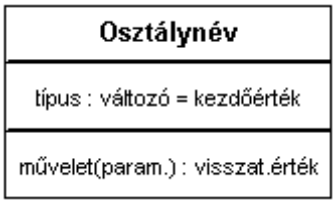
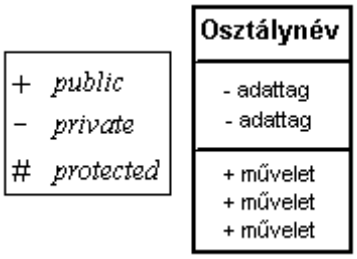
Class Diagram

Leírja a rendszer statikus szerkezetét. Egy osztály az entitás absztrakciója, általános jellemzőkkel. Az asszociációk az osztályok közötti viszonyokat jelentik.

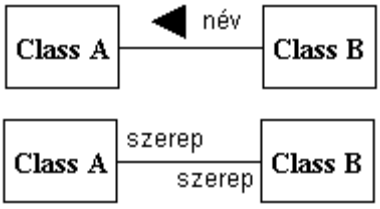


Az osztályok jelölése egy téglalap, ami három részre van osztva:
- Az osztály neve
- Adattagok (Változók)
- Műveletek (Függvények)



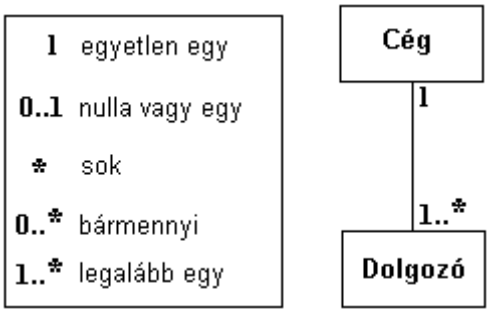
A láthatóságnak (Visibility) három fajtája van:
- Csak class-on belül (private)
- Bárhonnan látható (public)
- Public öröklléssel is private marad (protected)

Az asszociációt két osztály között egy őket összekötő vonallal jelöljük. A viszony irányát teli nyíllal jelölhetjük, vagy a vonal végéhez odaírhatjuk a szerepet. A szerep azt jelöli, hogy az osztályok miként látják egymást. A jelölést egyszerre nem szokás használni.

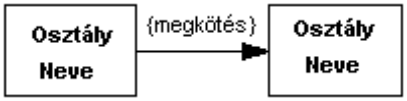


egy

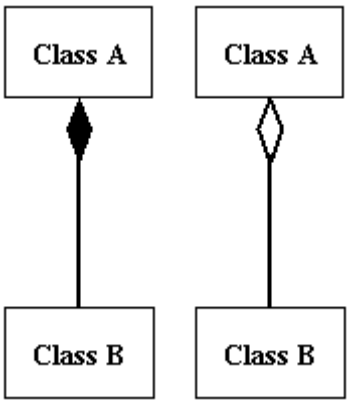
két



Az asszociációk végeihez elhelyezhetünk jelöléseket arra utalóan, hogy az adott osztály a másikkal hány kapcsolatban áll. Például a cégnek több alkalmazottja is lehet, de az adott dolgozó csak egy cégnél dolgozik.



