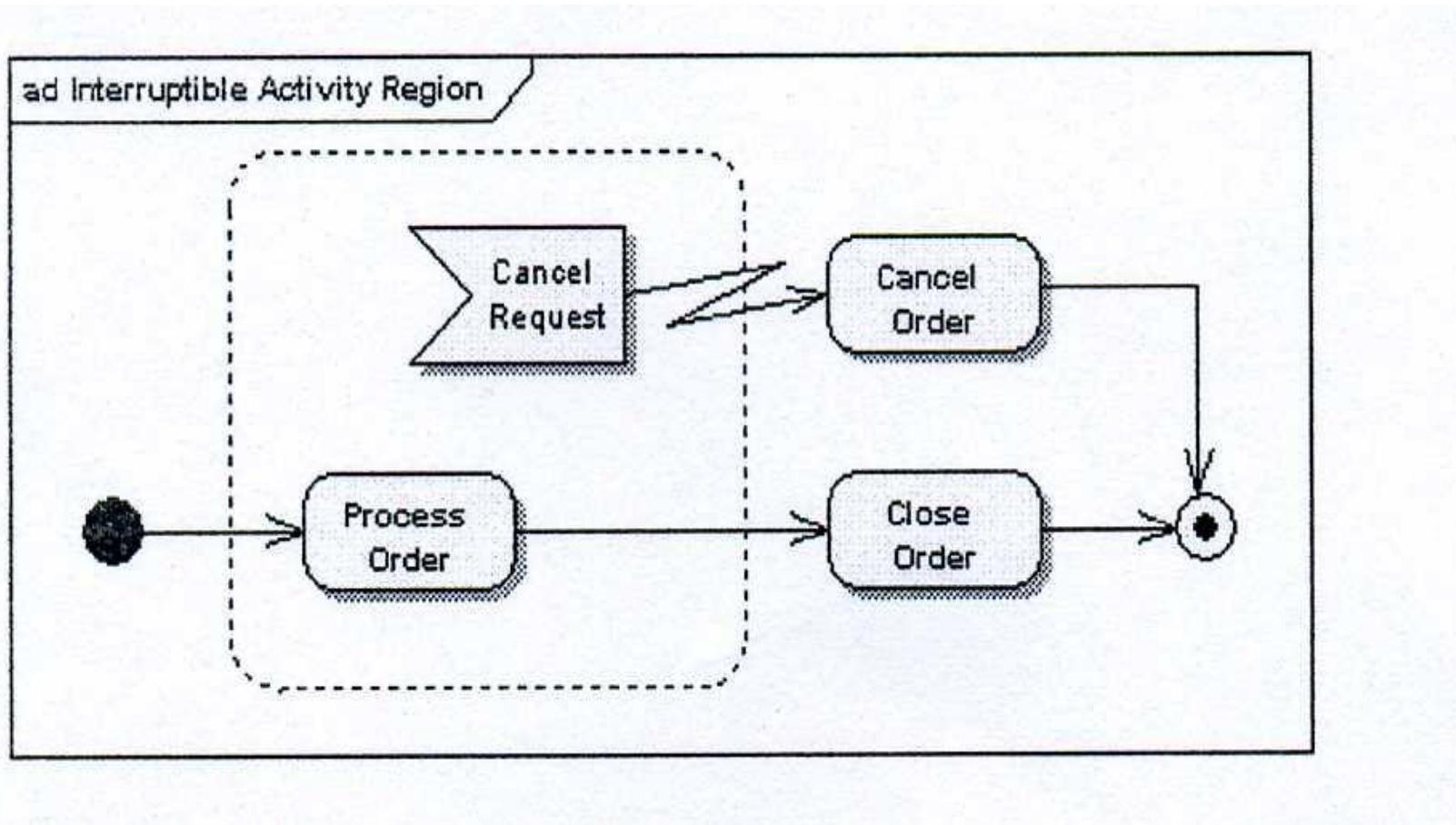


# Kivételek kezelés (exception handlers),

Exception Handlers can be modelled on activity diagrams as in the example below.

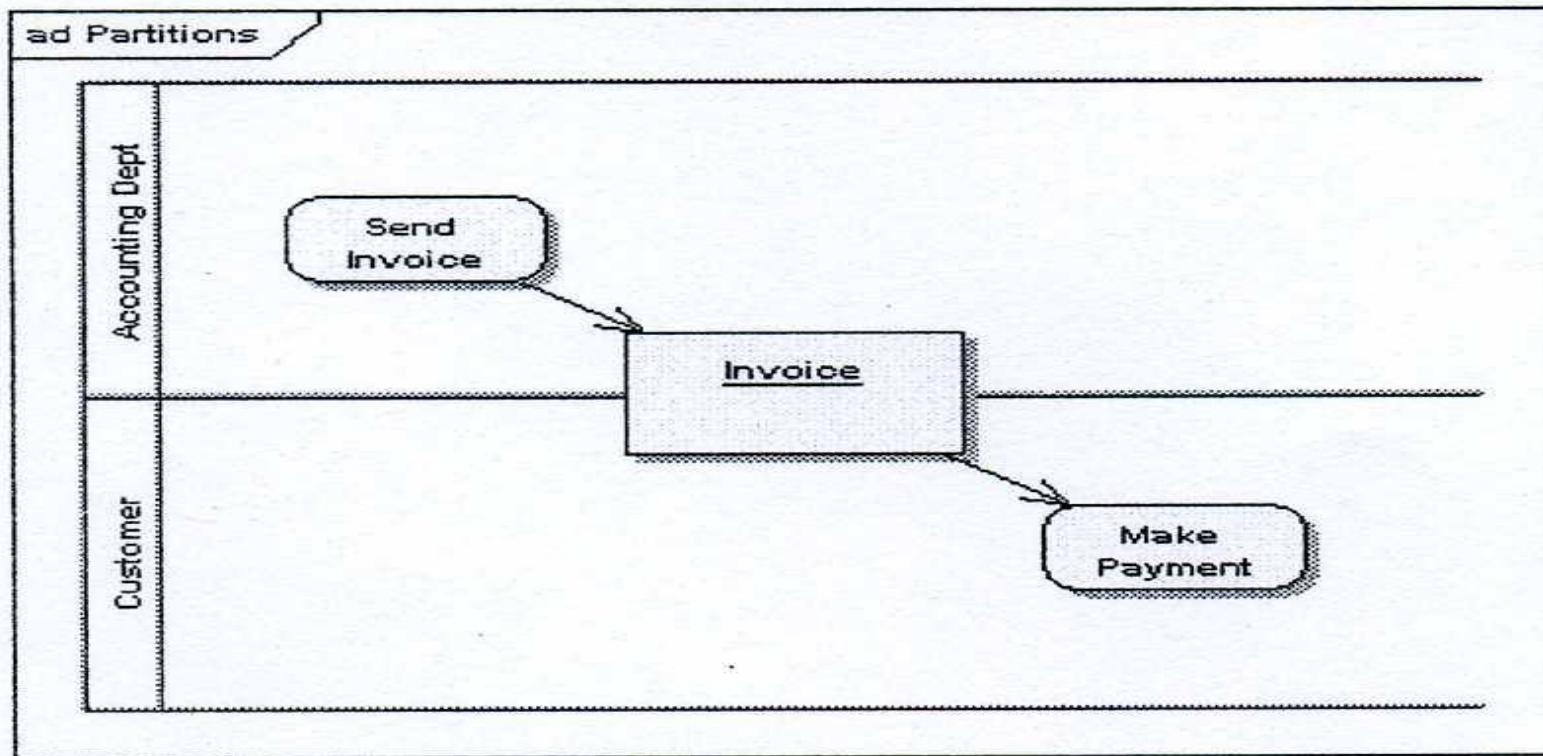


# Megszakítási régió (interruptible activity region)



# Particionálás (partition)

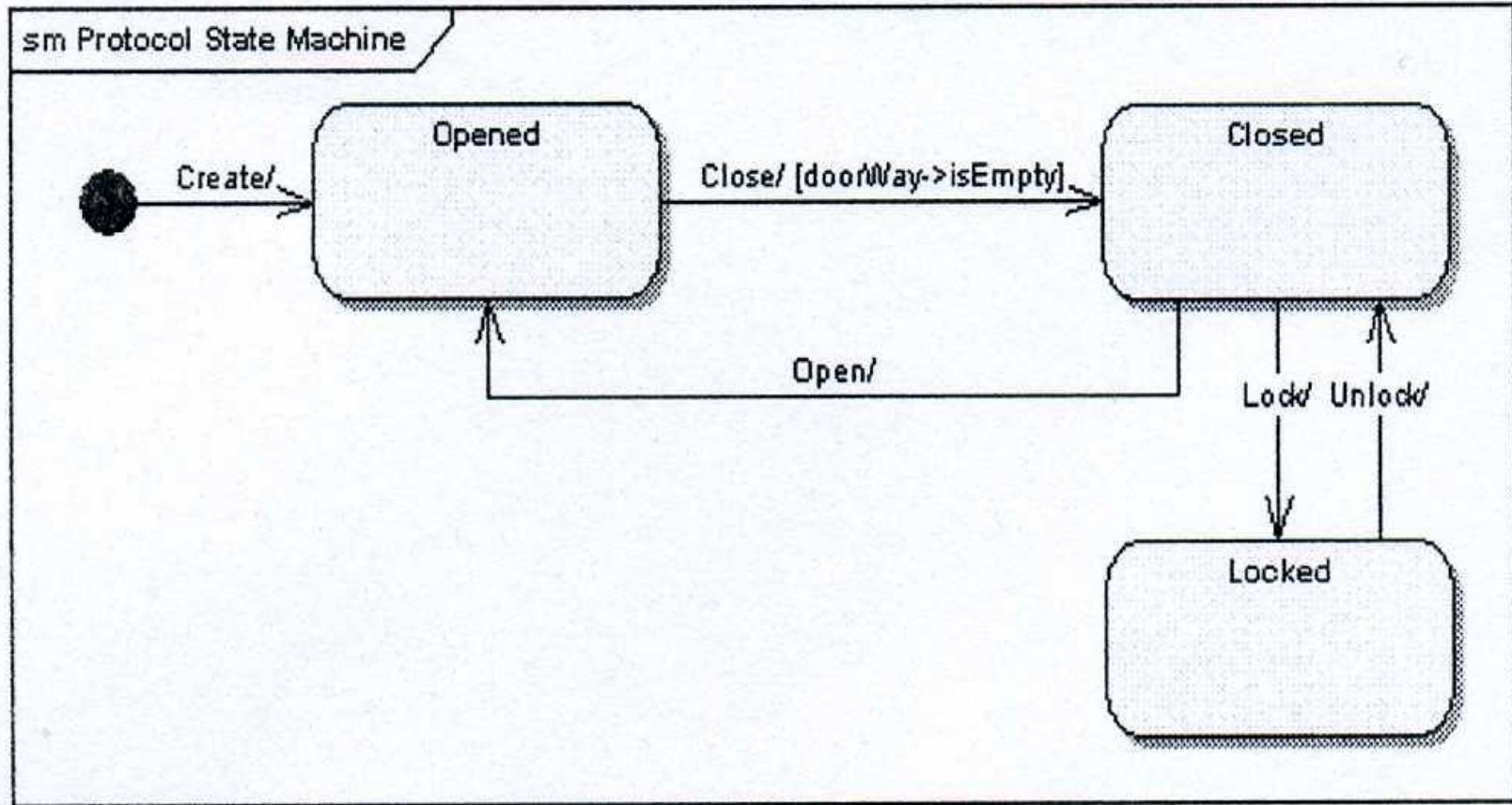
All activity partition is shown as either a horizontal or vertical subdivision, in this it actions within an activity into those performed by the accounting department and



# UML2: állapotgép diagramok

- Az állapotgép diagramok egyetlen objektum viselkedését modellezik.
- Egy kérésre adott válasz generálása közben bekövetkező események szekvenciáját specifikálják.
- A következő ábrán egy ajtó életciklusát leíró állapotgépet mutatunk be.

