

Szoftvertchnológia és -technikák

5. Előadás

Használati eset diagram és kitekintés



I. Házi feladat

- A tanszéki weboldalon megtalálható jövő hét elejétől
- Modellezési feladat
 - > Nem kell kódot írni
- Szabad specifikáció
- Beadás: portálon keresztül
- Beadási határidő: 10. hét péntek (12:00)



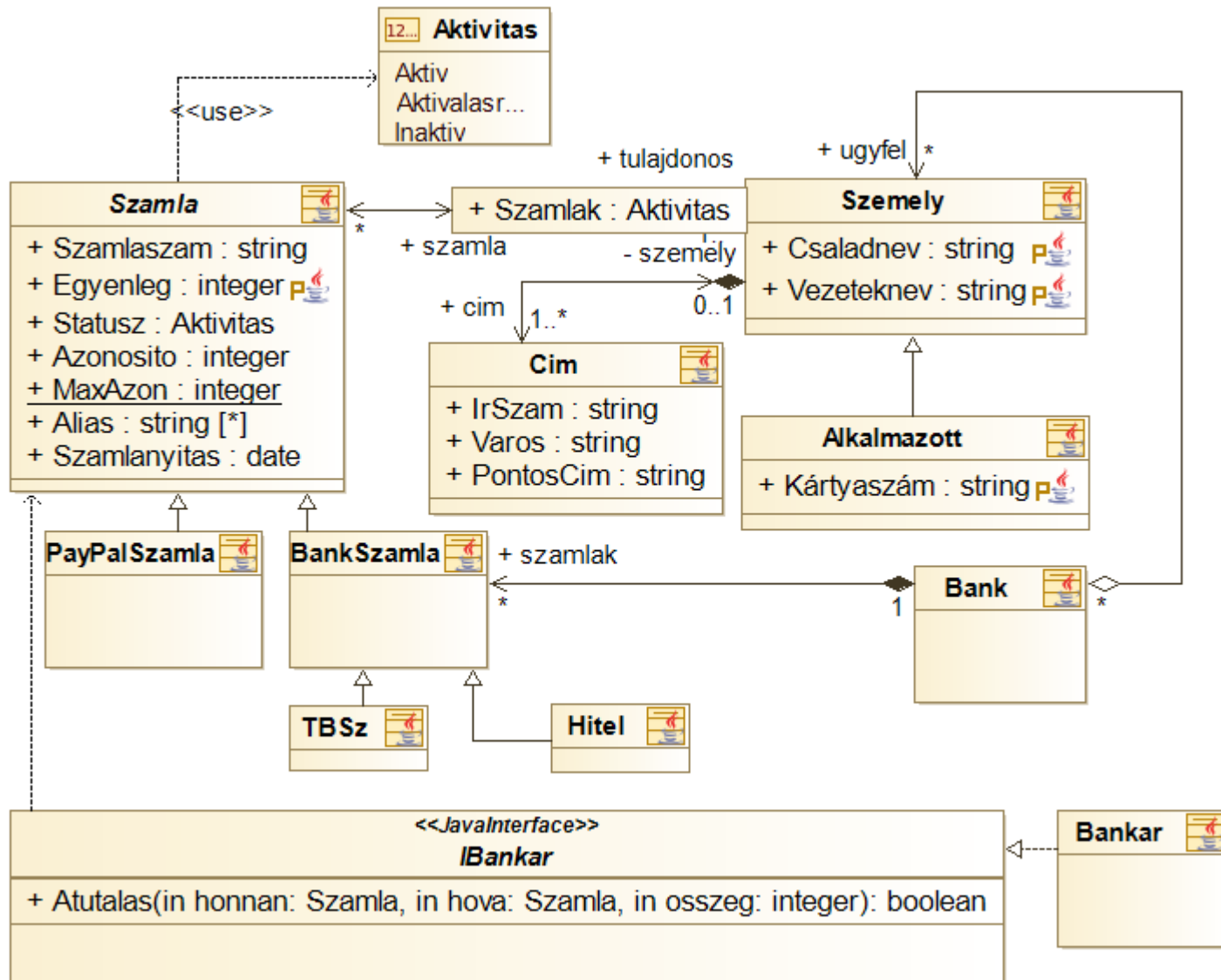
ZH

- Jelenléti ZH!
 - > Csak kedvezményes tanulmányi rendi kérvénnyel (022) lehet online teljesíteni!
- 2020. nov. 06. 08:15-
- Lesz minta ZH
- Számonkért anyag: ZH előtt 1-2 héttel