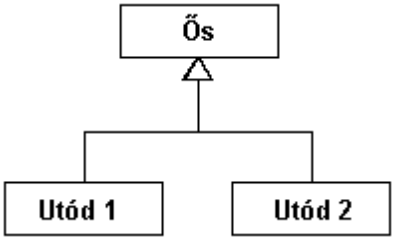
Egyszerű megkötés

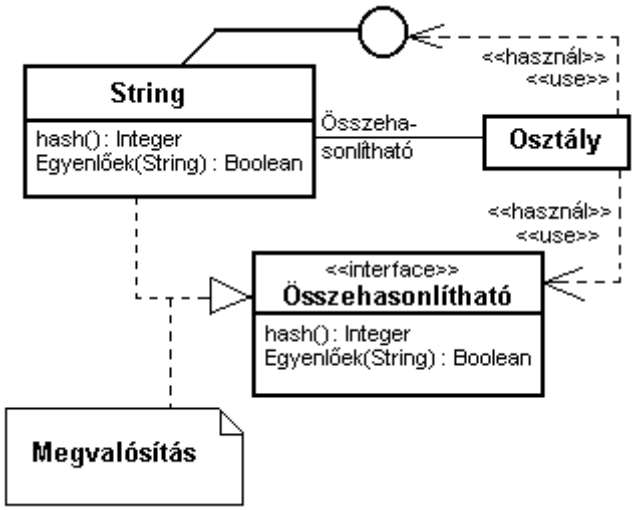


Bonyolult megkötés
Az asszociációkra vonatkozó megkötéseket kapcsos zárójelbe rakjuk.

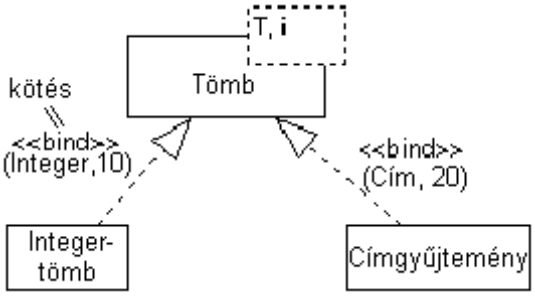
A teli rombusz egy erősebb megkötést jelent. A osztály B-ből épül fel, s rajta kívül semmiből. (Összetétel, Kompozíció)
Az üres rombusz jelenti, hogy A tartalmazza B-t, de rajta kívül akár más osztályt is. (Aggregáció)

Az öröklést egy üres nyíllal jelöljük. Az öröklés következménye, hogy az Ős helyettesíthető bármely Utóddal.
Különbség az aggregációval szemben, hogy így egy adott osztálynak nem csak a public láthatóságú adattagjaihoz férhetünk hozzá.





Az osztály diagramban lehetőség van interface-k leírására is. Az interface azt jelenti, hogy amely osztály őt megvalósítja (implementálja), annak az interface leírásában megadott függvényeket is definiálnia kell. Az ábrán a String implementálja az Összehasonlítható inteface-t.
Az implementálást szaggatott szárú üres nyíllal jelöljük.
A szamárfüles téglalap megjegyzés.

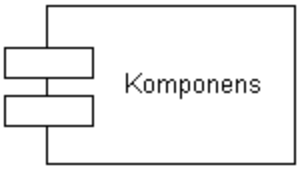


Az ábrán a Tömb egy sablon (Template), ami több különböző osztályra is alkalmazható, és olyan funkciókat valósít meg, amely a hozzá kapcsolódó osztályokat általánosságban kezelni tudja.
A szaggatott vonalas téglalap azt jelzi, hogy az adott sablon vagy osztály milyen adatszerkezeteket kezel. (pl. Integerekből álló verem: osztály = verem, kezelt adatszerkezet = integer)

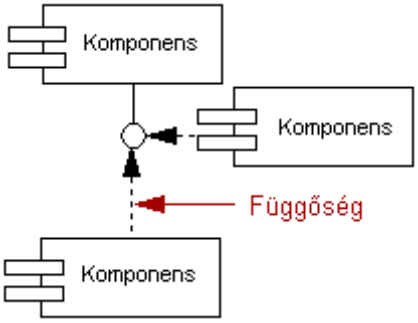
Component Diagram

Leírja a rendszer fizikai komponenseinek szervezését.

A komponens a rendszer fizikai építőeleme.



Az interface olyan függvények halmaza, amelyeket egy adott komponens megvalósít.



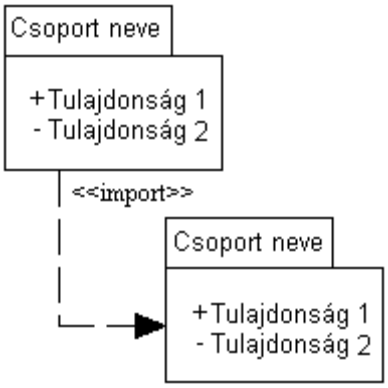
A függőségeket szaggatott szárú, teli nyilakkal jelöljük.