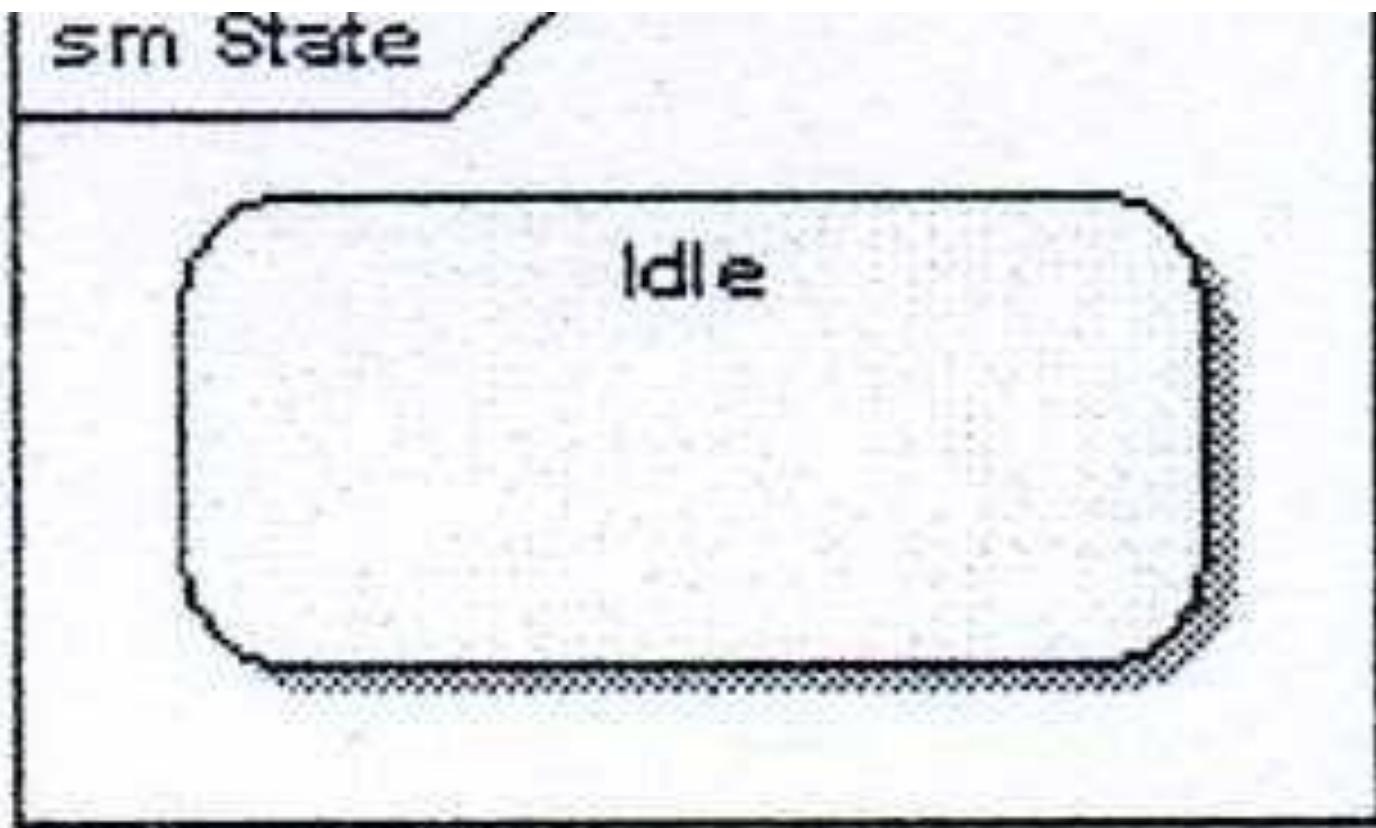
# Egy ajtó életciklusa



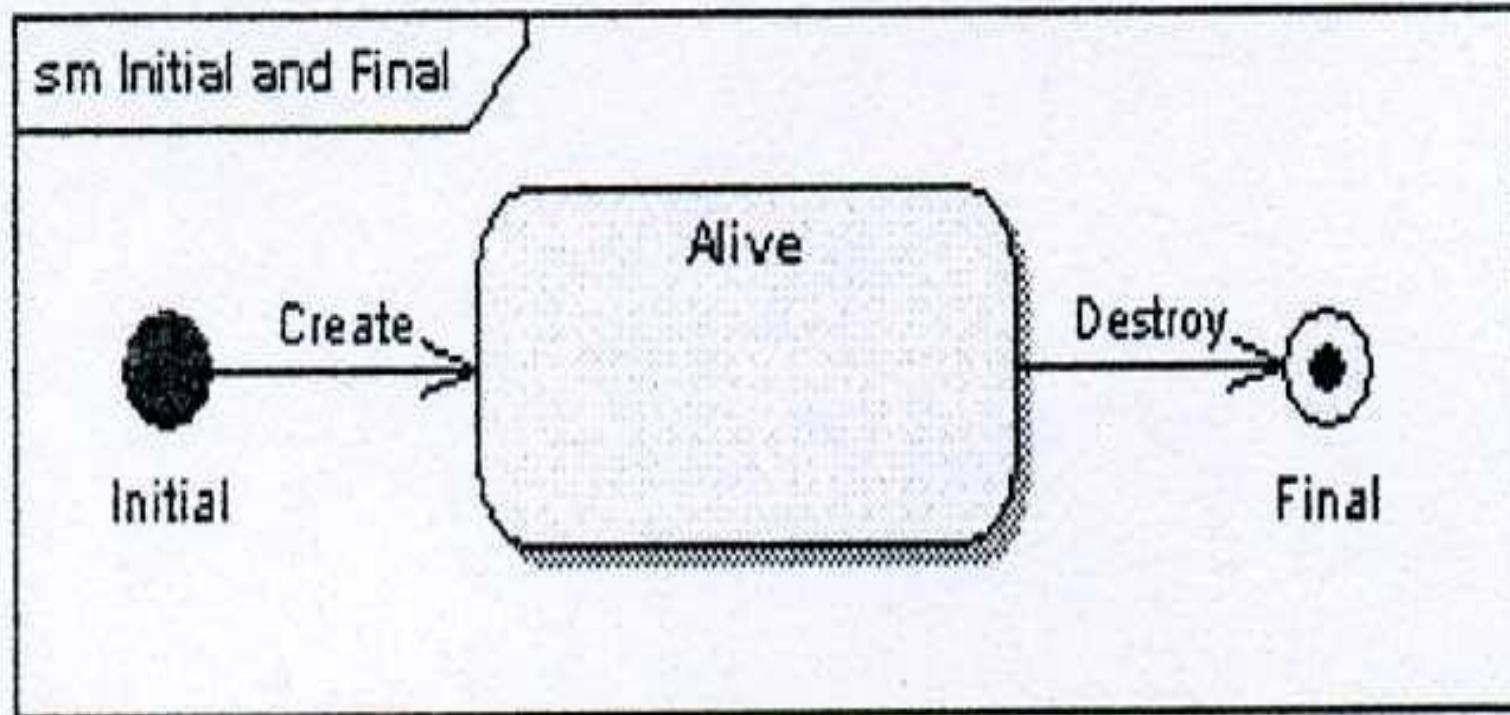
# UML2: állapotgép diagramok

- Elemei:
  - állapotok (states),
  - kezdeti és végállapotok (initial and final states),
  - átmenetek (transitions),
  - állapot akciók (state actions),
  - self-transitions,
  - összetett állapotok (compound states),
  - belépési pont (entry point),
  - magasabb belépési pont (higher entry point),
  - kilépési pont (exit point),
  - választás (choice pseudo-state),
  - csomóponti elágazás (junction pseudo-state),
  - történeti állapot (history state),
  - konkurens régiók (concurrent regions).

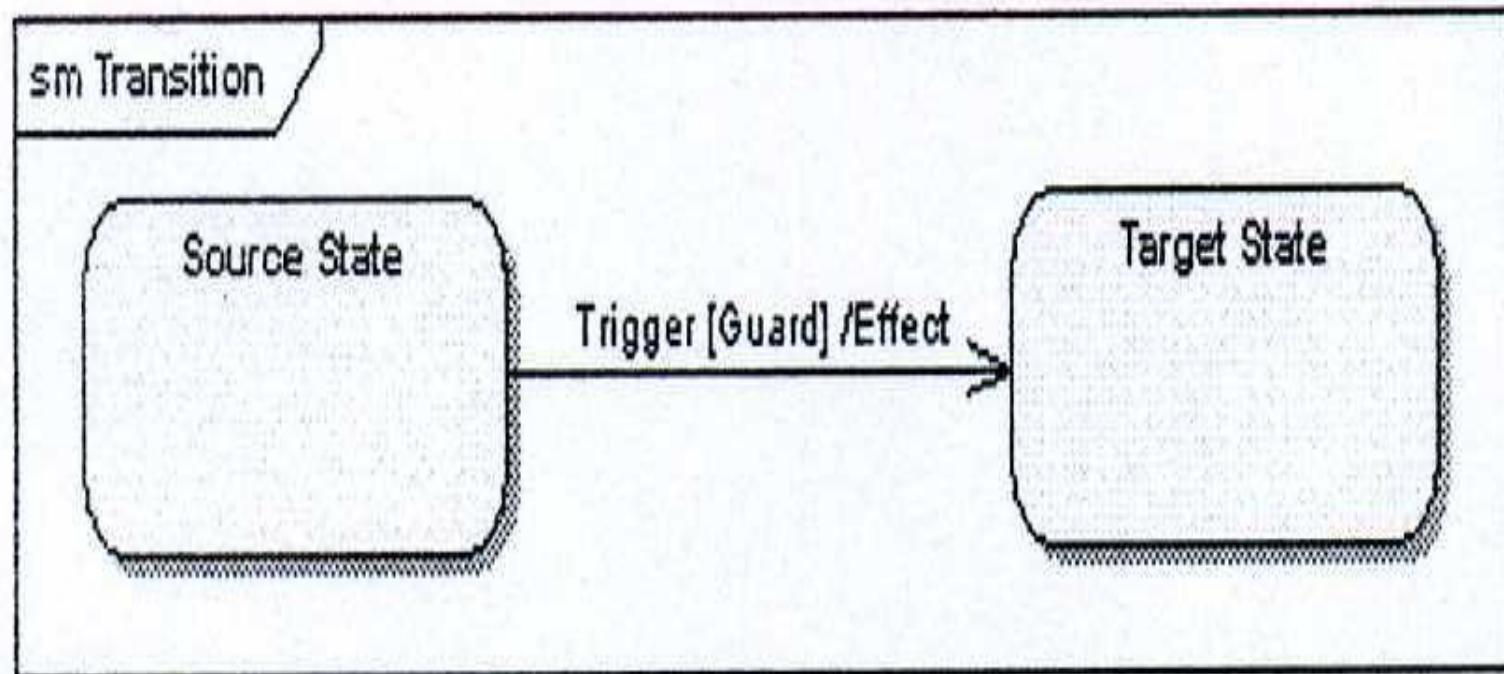
# Állapotok (states)



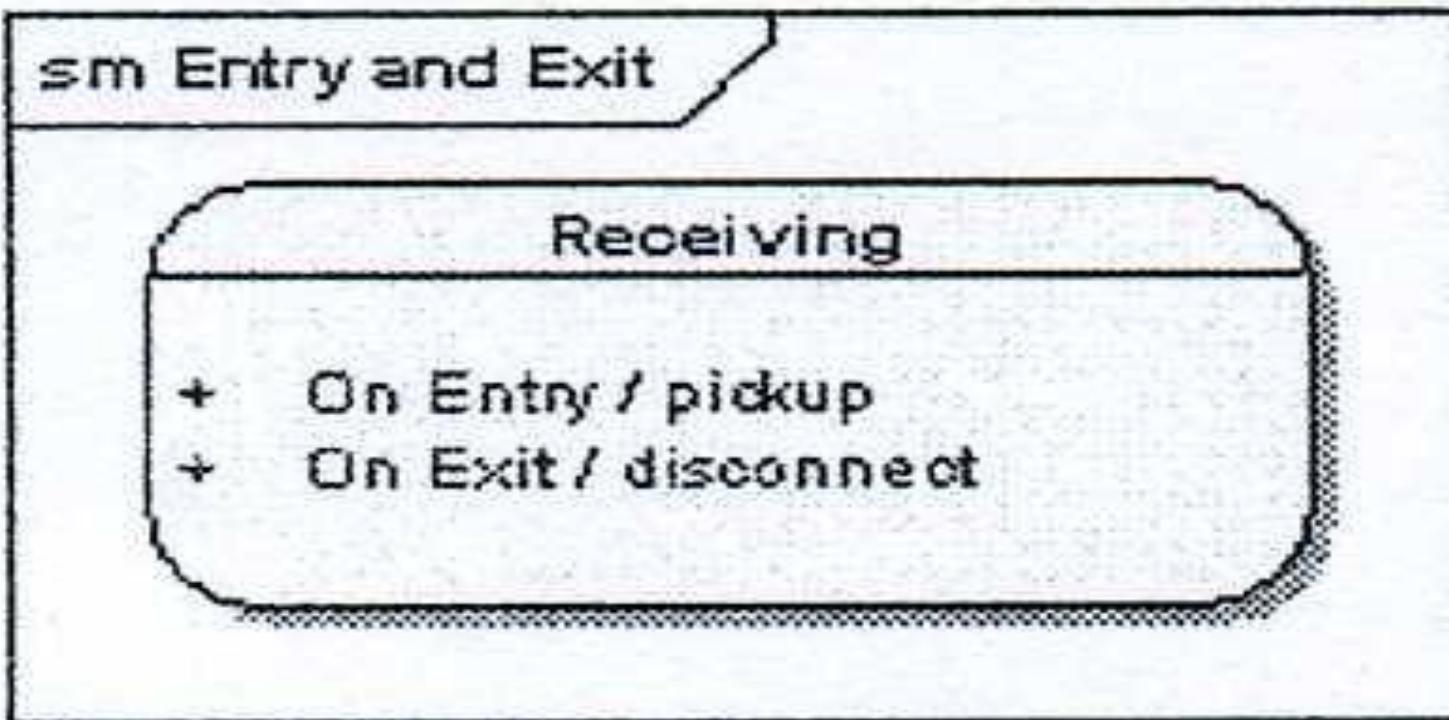
# Kezdeti és végállapotok (initial and final states)



# Átmenetek (transitions)



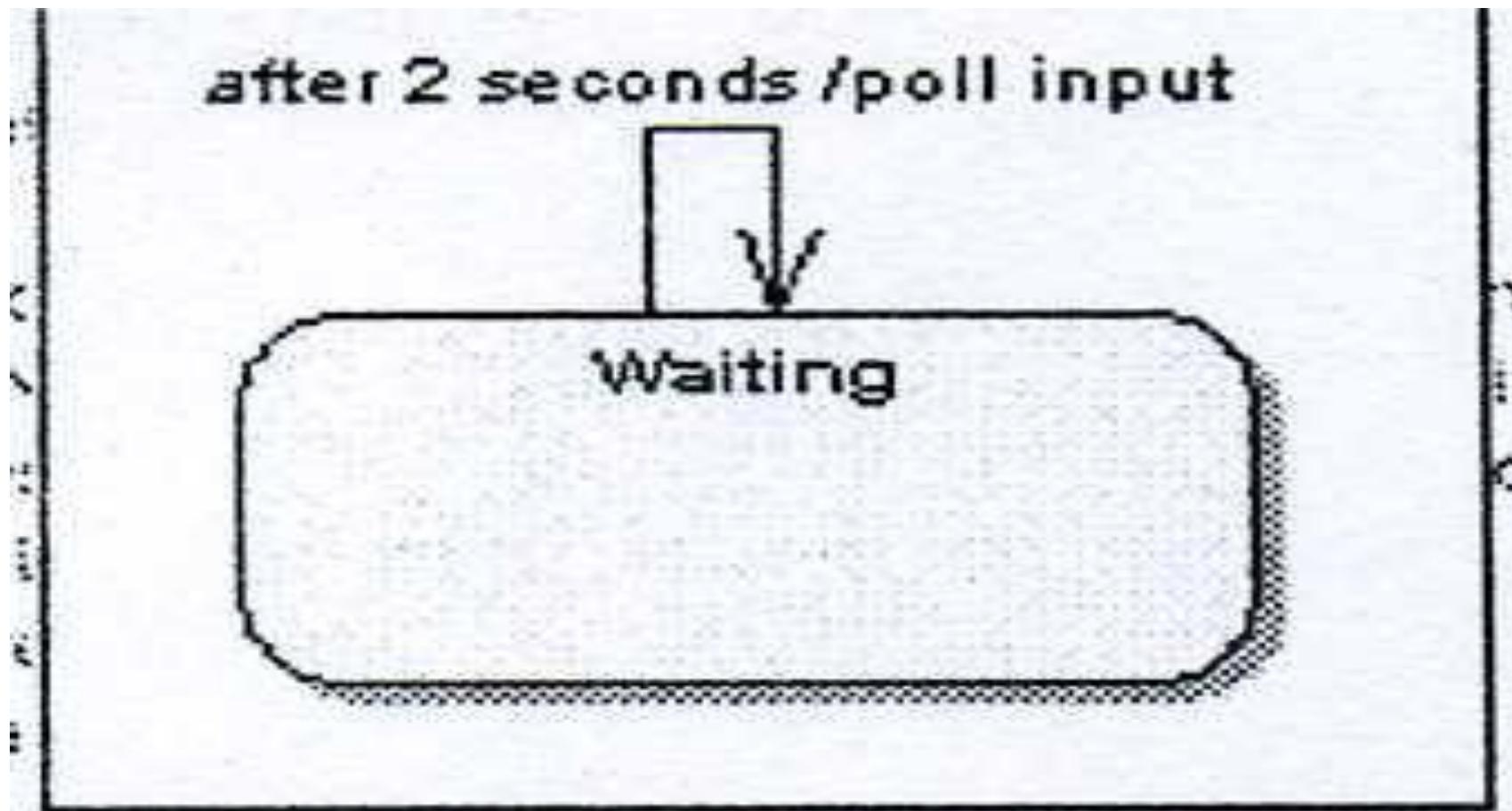
# Állapot akciók (state actions)