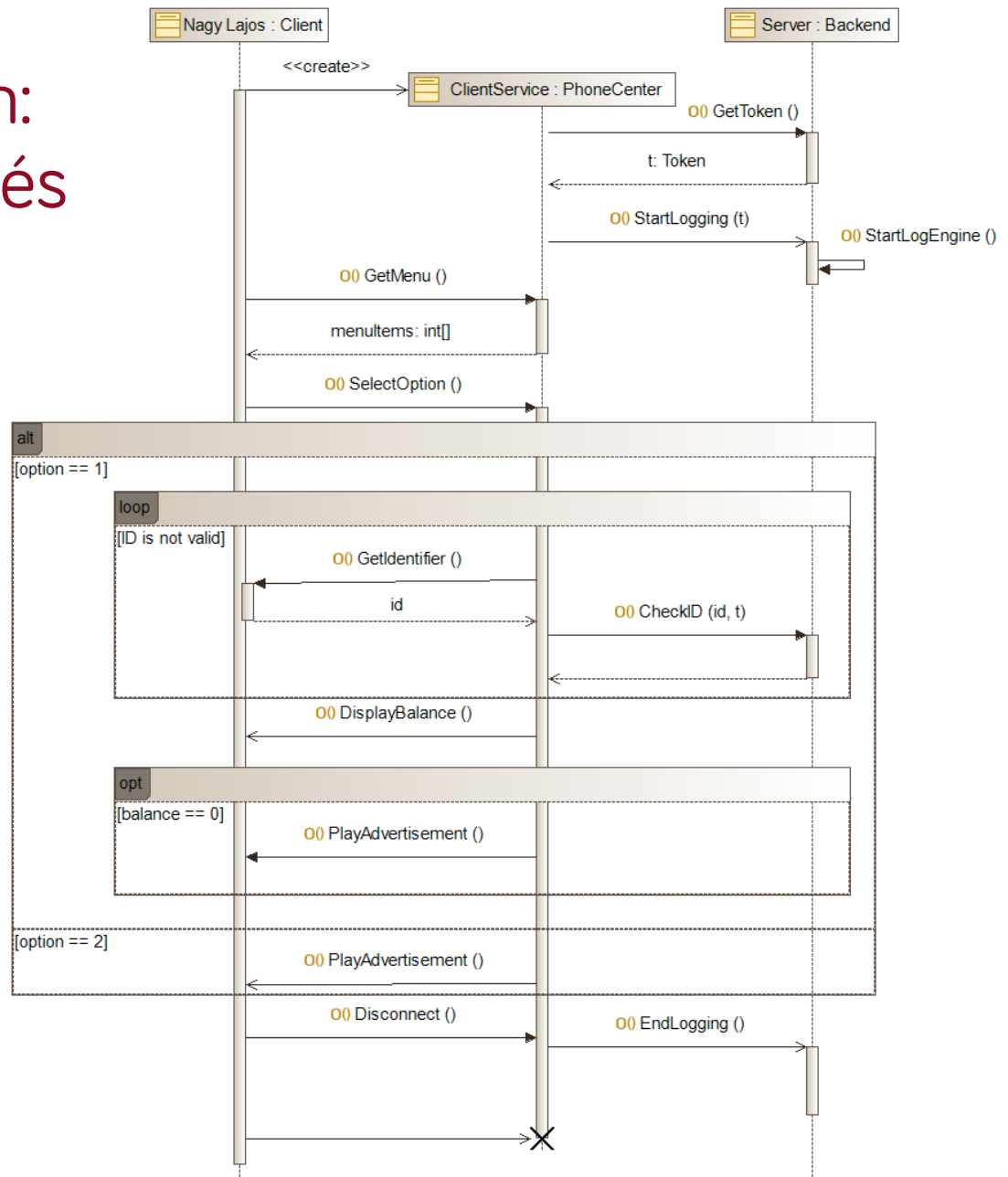


UML diagramok