Package Diagram

Csoportokba rendezi a rendszer elemeit, hogy minimalizálja köztük az asszociációkat.

A csoportokat (Package) füles mappákkal jelöljük. Felsorolhatjuk hozzá a tulajdonságokat is, mint az osztályoknál.

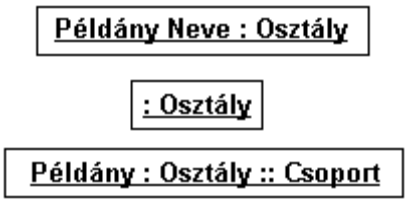
A csoportokra értelmezett láthatóság az osztályoknál megismertek.



A csoportok közötti függőséget egy szaggatott vonalas, teli nyíllaj jelöljük. Az import a függőség speciális esete, amikor is garantáljuk egyik csoport számára, hogy az láthassa a másik tartalmát.

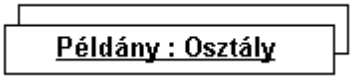
Object Diagram

Szintén szorosan kapcsolatos az osztály diagrammal. Az osztályokat példányosítjuk. (Példány = Object) Ez a diagram a rendszer statikus felépítését figyeli meg egy adoot részidőre, segítségével megállapíthatjuk az osztály diagram pontosságát.

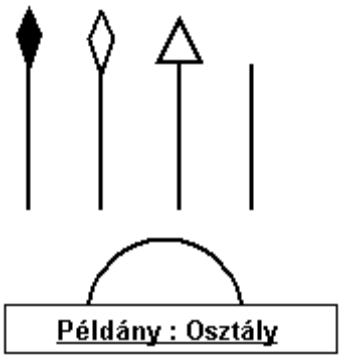


Létrehozhatunk név nélkül példányokat, elláthatjuk névvel, sőt, előfordulhat, hogy az osztály egy csoport tagja. Fontos, hogy a példány nevét alá kell húzni.

A példányoknál is lehetőség van az adattagok felsorolására, de ellentétben osztályokkat itt konkrét értéket kell nekik adni.



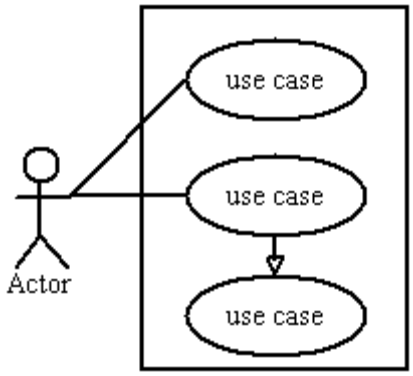
Több példányt is jelölhetünk egyszerre, ha az adattagok nem különbözőek.



Az asszociációk az osztály diagramban megszokottak, leszámítva, hogy az adott példány önmagához is kapcsolódhat.
Ez utóbbira példa: Legyen Mark a CégFőnök osztály tagja, aki egyszerre két cég főnöke. Ha ez a két cég üzletet kötnek egymással, Mark asszociációba kerül önmagával.

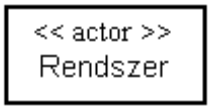
Use-Case Diagram

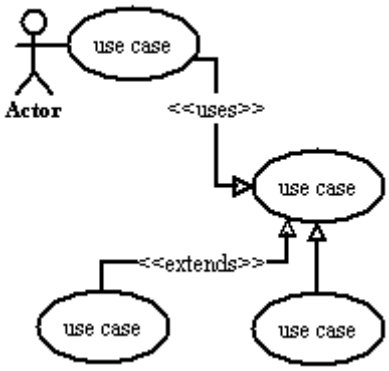
Ez a modell jellemzi a rendszer működését a felhasználó (Actor) és a felé irányuló szolgáltatások illetve funkciók (Use Case) között.



A rendszert egy téglalappal kerítjük körbe, a felhasználó ezen kívül helyezkedik el. A felhasználókat általában pálcikaemberekkel jelöljük, a use case pedig egy ovál, minek címkéje a funkció/szolgáltatás, amit megvalósít.

Egy adott rendszer lehet egy másik felhasználója. Ekkor pálcikaember helyett téglalappal jelöljük az actor-t.