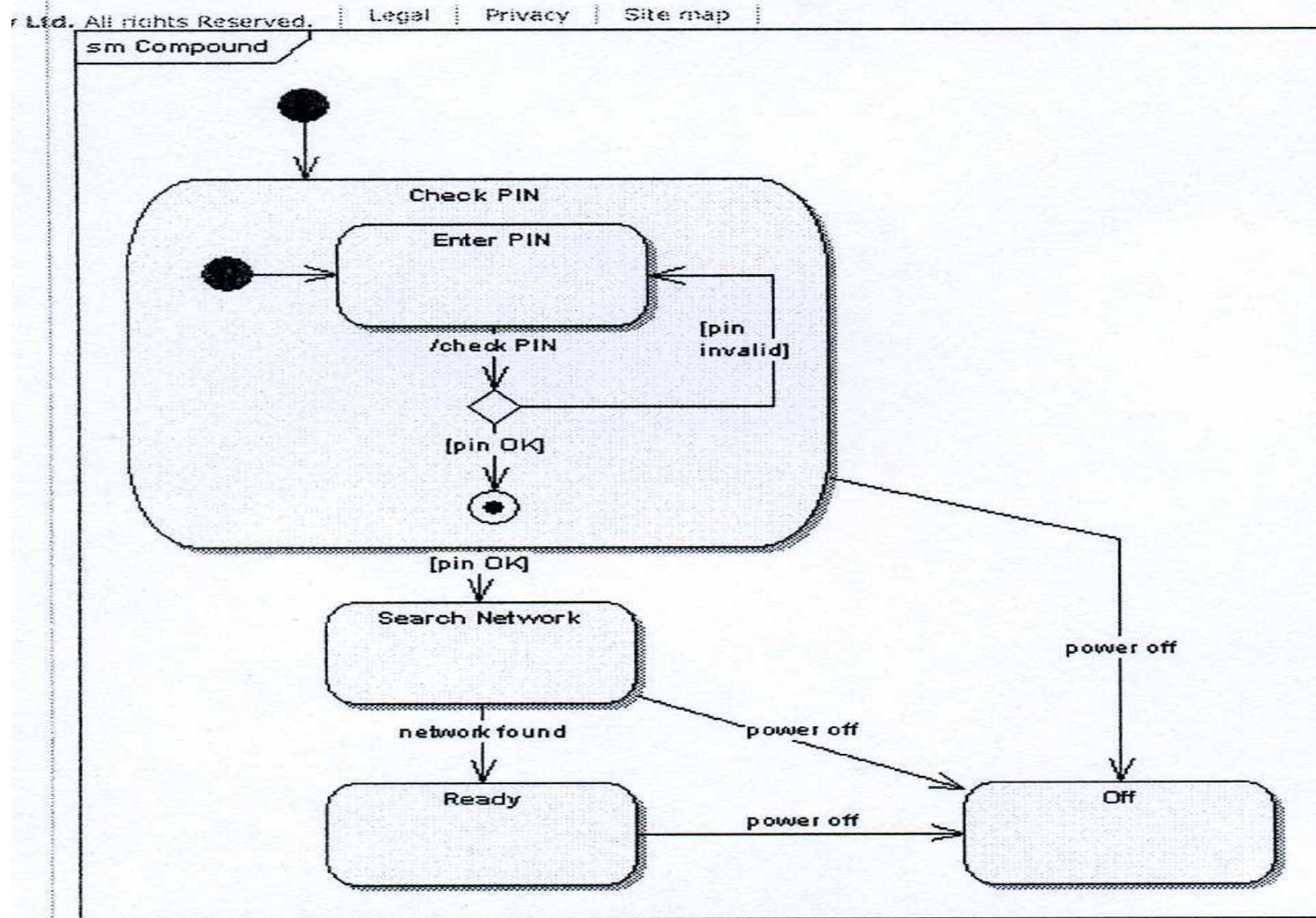
# Self-transitions

Egy állapotnak lehet olyan átmenete, amely önmagához tér vissza

sm Self Transition

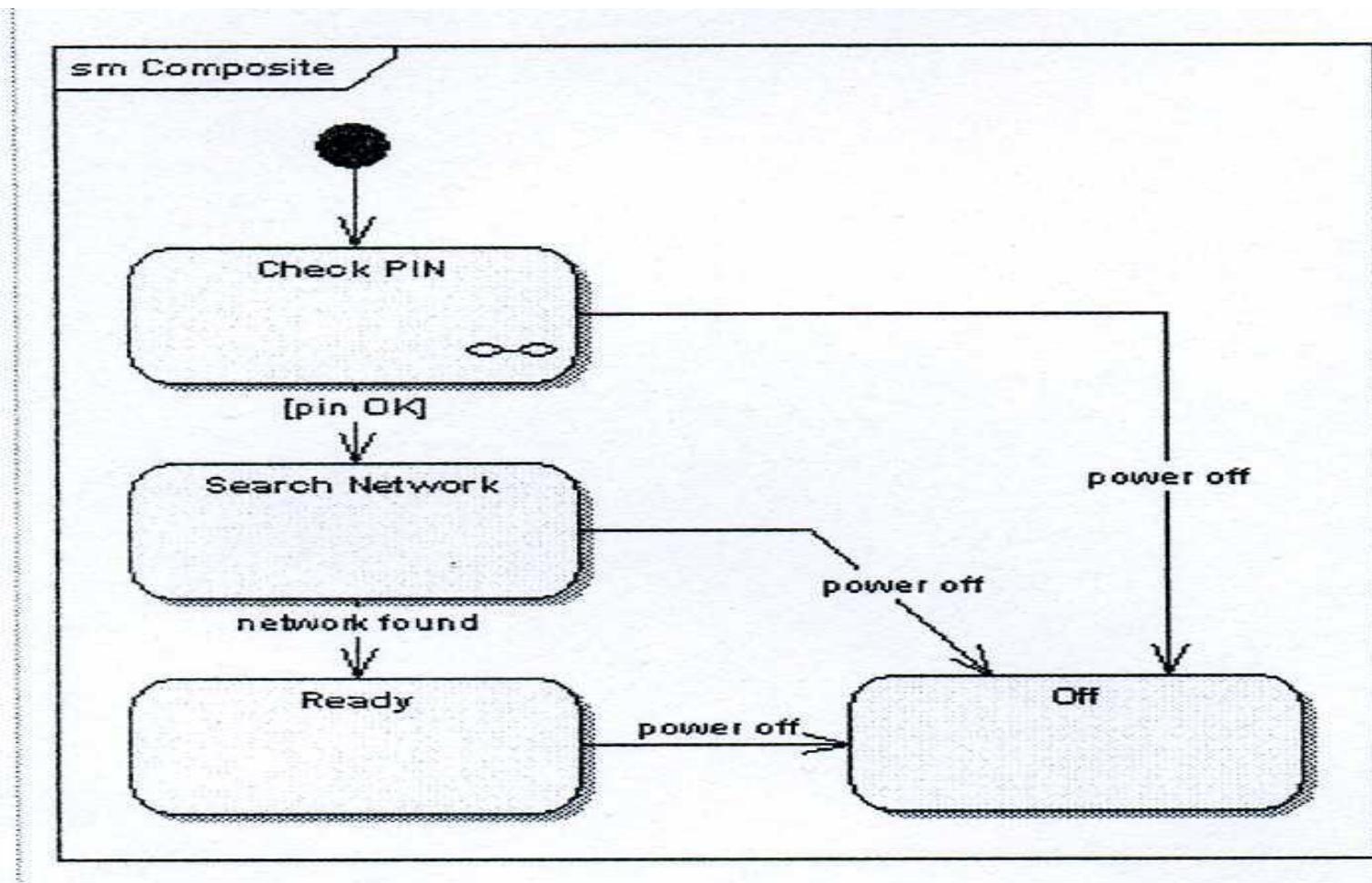


# Összetett állapotok (compound states)

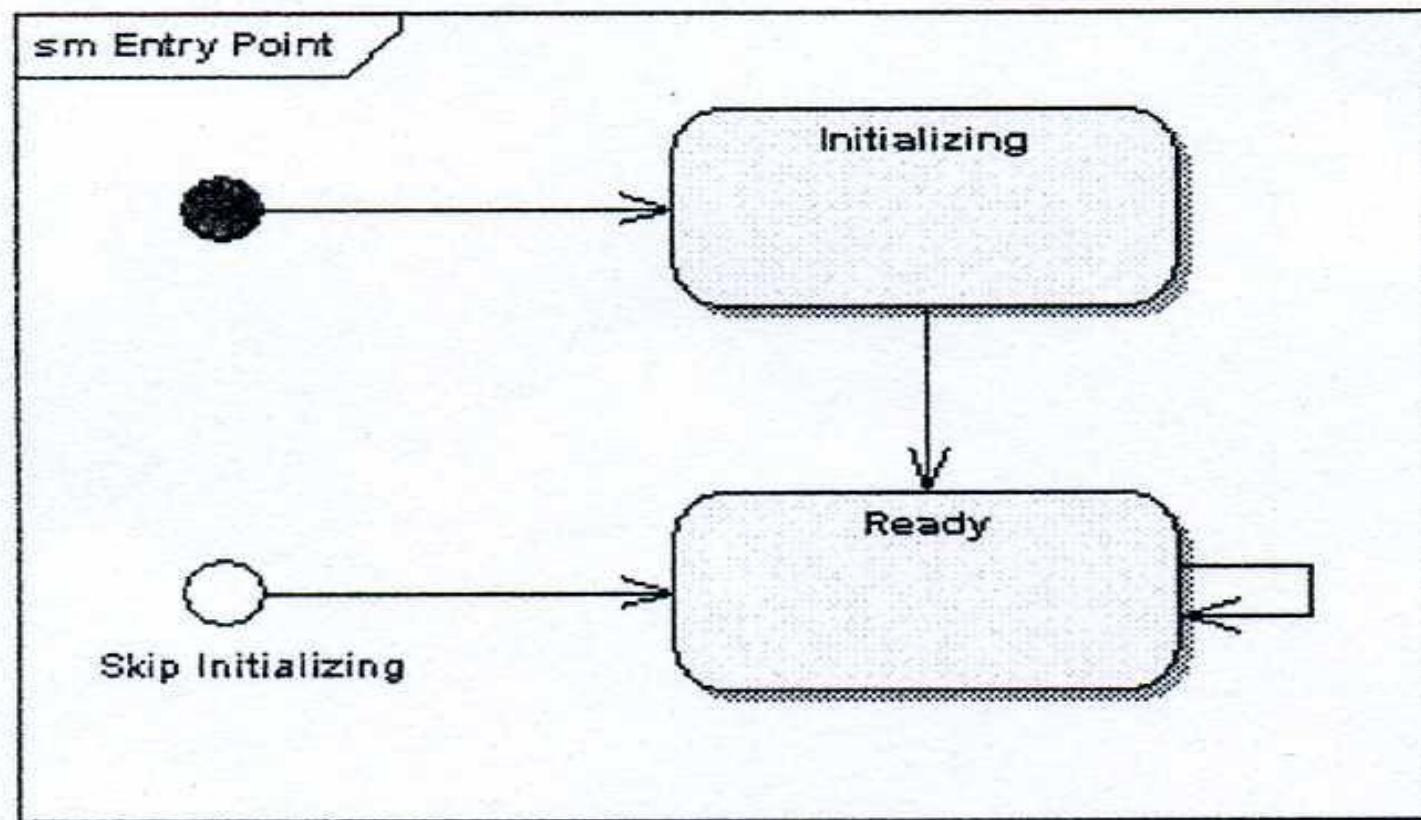


# Alternatív jelölés az előbbire

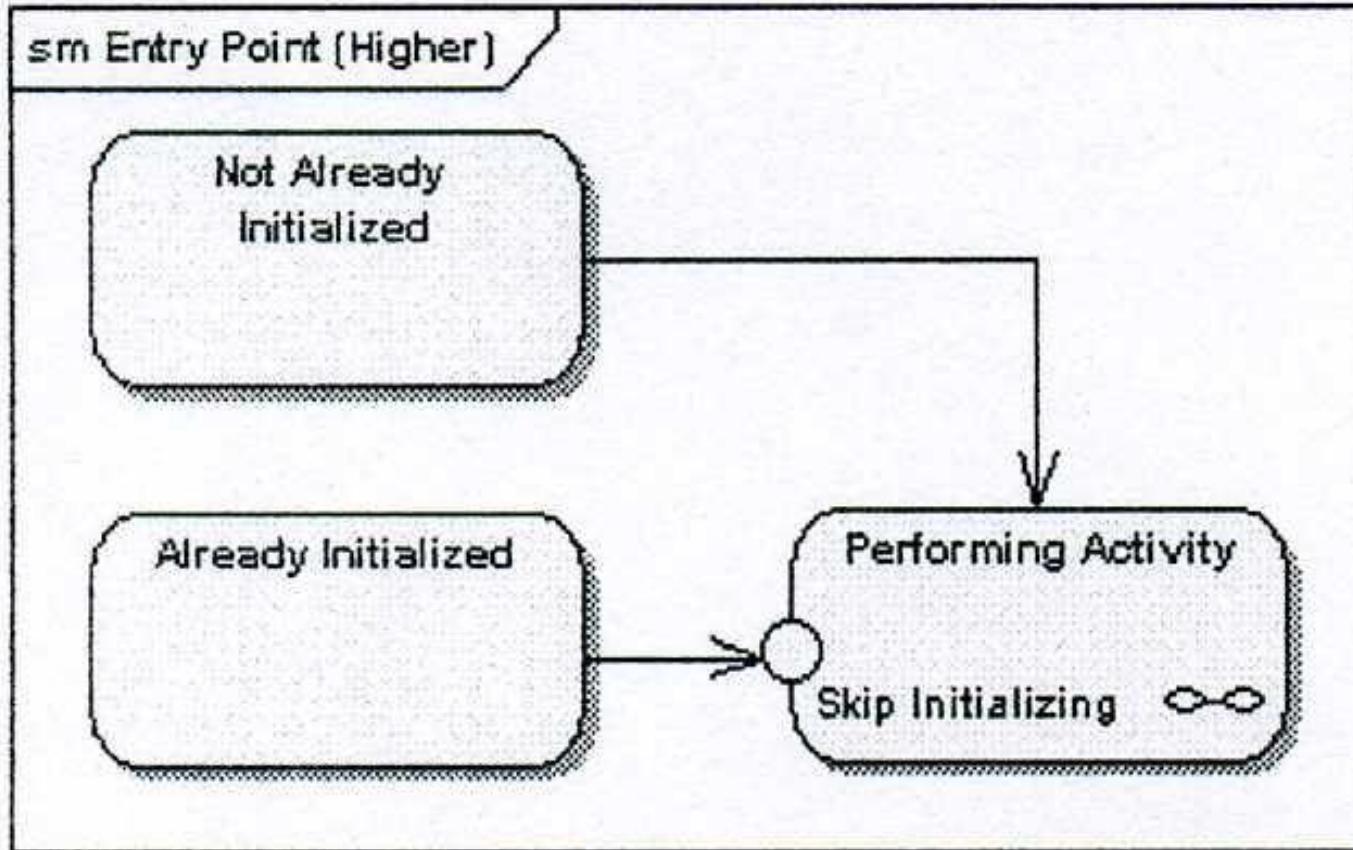
Ezen az ábrán a Check PIN dobozban csak jelöltük, hogy annak részleteit egy másik diagramon adjuk meg



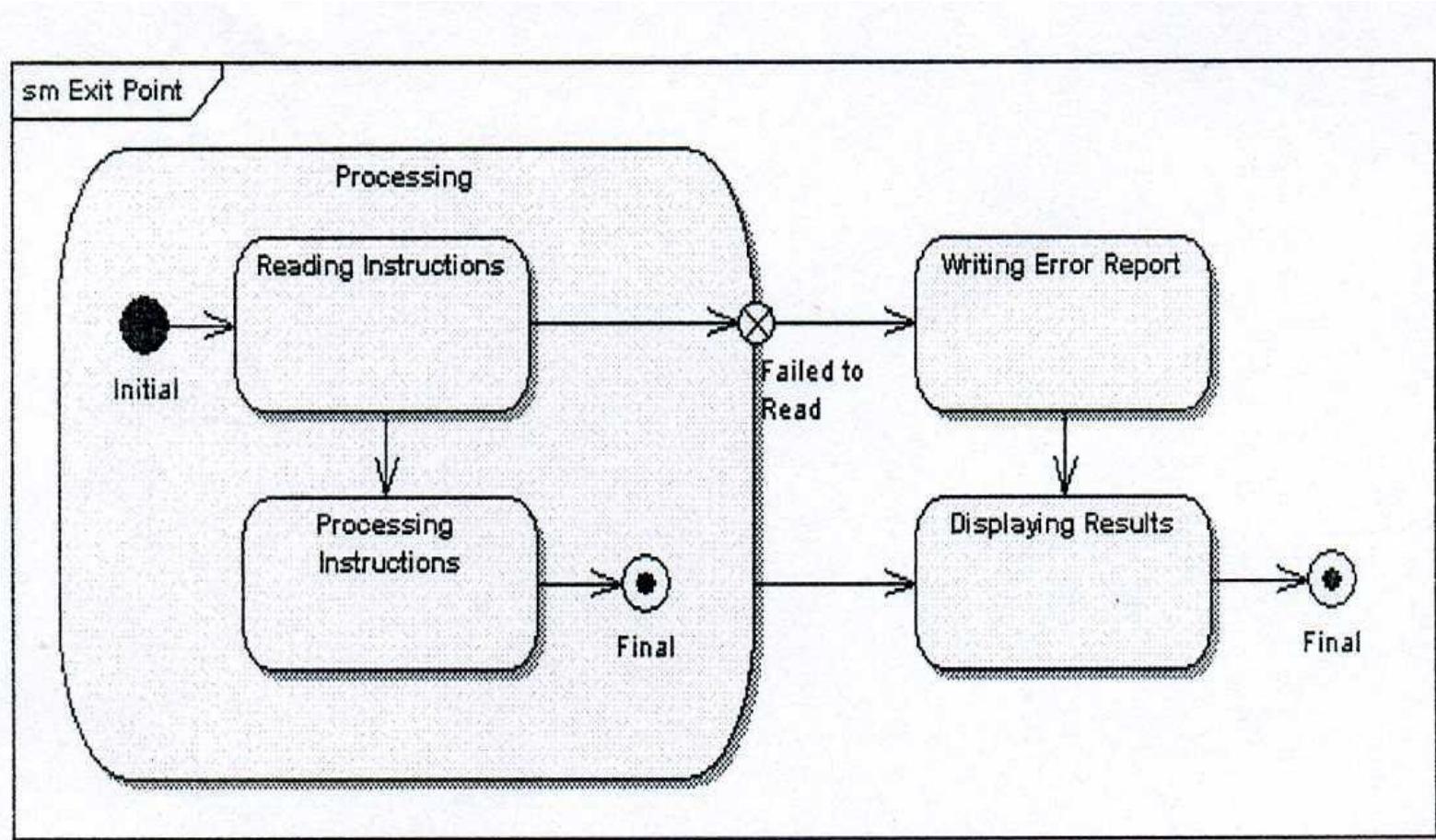
# Belépési pont (entry point)