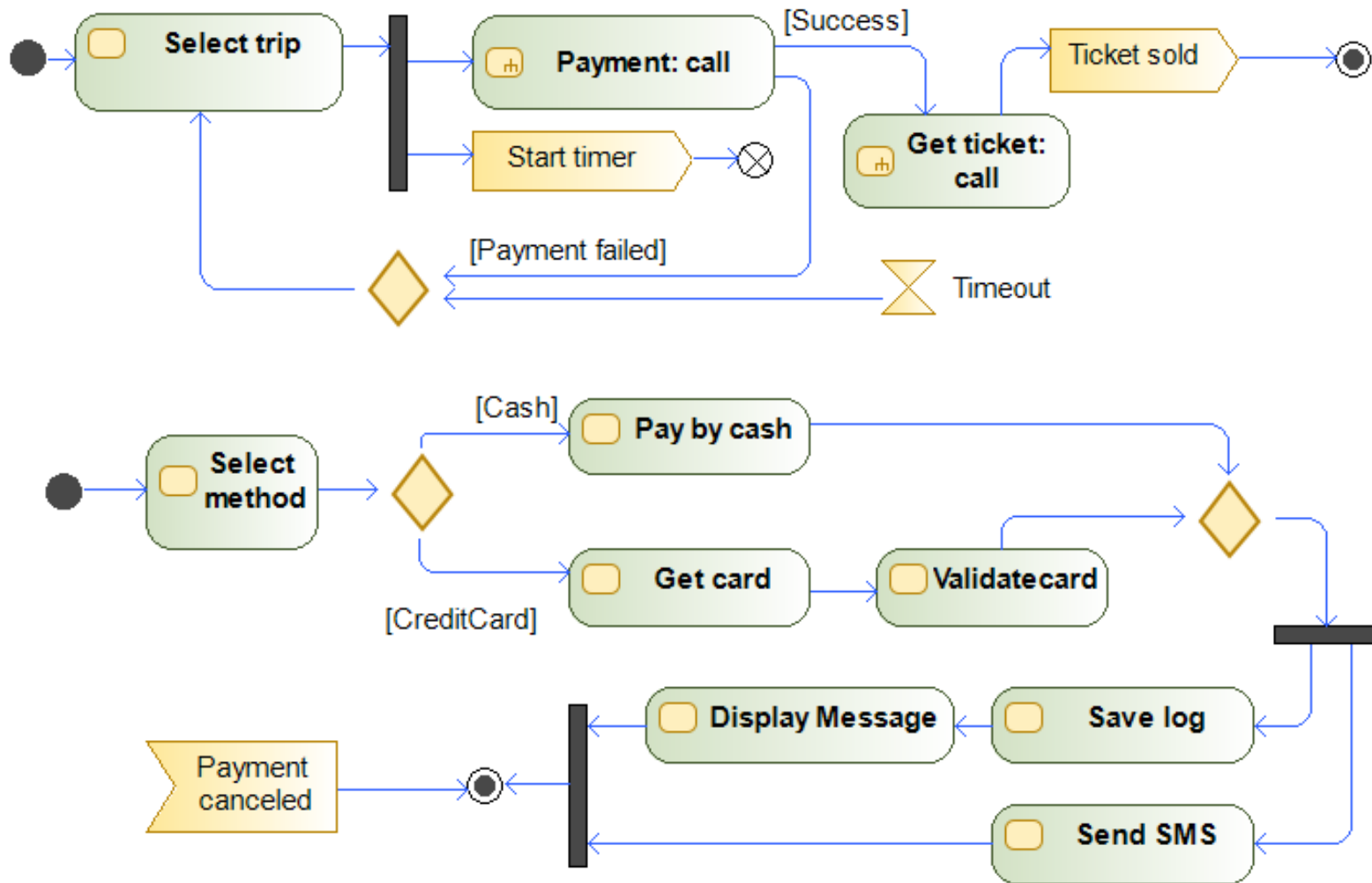
Osztálydiagram: Banki adminisztrációs rendszer