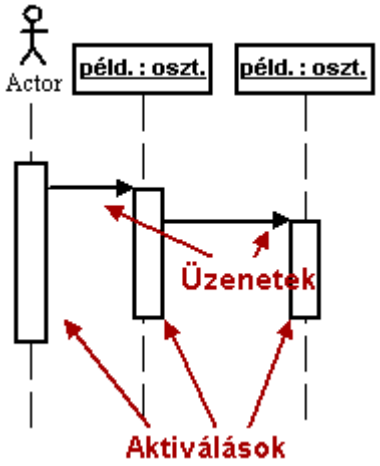




A különböző relációk közt lehet olyan eset is, mikor az egyik use case meg kell hogy hívja a másikat, hogy véghezvihesse a hozzá rendelt taskot (uses), illetve hogy az egyik use case a masikban valszthato lehetőségként szerepel (extends).

Sequence Diagram

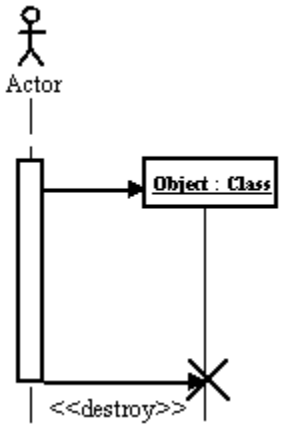
Az osztályok közötti időbeli cselekményt írja le.



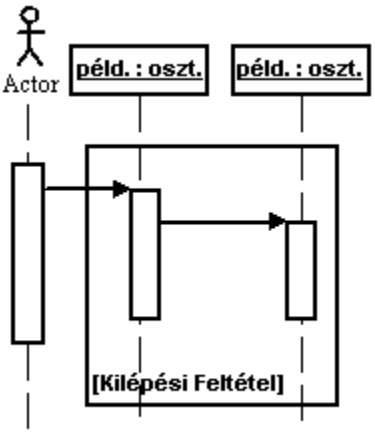
A rajzolokon adott felhasználókat, példányokat (adattagok nélkül) és életvonalukat tűntetjük fel. A folyamatokat, pontosabban azok hosszát az életvonalon vékony téglalappal jelölünk.

Nyílak	Jelentésük
→	Egyszerű
→	Szinkron
→	Aszinkron
↩	Kikerül
◷→	Időt kér

Különböző üzenetküldési módok léteznek. Aszinkron esetben az üzenetet küldő task befejeződése előtt nem várja meg az elindított folyamat válaszát.

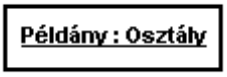


Létrehozhatunk példányokat, illetve el is pusztíthatjuk őket (destroy).

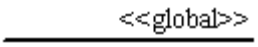


Az esetleges ciklust téglalappal jelöljük, aminek bal alsó sarkába, szögeletes zárójelbe írhatjuk a kilépési feltételét.

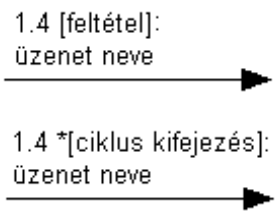
Collaboration Diagram



A példányokat a szokott módon jelöljük.



Megadhatjuk, hogy az asszociációk miként viselkednek adott részszituációkban. Egy példány szerepét az asszociációhoz írt címképevel adhatjuk meg.



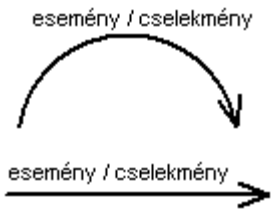
A kollaborációs diagram - ellentétben a szekvenciális diagrammal - nem időt kezel, hanem üzenetsorrendet. Ezt számozással érjük el, pl.: az első üzenetcsoportbeli elemek számozása 1.1 1.2 1.3 ...

Az üzenetek kiadása lehet feltételtől függő vagy ciklikus. A megfelelő kifejezést szögletes zárójelbe tesszük. A ciklust az különbözteti meg a feltételtől, hogy a számozás után csillag szerepel.