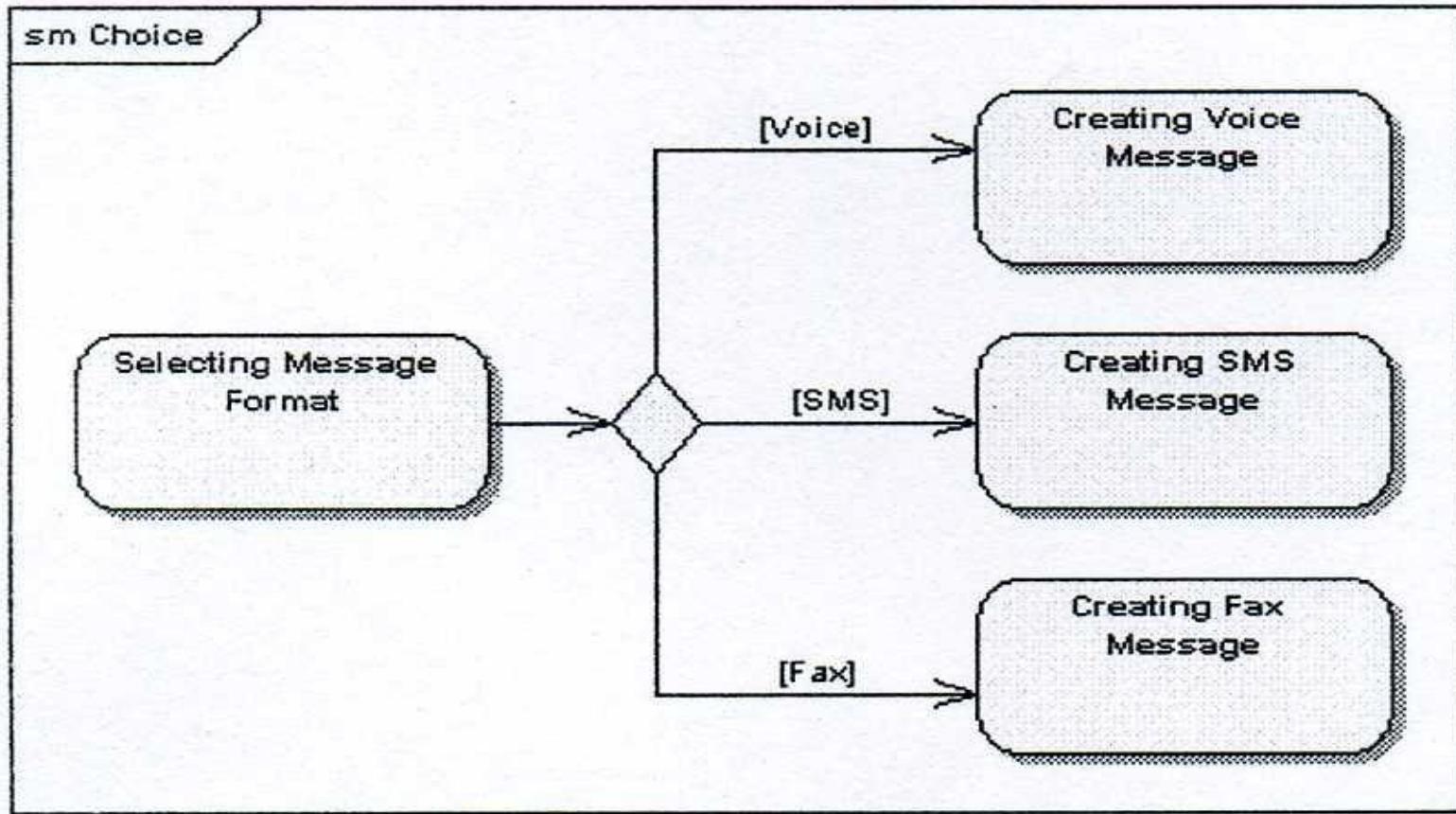
# Magasabb belépési pont (higher entry point)



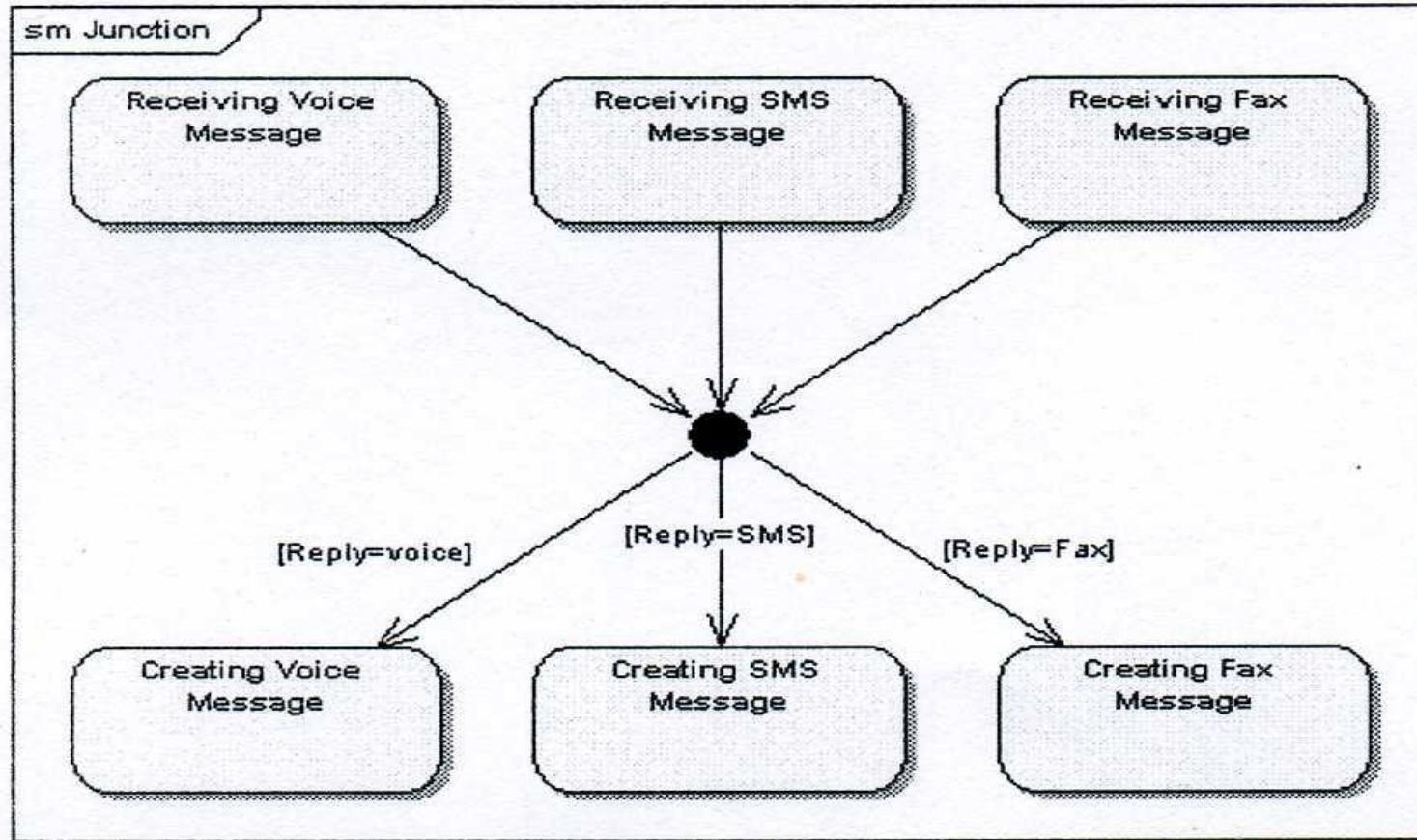
# Kilépési pont (exit point)



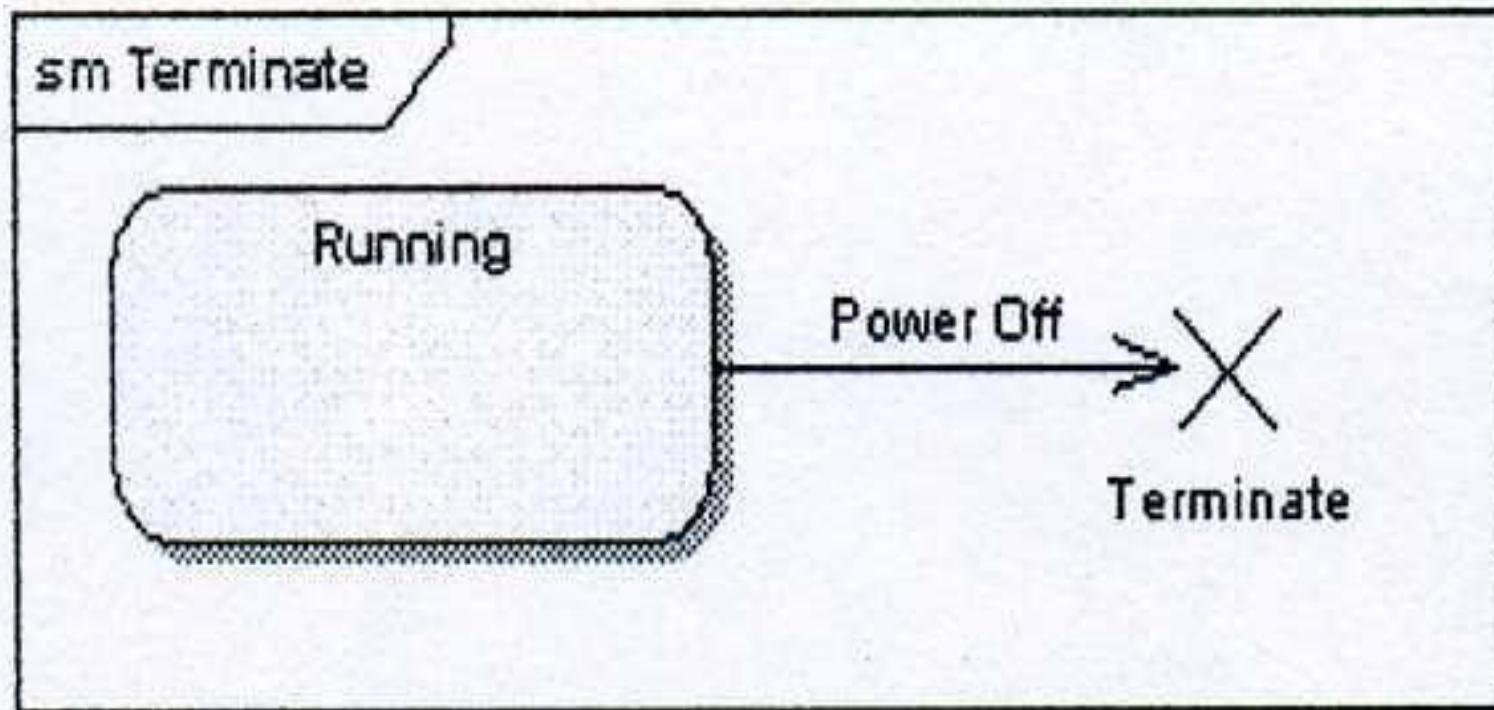
# Választás (choice pseudo-state)



# Csomóponti elágazás (junction pseudo-state)

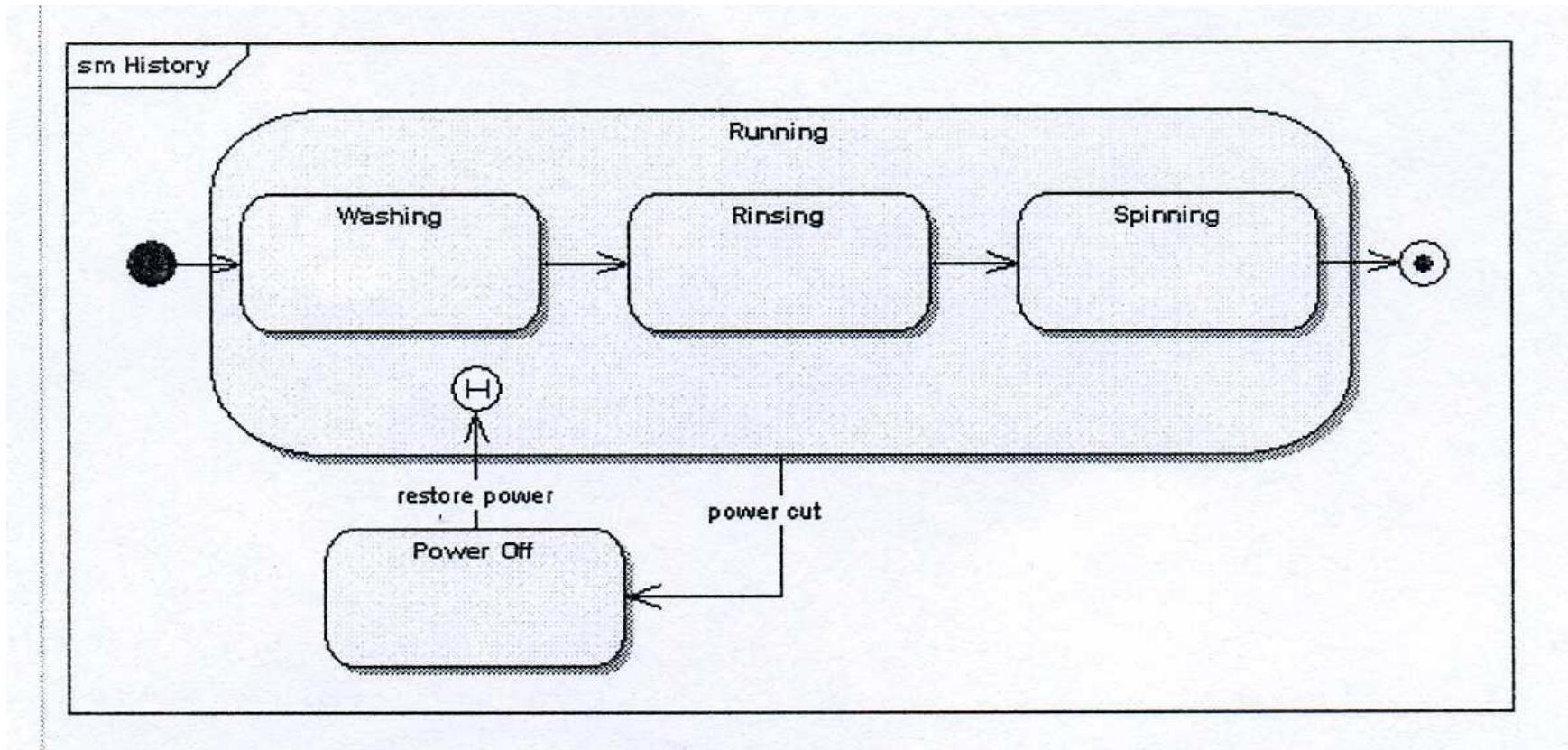


# Terminál állapot (terminal pseudo state)

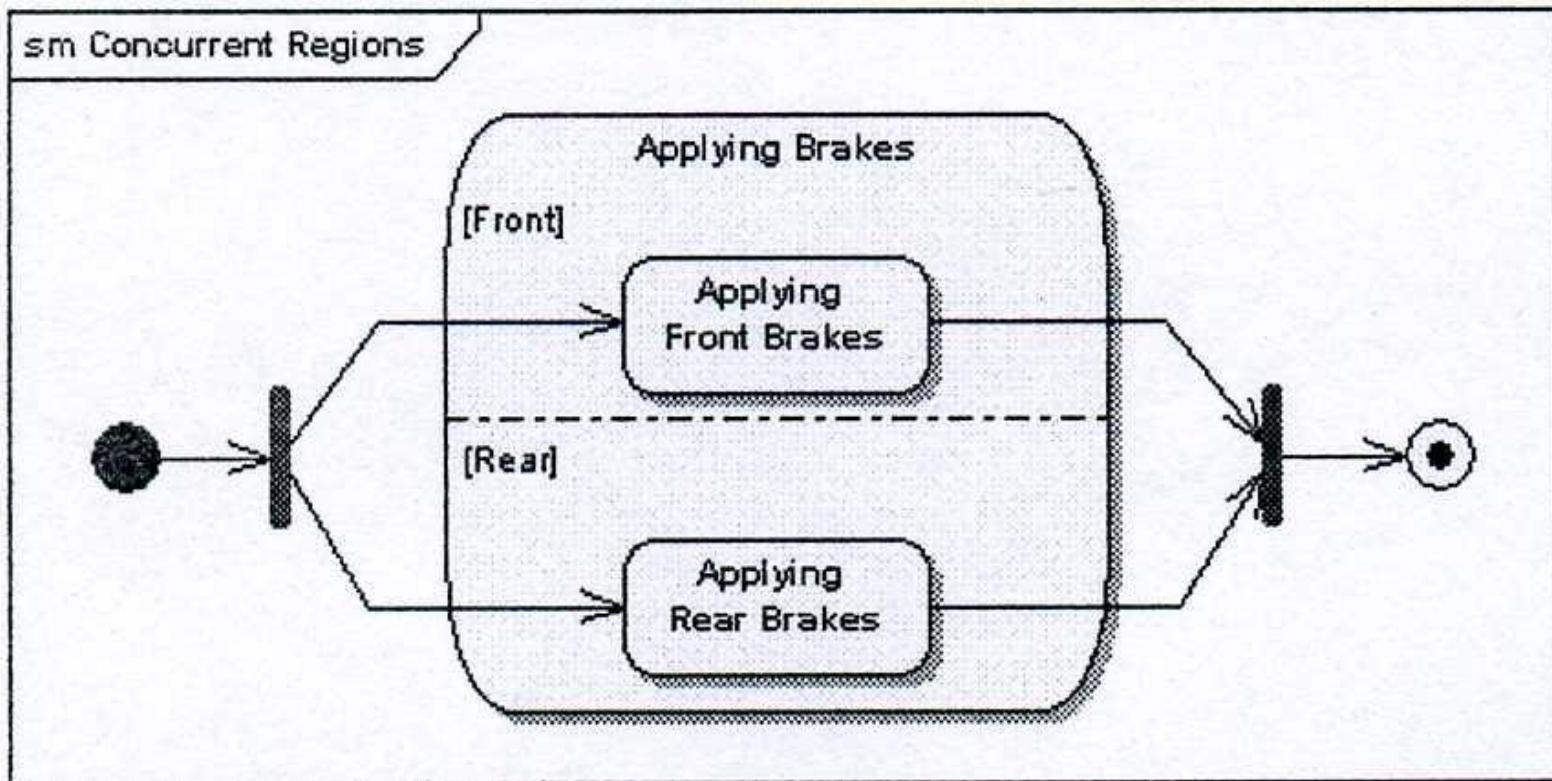


# Történeti állapot (history state)

Példa a mosógép működése, áramkimaradás esetén a gép „power off” állapotba kerül, majd az áram visszatérte után a „History State” szimbólum jelzi, hogy a mosás a megszakítás állapotától folytatódik.