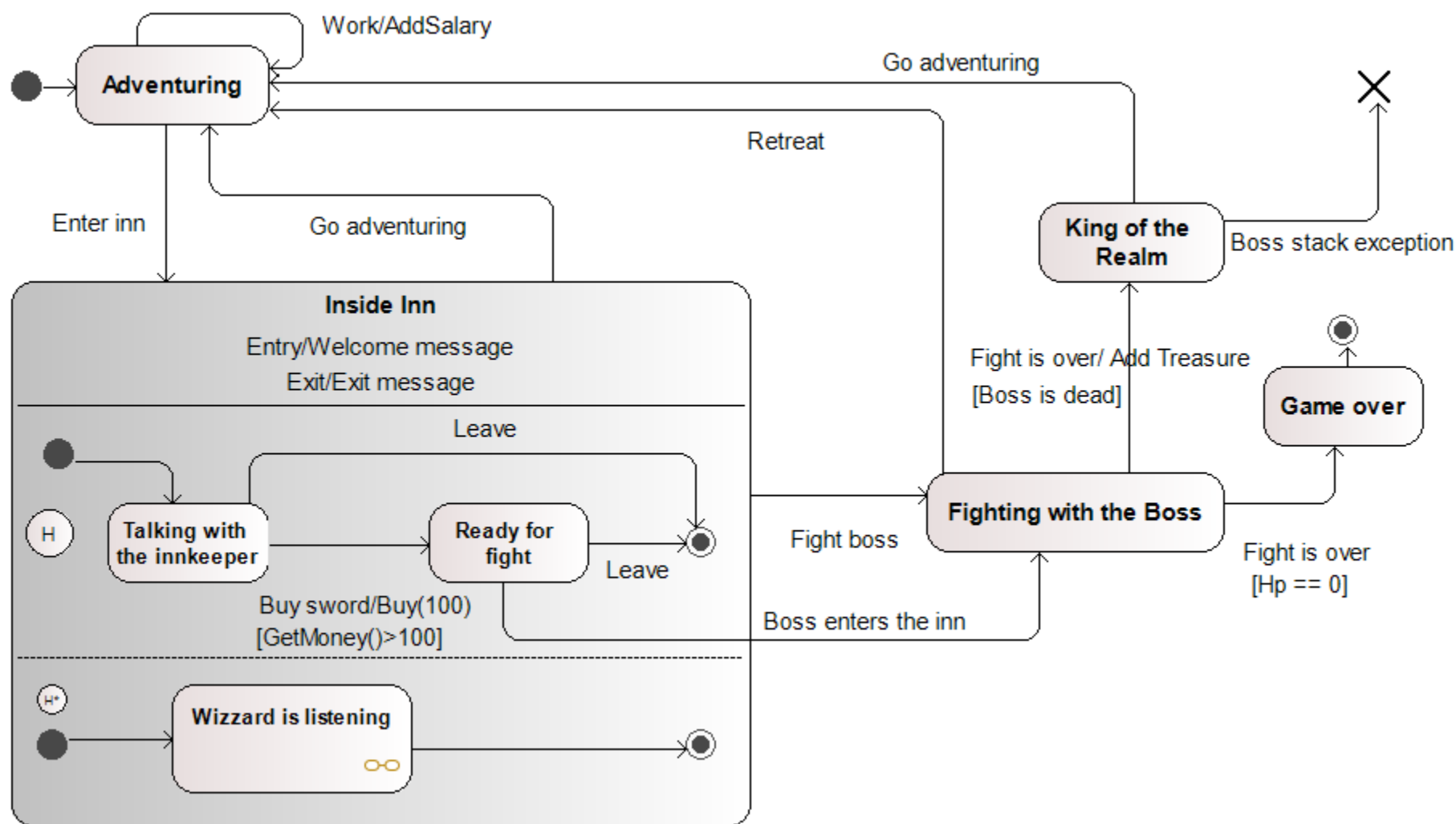
Szekvenciadiagram: Telefonos ügyintézés



Aktivitásdiagram: BKV jegyvásárlás



Állapotgép: Kalandjáték



UML diagramok

- A kódstruktúra: osztálydiagram
- A hívások sorrendje: szekvenciadiagram
- A folyamatok: aktivitásdiagram
- Az állapotátmenetek: állapotgép
- ... de hogyan fogjuk össze a teljes rendszert?