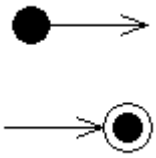
Statechart Diagram

Osztályok viselkedését figyeli külső hatások következtében. A rendszer állapotról állapotra folyó adatáramlását modellezi.

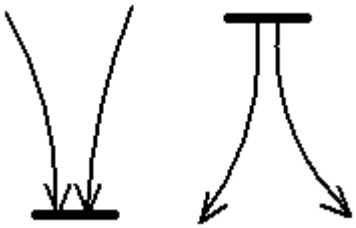
Az állapot a példány életében egy szituációt jelent.



Az asszociációk egy-egy állapotváltást jelölnek. Feltüntetjük rajta ugyanakkor az állapotváltást kiváltó eseménnyel párosuló cselekményt is.

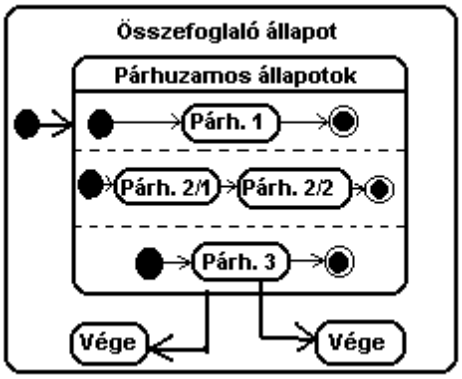


Kezdőállapotból csak egy lehet, végállapotból több is.

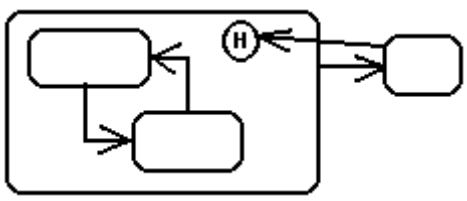


A vezérlést szinkronizálni, valamint állapotok közt megosztni tudjuk.
A szinkronizálás azt jelenti, hogy a következő állapotba lépés feltétele mindkét vezérlőjel megérkezése.

Az állapotoknak megadhatunk több jellemzőt is:
&nbps; - Adattagok értékének megváltoztatása
- Állapotba érkezéskor végrehajtódó funkciók (entry)
- Állapotban tartózkodáskor folyamatosan végrehajtódó funkciók (do)
- Állapot elhagyásakor végrehajtandó funkciók (exit)
- Ségítség-információ



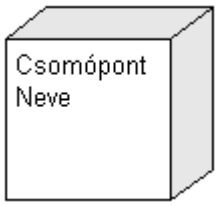
Egyes állapotokat kinagyíthatunk (az Összefoglaló állapot), valamint párhuzamosíthatjuk őket. A párhuzamosan folyó feladatok állapotából két féleképpen lehet kilépni:
- Egy belső állapotból kilépünk egy külsőbe.
- Az összes párhuzamos folyamat végállapotba került.



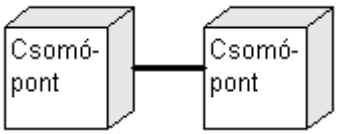
A History Indicator (jelölése: H betű egy körben) megjegyzi, az adott állapotból milyen alállapot lévén kerültünk ki. Ha visszalépéskor a History Indicatorba lépünk, ebbe az alállpotba kerülünk.

Deployment Diagram

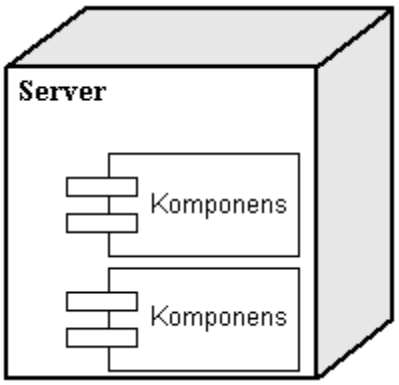
Leírást ad a rendszerről csomópontok, összeköttetések és komponensek formájában.



A csomópont egy fizikai forrás, ami kódrészeket hajt végre.



Az asszociációk fizikai összeköttetések. pl.: Ethernet



A komponenseket az őket felvonultató csomópontban lehet megjeleníteni.