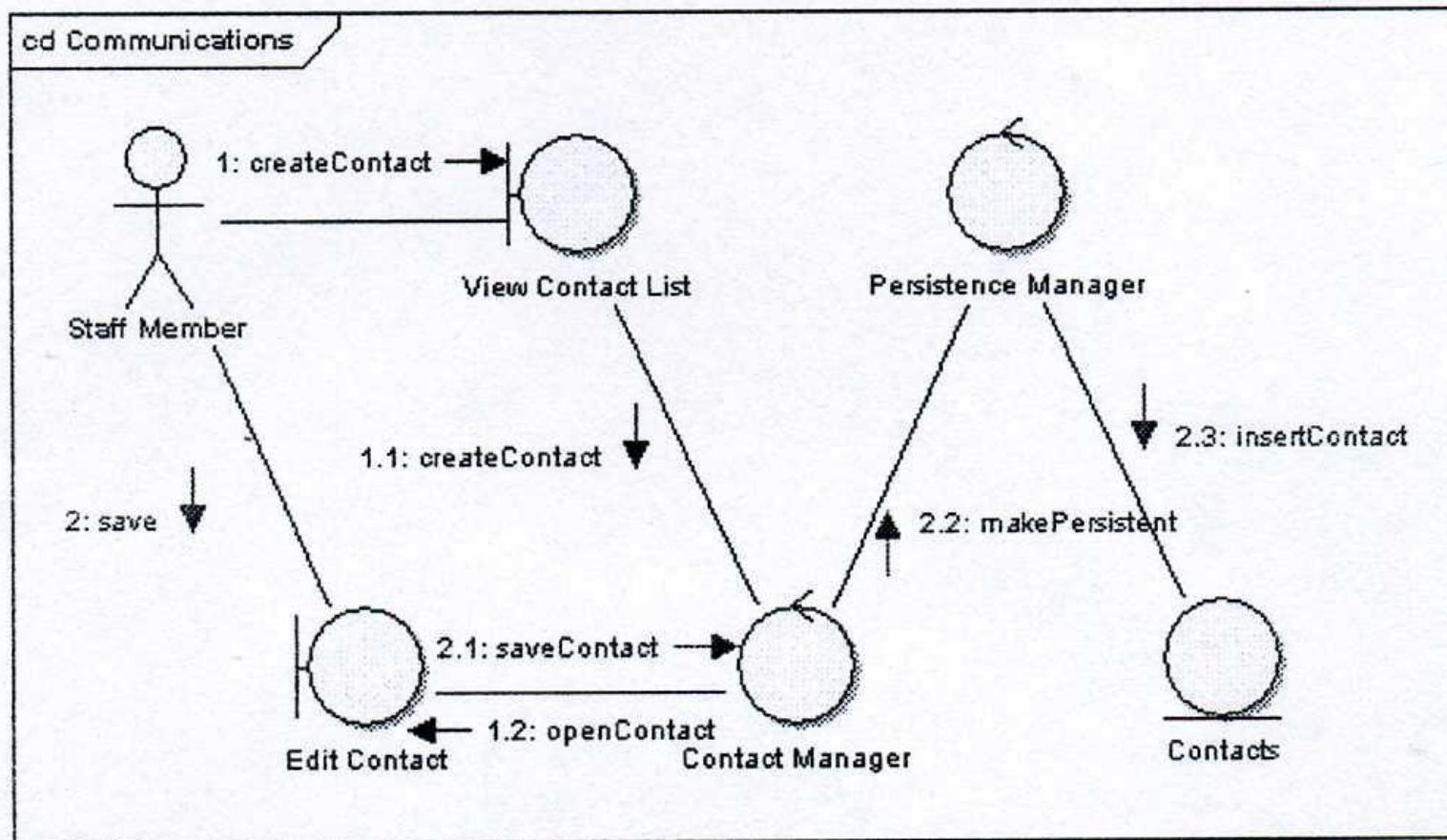
# Konkurens régiók (concurrent regions)



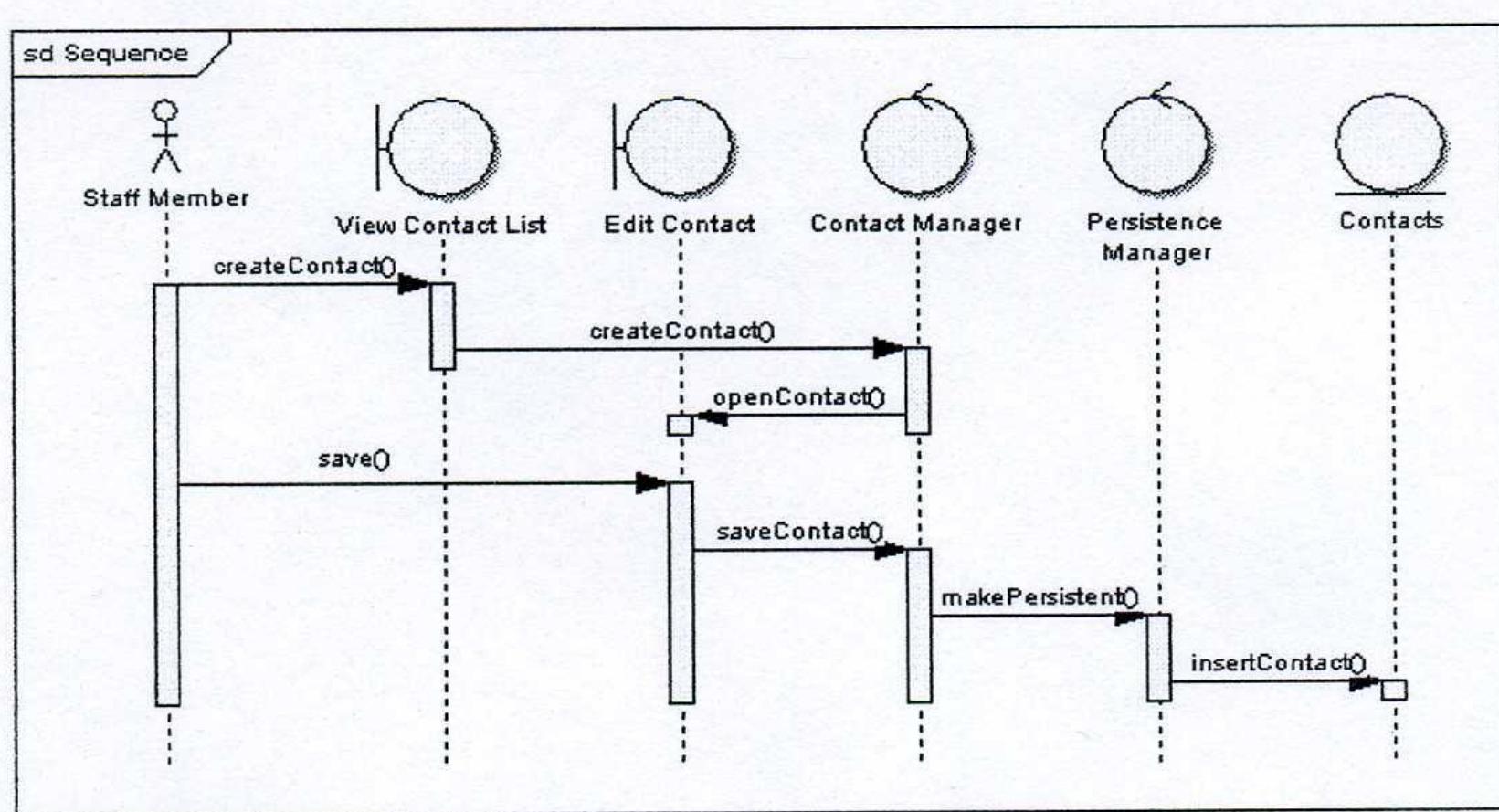
# UML2: kommunikációs diagram

- Korábban együttműködési diagramnak hívták.
- Hasonló információkat hordoz mint a szekvencia diagram
- A diagram elsődleges fókuszában az objektumok kapcsolatai vannak.
- A következő két diagram ugyanazon információt hordozza, vessük össze őket.

# Példa kommunikációs diagramra



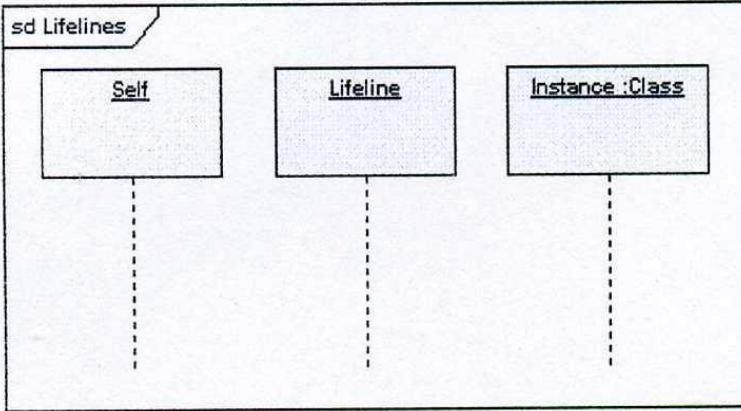
# Ugyanazon példa szekvencia diagramja



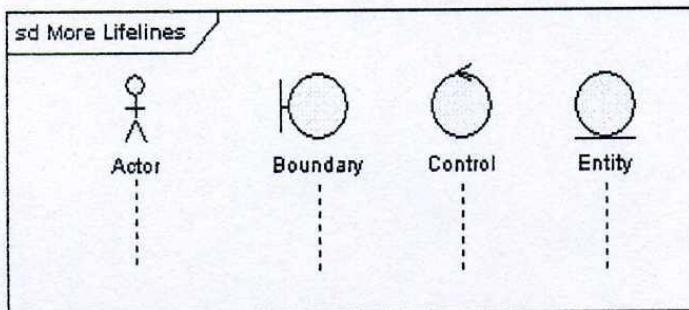
# UML2: szekvencia diagram

- Elemei:
  - életvonalak (lifelines),
  - üzenetek (messages),
  - rekurzív üzenetek (self messages),
  - elvesztett és megtalált üzenetek (lost and found messages),
  - életvonal kezdete és vége (lifeline start and end),
  - idő és időtartam megszorítások (duration and time constraints),
  - kombinált részek (combined fragments),
  - kapu (gate),
  - rész dekompozíció (part decomposition),
  - állapot invariáns (state invariat / continuations).

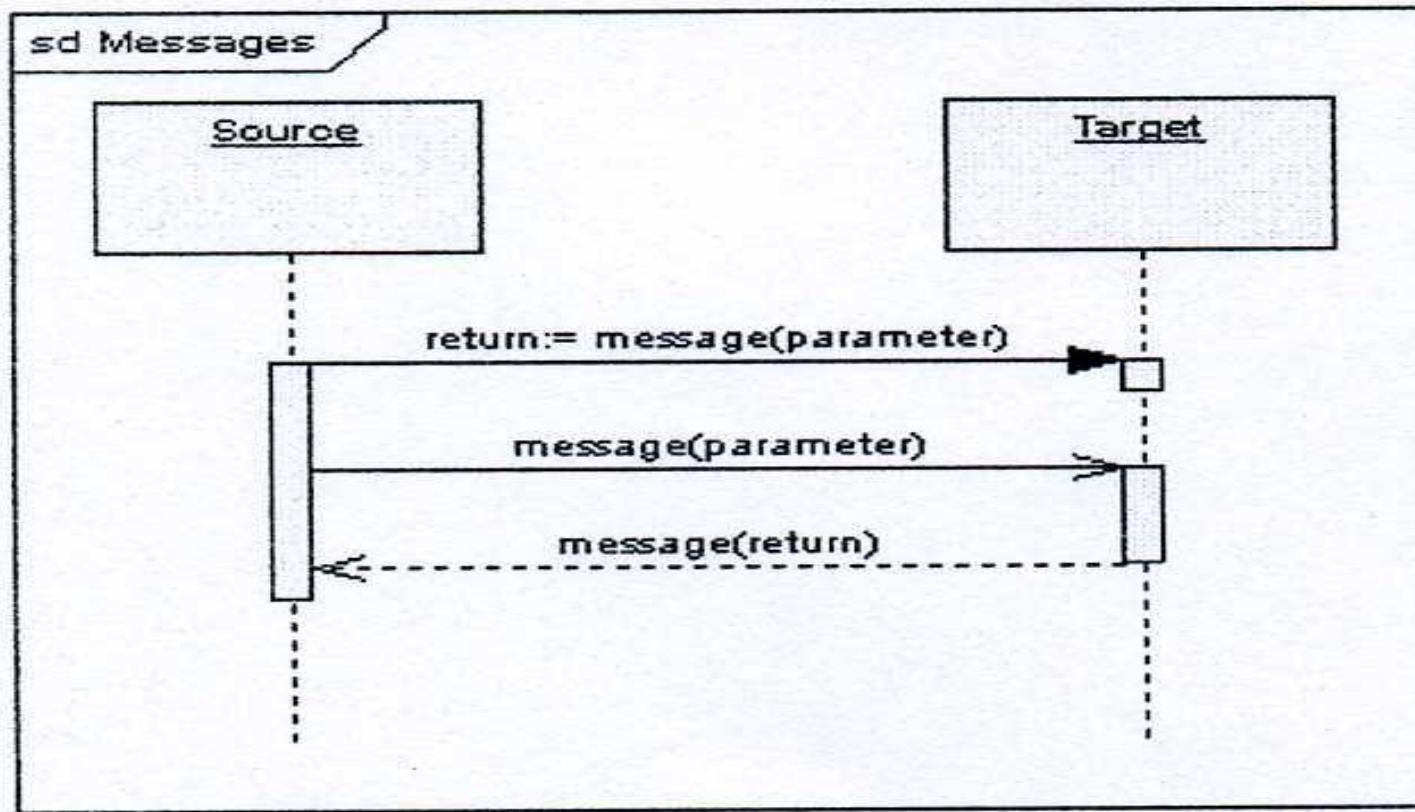
# Életvonalak (lifelines)