Használati eset diagram

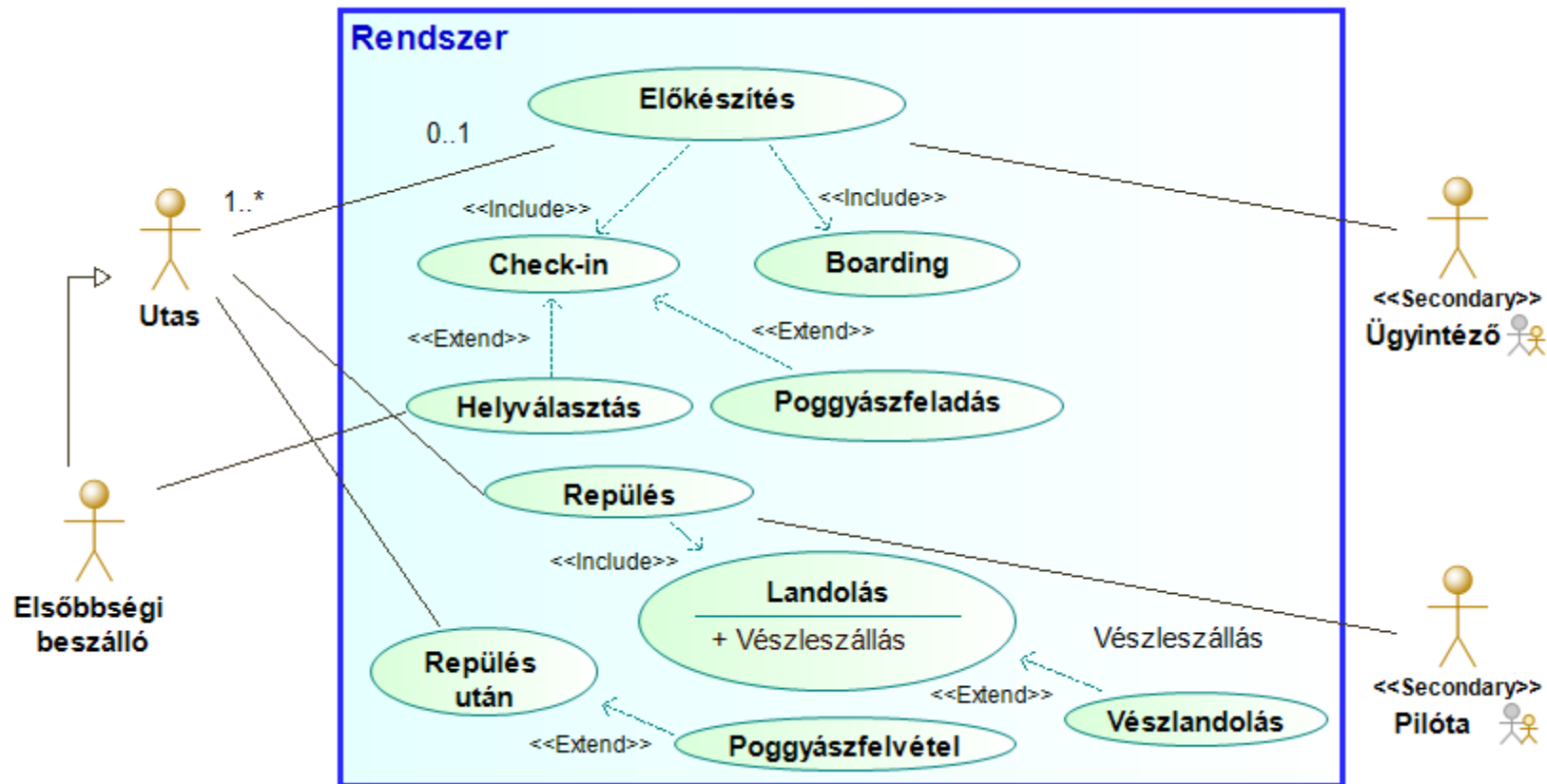
Use case diagram

Használati eset diagram

- A teljes rendszer áttekintése
 - > Mi tartozik bele? Mi az, ami nem?
 - > Kik a szereplők?
 - > Milyen funkciókat kell megvalósítani?
- **Használati eset diagram**
 - > Rendszerszintű áttekintés
 - > Funkcionális követelmények
 - > Kommunikáció a megrendelővel

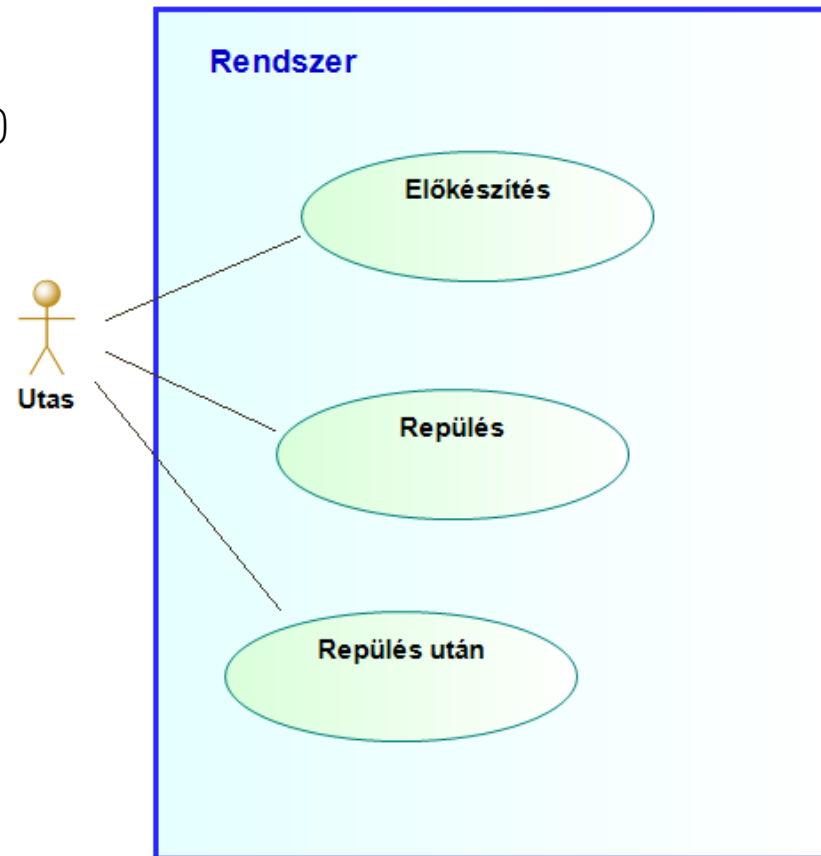


Használati eset diagram: Repülés



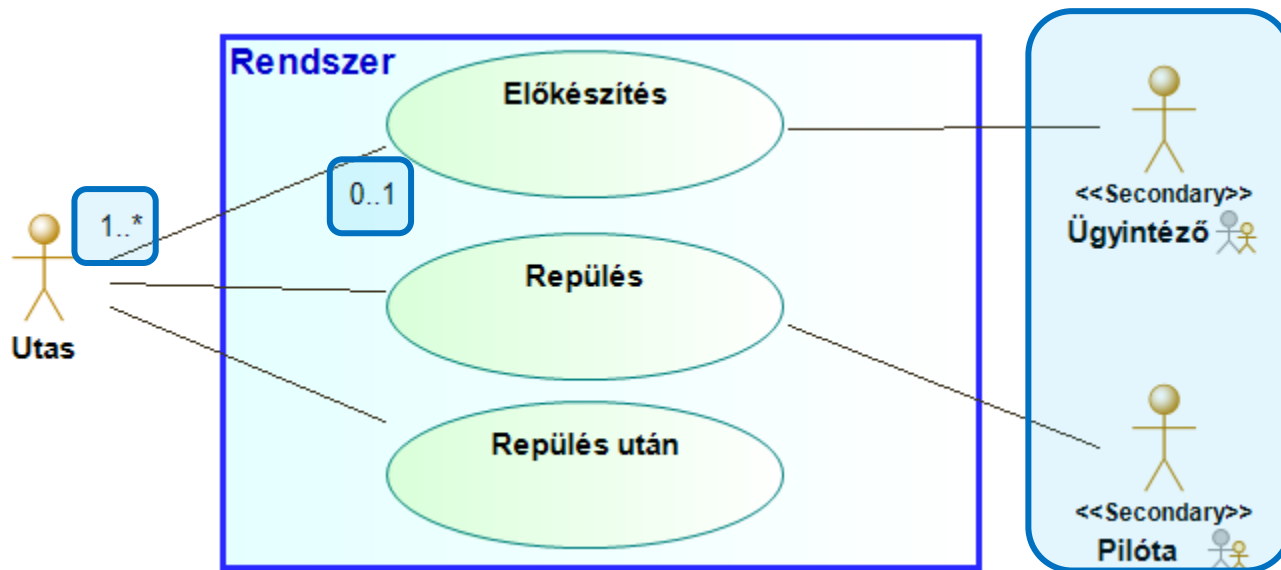
Funkciók

- A repülés három tevékenységből áll: előkészítés, repülés és befejezés. A főszereplő az utas.
 - > **Rendszer határ (system boundary)**
(a Modelio közvetlenül nem támogatja a jelölést)
 - > **Aktor (actor)**
 - Egy adott szerepkör (ember, gép, folyamat, ...)
 - > **Használati eset (use case)**
 - Actor – rendszer interakció reprezentálása
 - > **Asszociáció (association)**