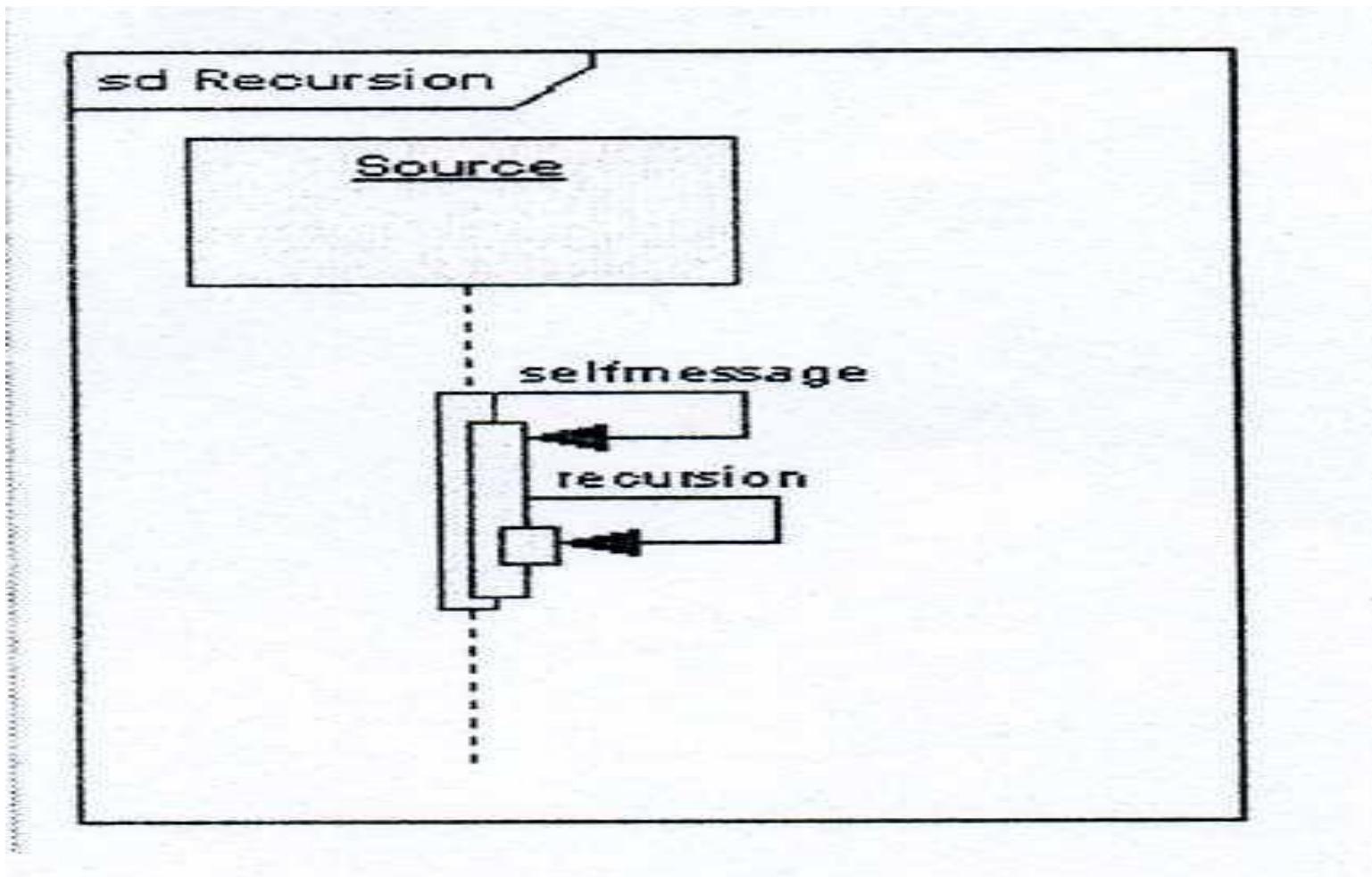
Sometimes a sequence diagram will have a lifeline with an actor element symbol at its head. This will usually be the case if the sequence diagram is owned by a use case. Boundary, control and entity elements from robustness diagrams can also own lifelines.



# Üzenetek (messages)

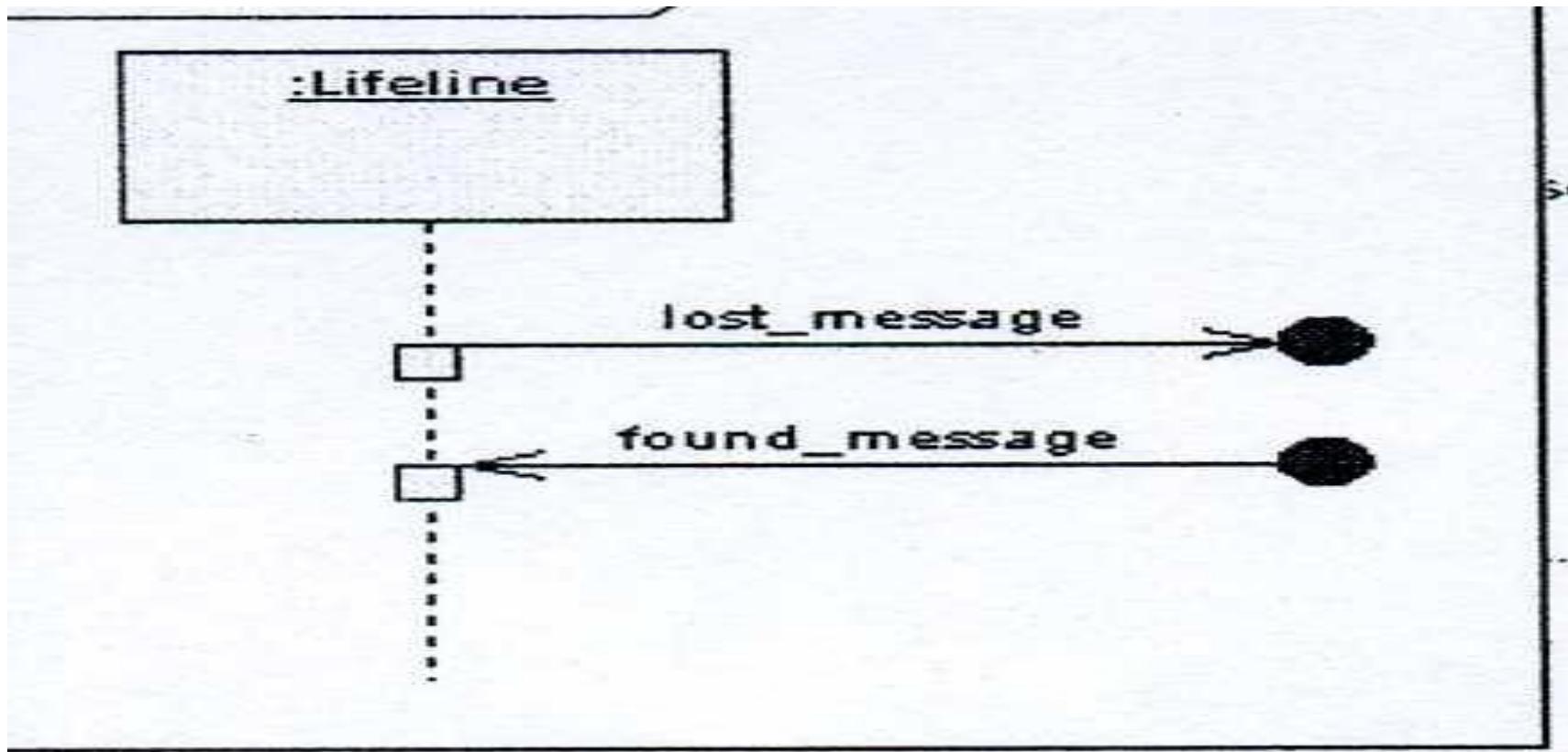


# Rekurzív üzenetek (self messages)

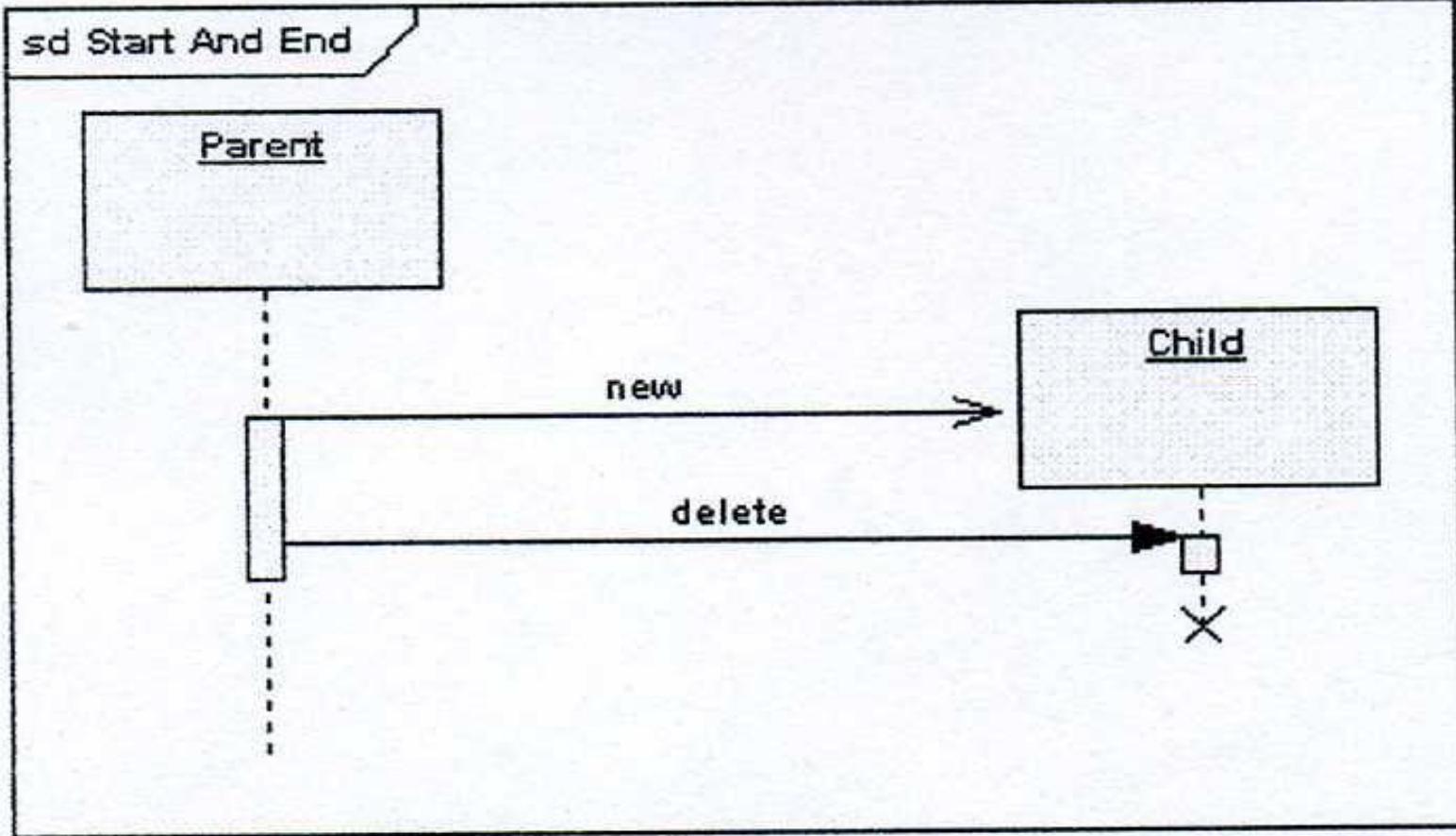


# Elvesztett és megtalált üzenetek (lost and found messages)

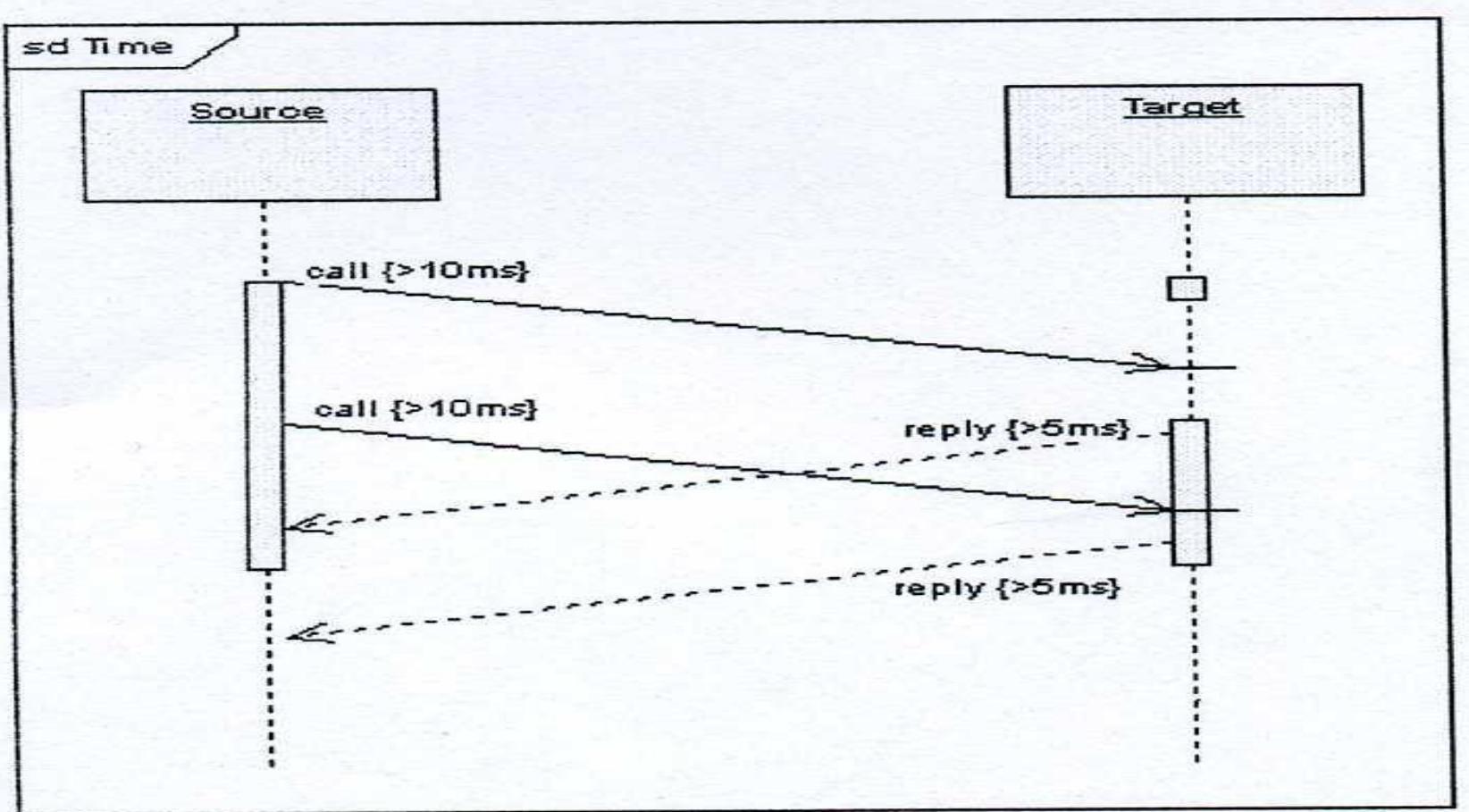
Sd Lost and Found



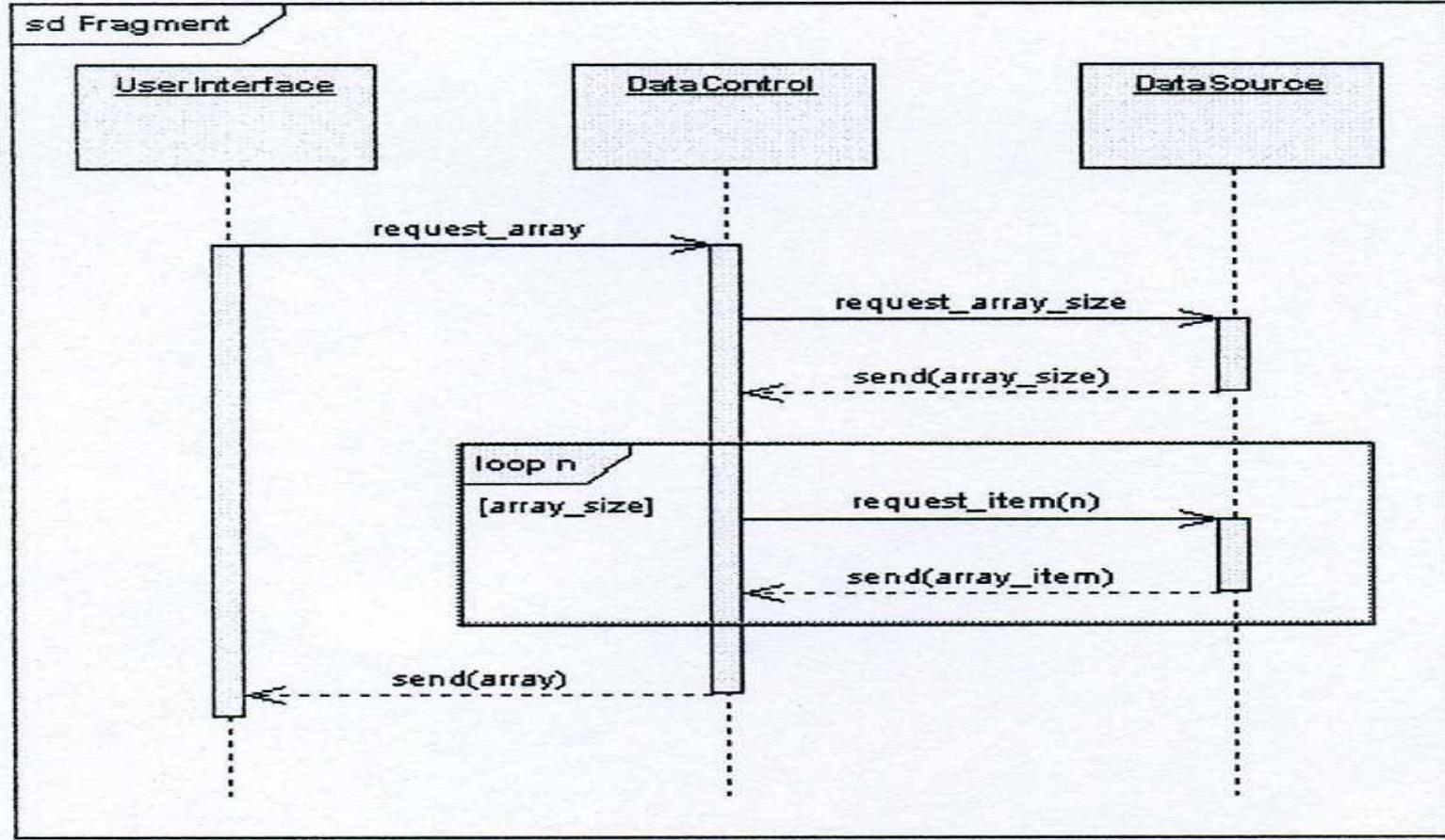
# Életvonal kezdete és vége (lifeline start and end),