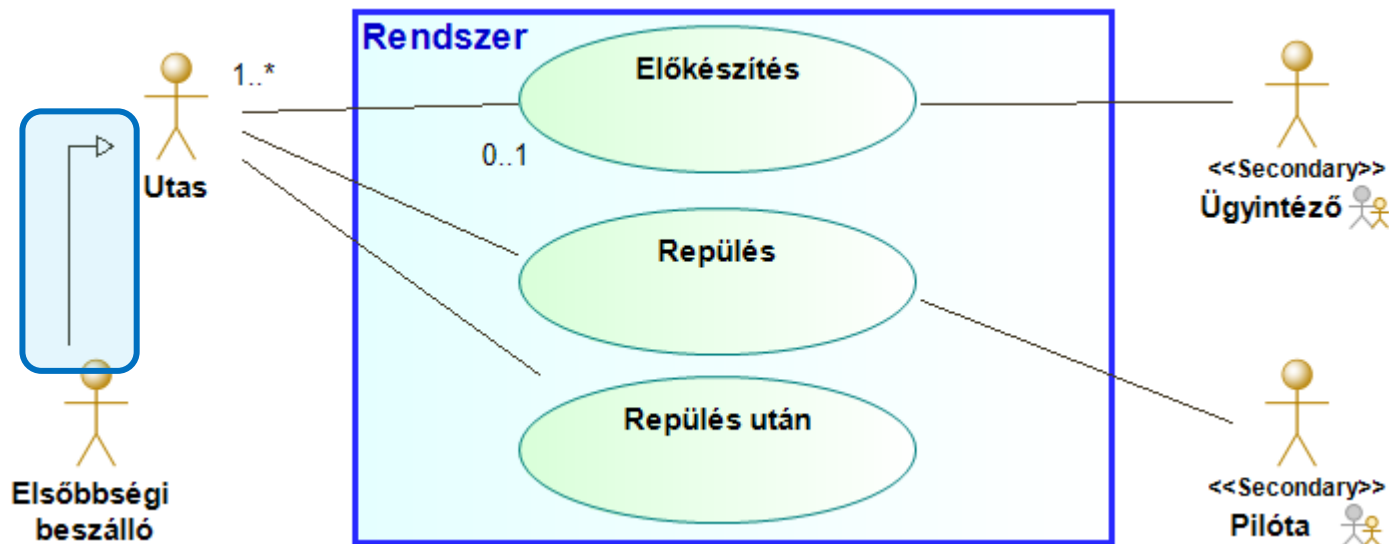
További szereplők

- A reptéri ügyintéző és a pilóta is fontos szereplő
 - > Más-más funkciókban kapnak szerepet
 - Elsődleges (primary) és másodlagos (secondary) aktorok
 - Sztereotípiával jelölt
 - > Multiplicitás megadható
 - Alapesetben 1..1