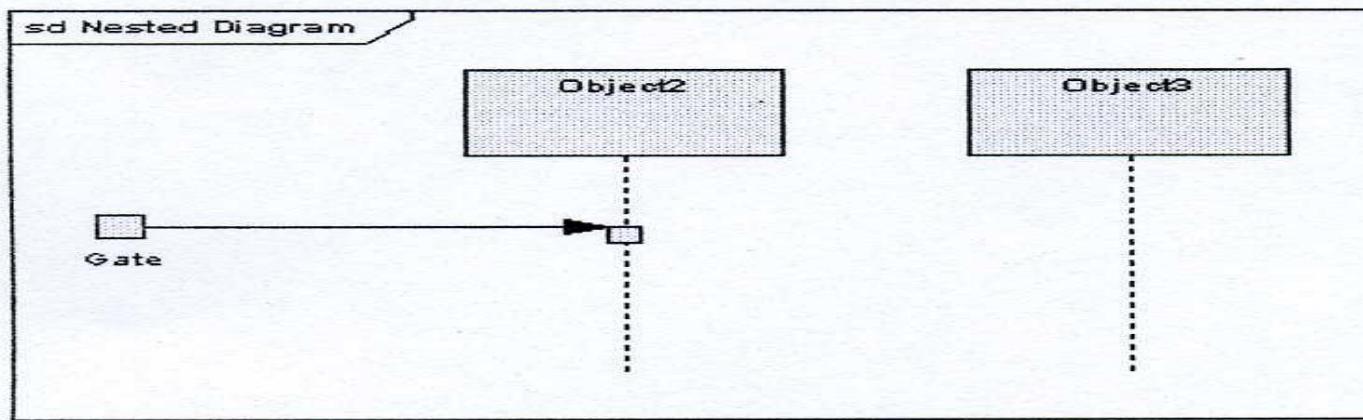
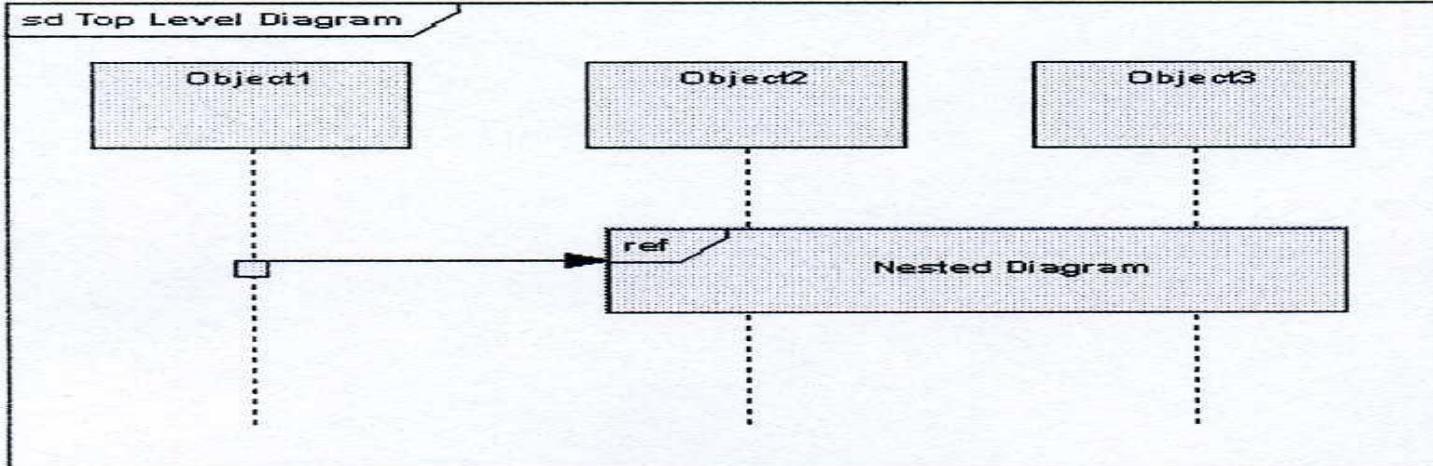
# Idő és időtartam megszorítások (duration and time constraints)



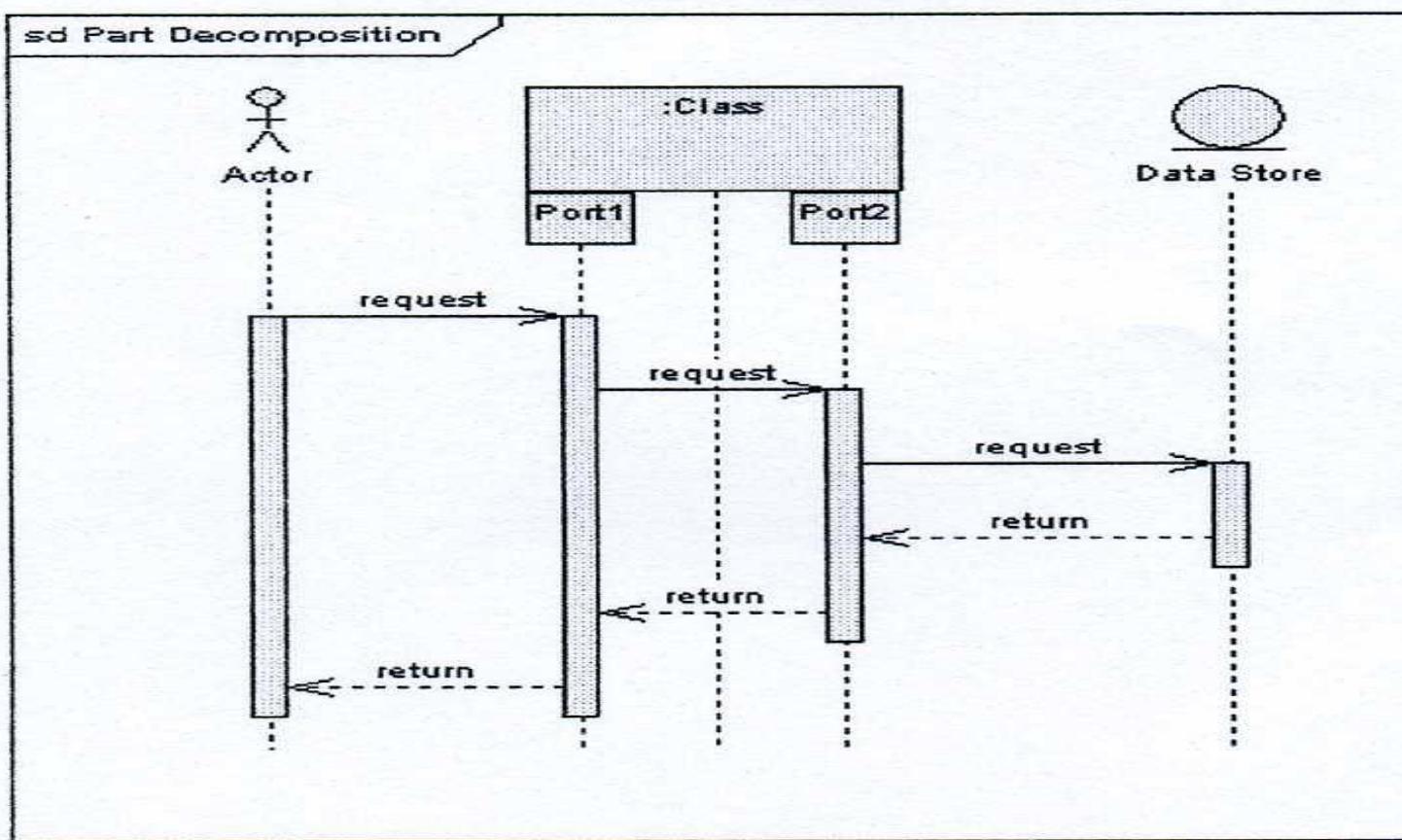
# Kombinált részek (combined fragments)



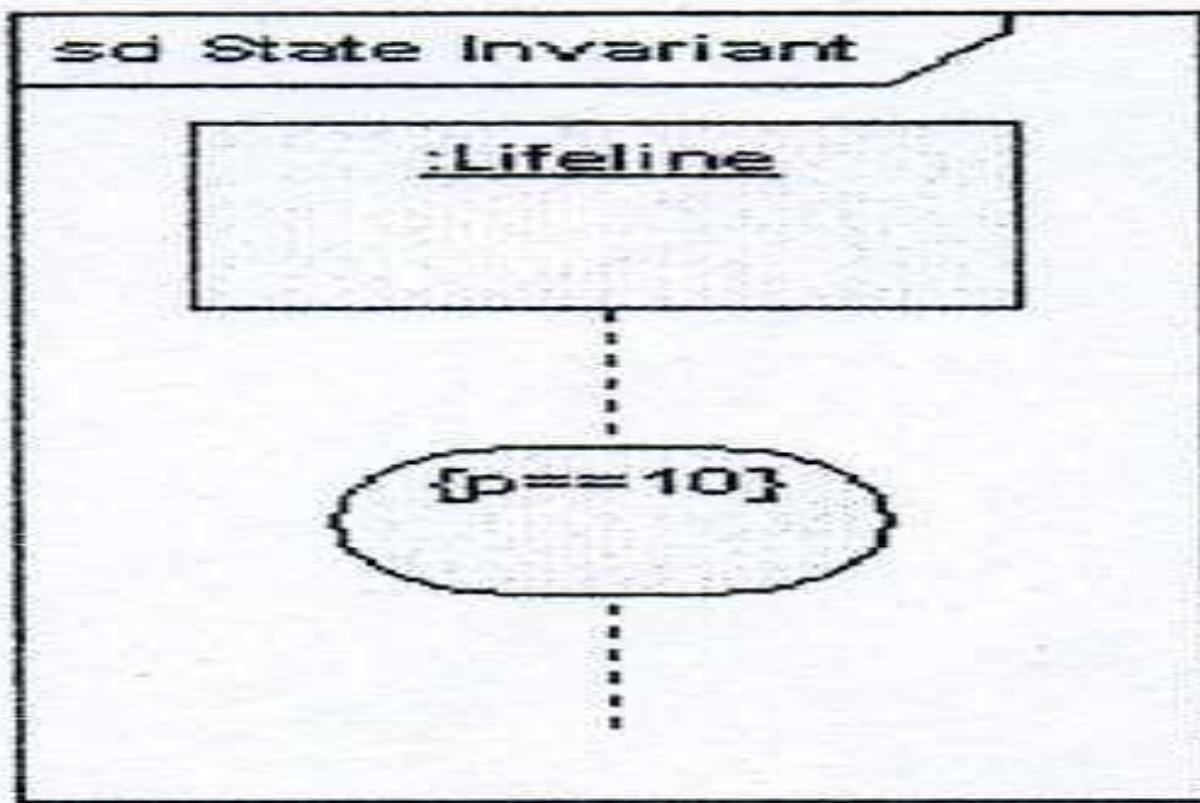
# Kapu (gate)



# Rész dekompozíció (part decomposition)



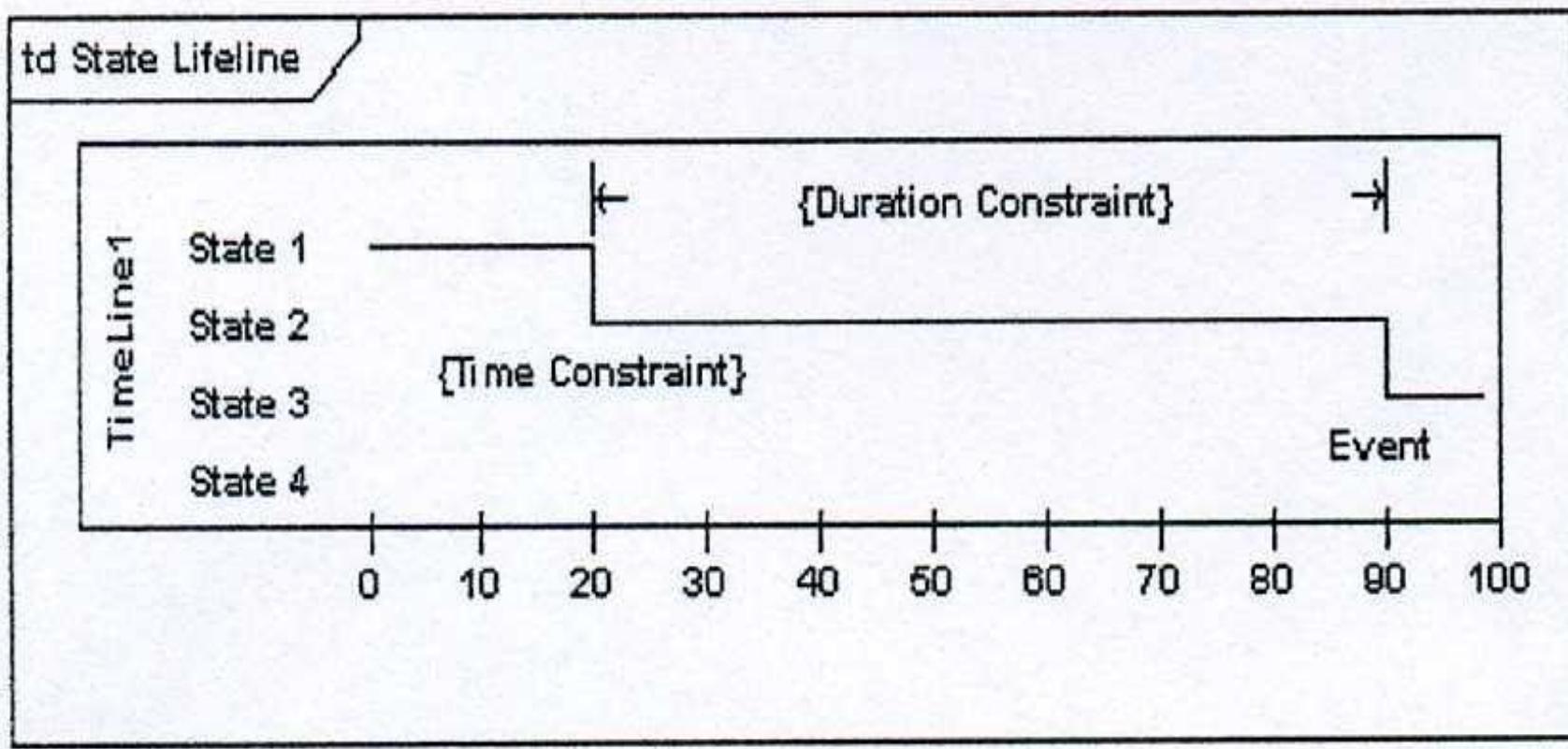
# Állapot invariáns (state invariant / continuations)



# UML2: időzítési diagram

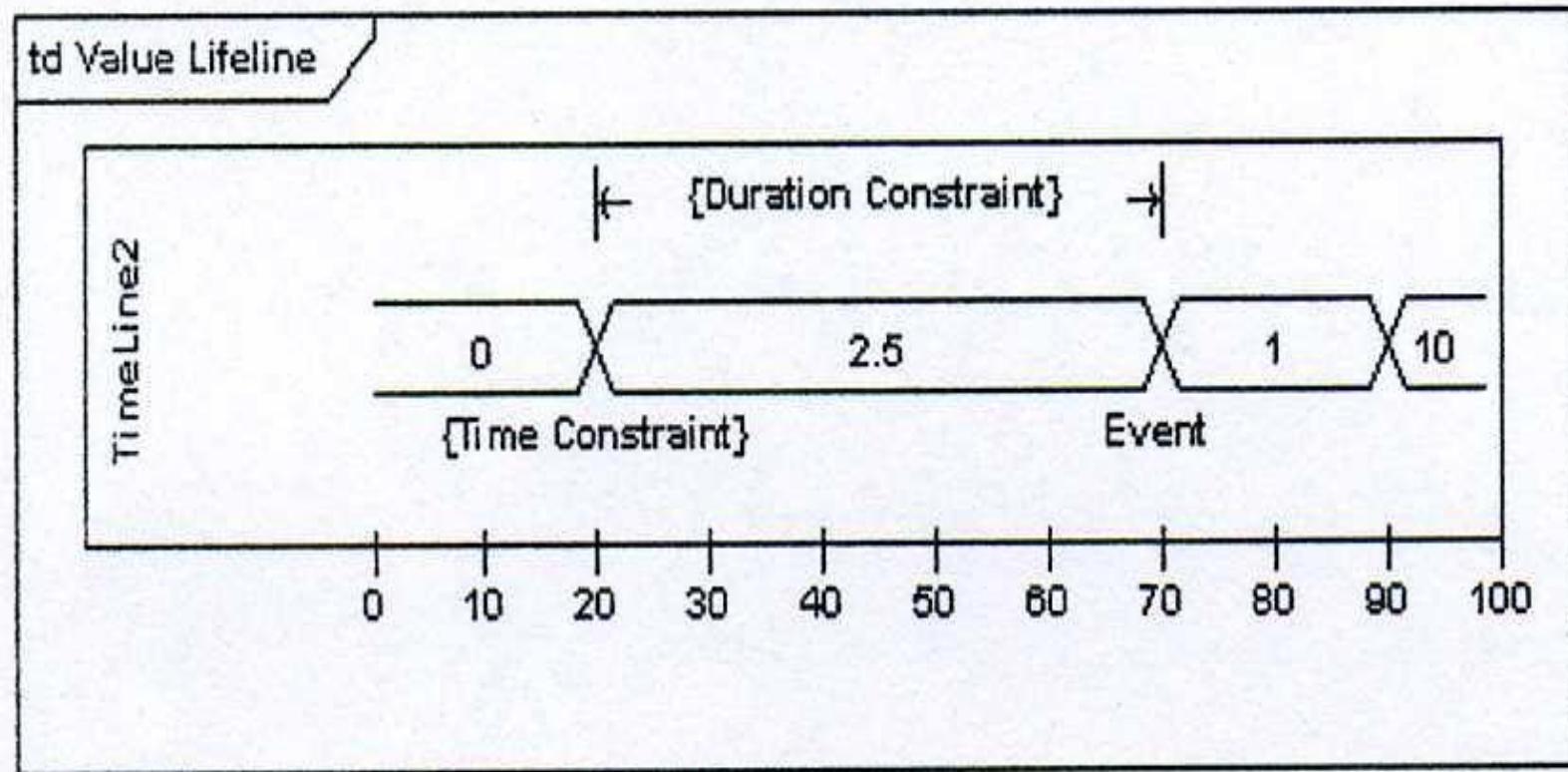
- Az időzítési diagram (timing diagram) elemei:
  - állapot életvonal (state lifeline),
  - érték életvonal (value lifeline),
  - az előbbi kettő kombinációja (putting it all together).
- A következő három ábra egy-egy példát tartalmaz az egyes elemekre vonatkozóan.

# State lifeline

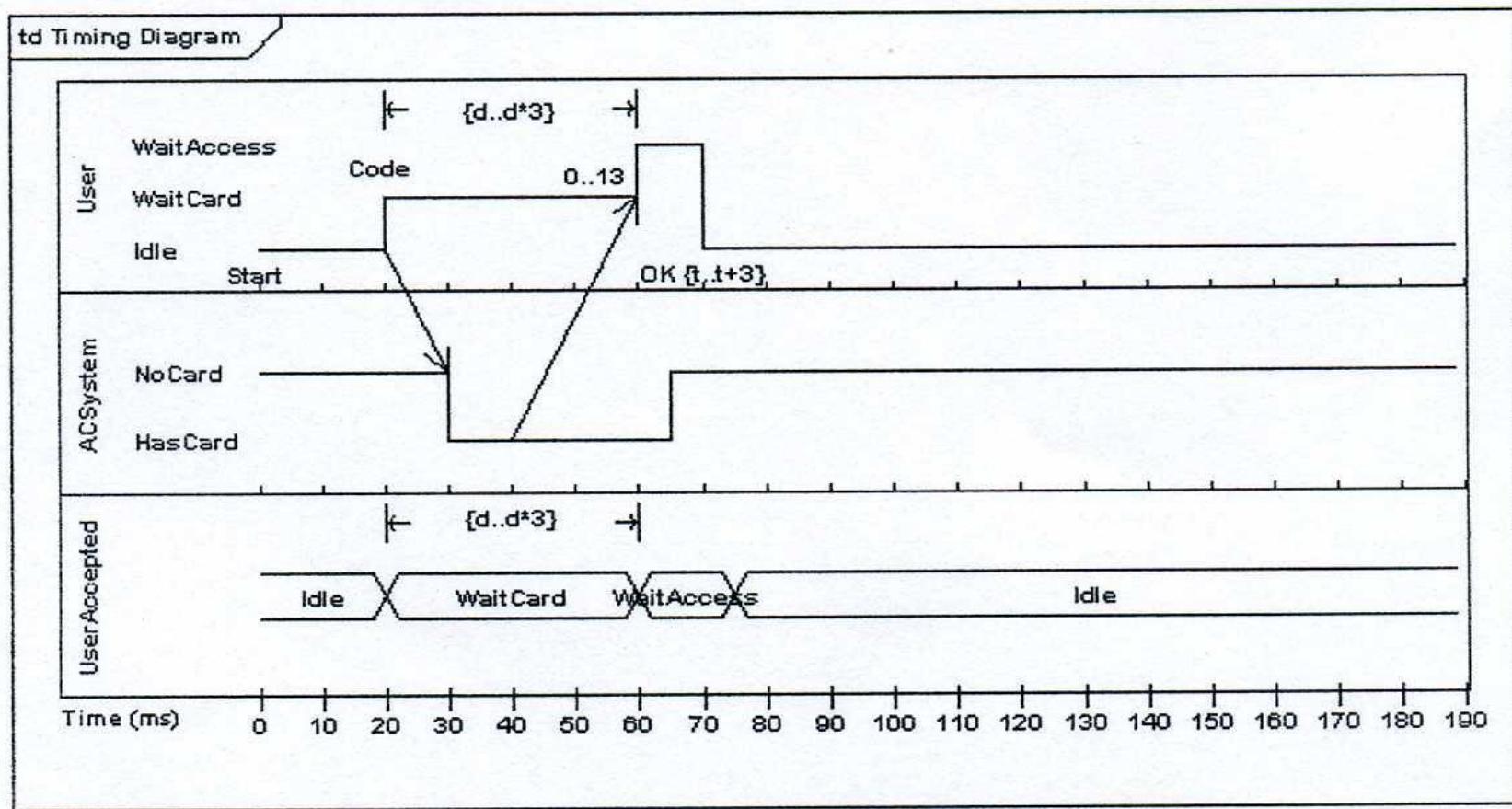


# Value lifeline

Timing is shown below.



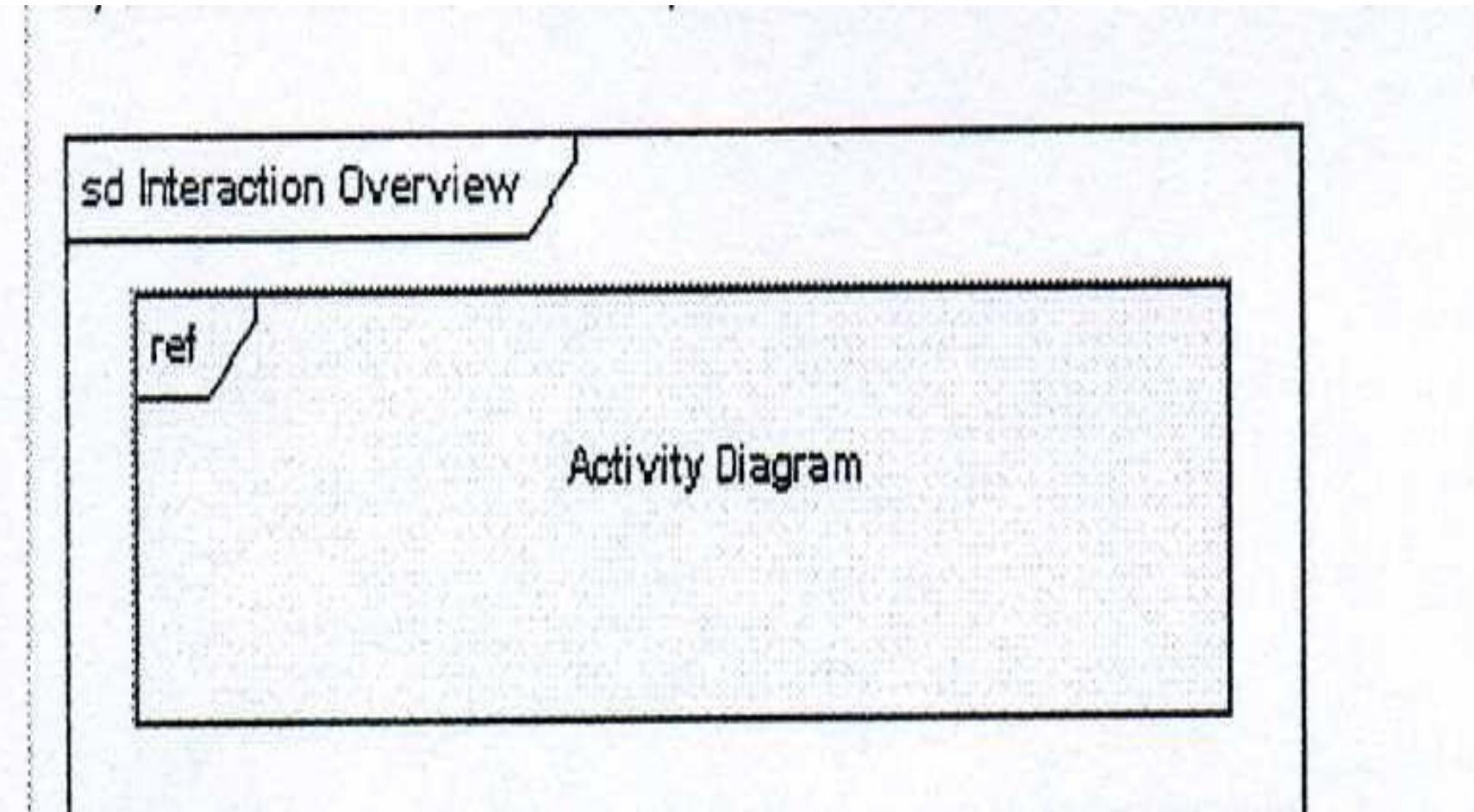
# A kettő kombinációja



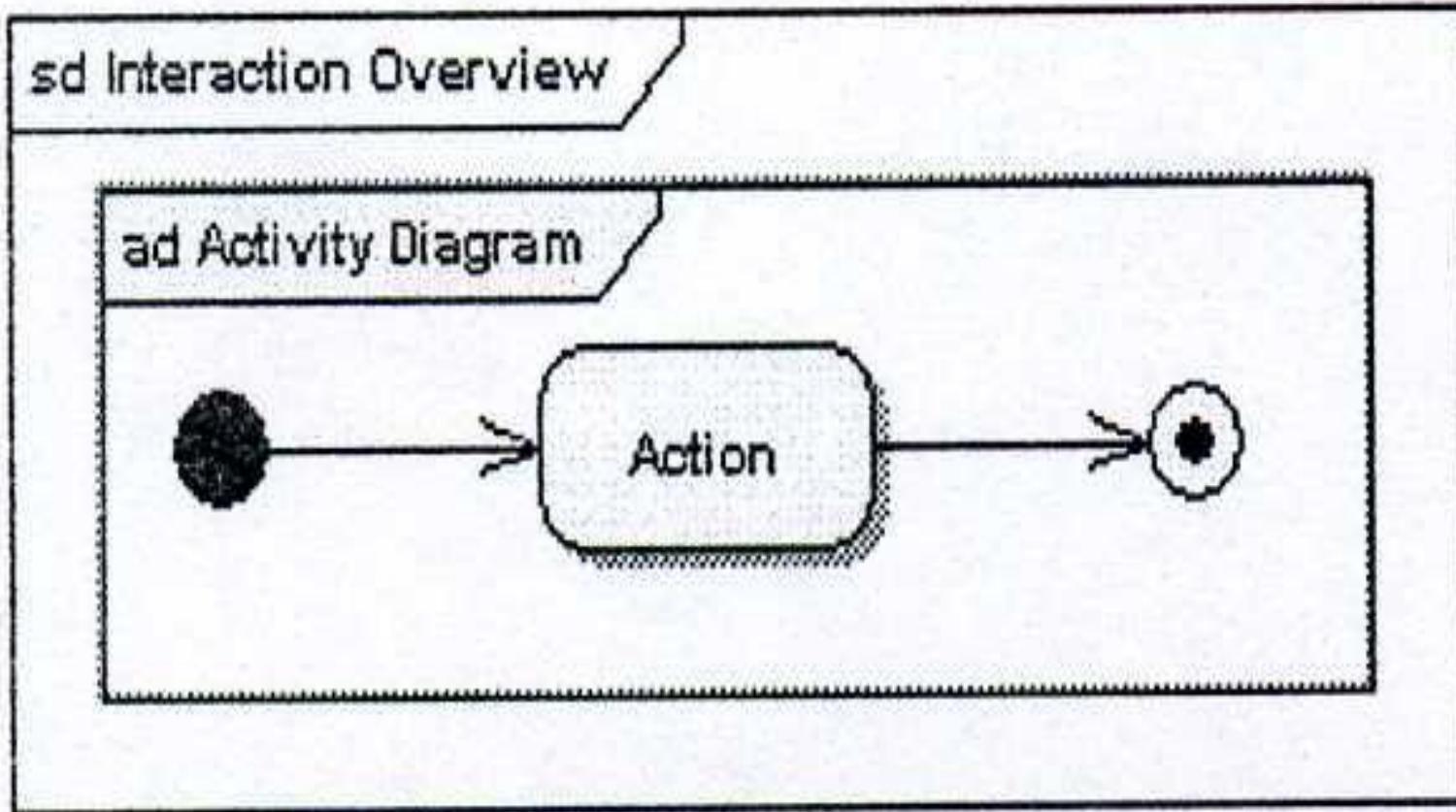
# UML2: kölcsönhatás diagram

- A kölcsönhatás (interaction overview diagram) elemei:
  - kölcsönhatás előfordulás (interaction occurrence),
  - kölcsönhatás elemei (interaction elements),
  - A fenti kettő egyesítése (putting it all together).

# Kölcsönhatás előfordulás (interaction occurrence)



# kölcsönhatás elemei (interaction elements)



# A fenti kettő egyesítése