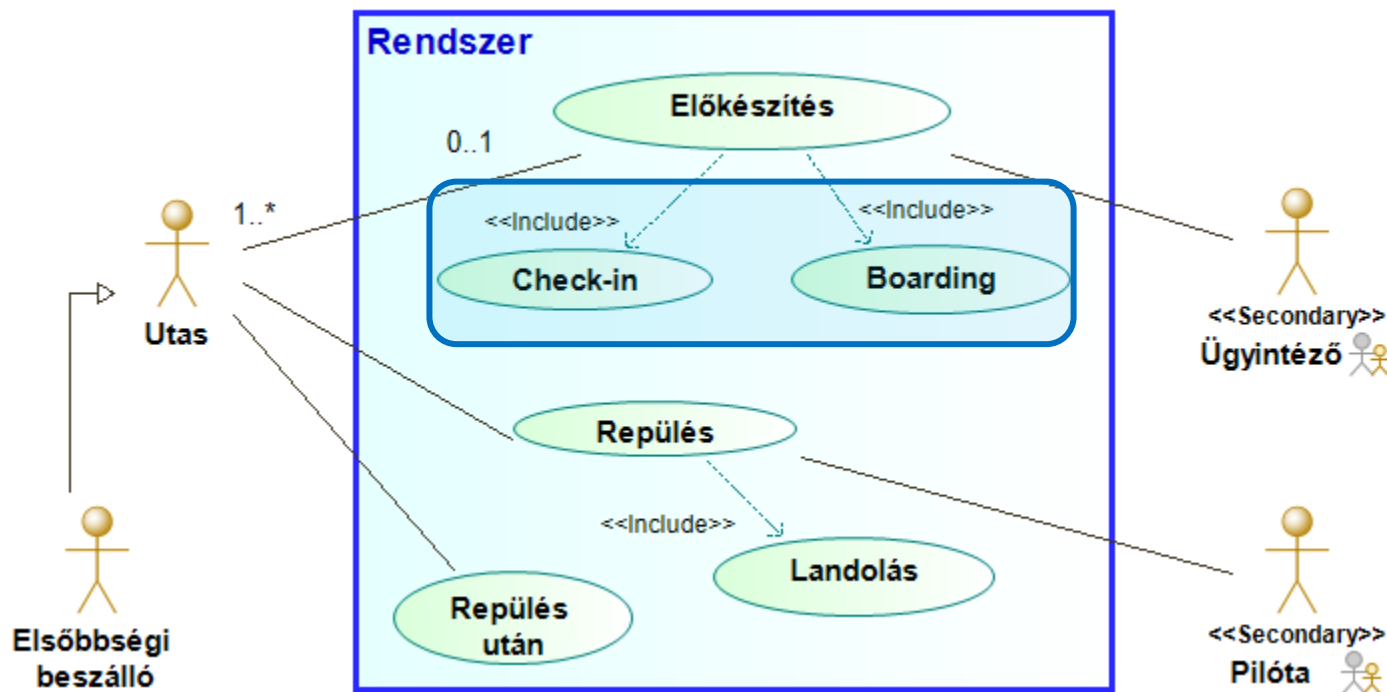
További szereplők

- Vannak elsőbbségi utasok
 - Mindent tehetnek, amit a többi utas, de jár nekik extra szolgáltatás is (l. később).
 - Általánosítás (generalization)
 - Aktorok és használati esetek esetében is meg lehet adni



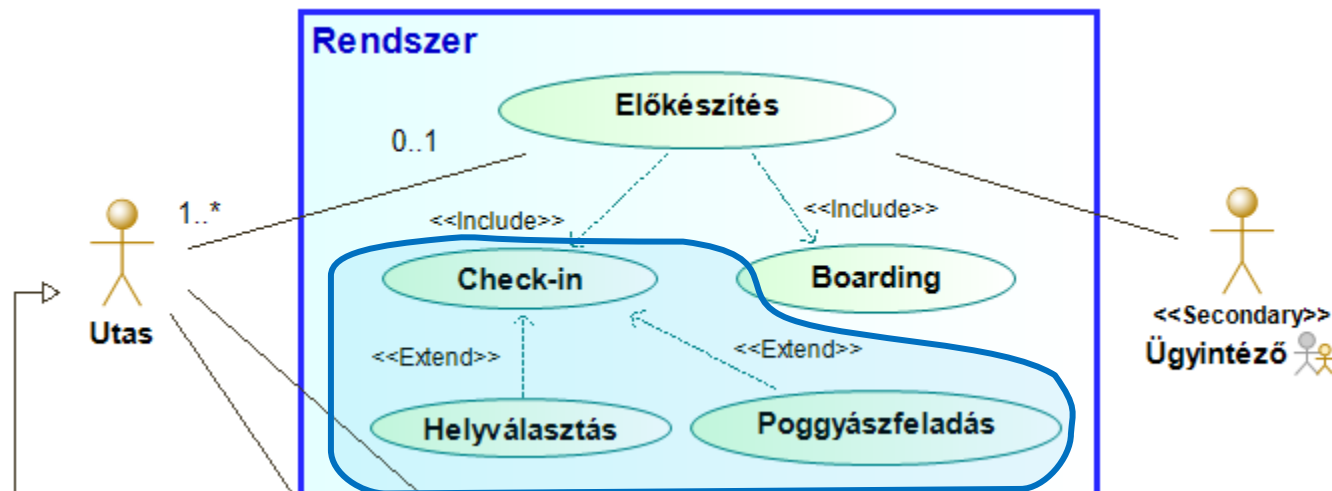
Részfunkciók

- Az előkészítés része a check-in és a boarding, a repülés része a landolás
 - > Tartalmazás (include) sztereotípa
 - > Részfunkció, rész use case (kompozíció)
 - > A tartalmazó use-case asszociációi a tartalmazottra is vonatkoznak



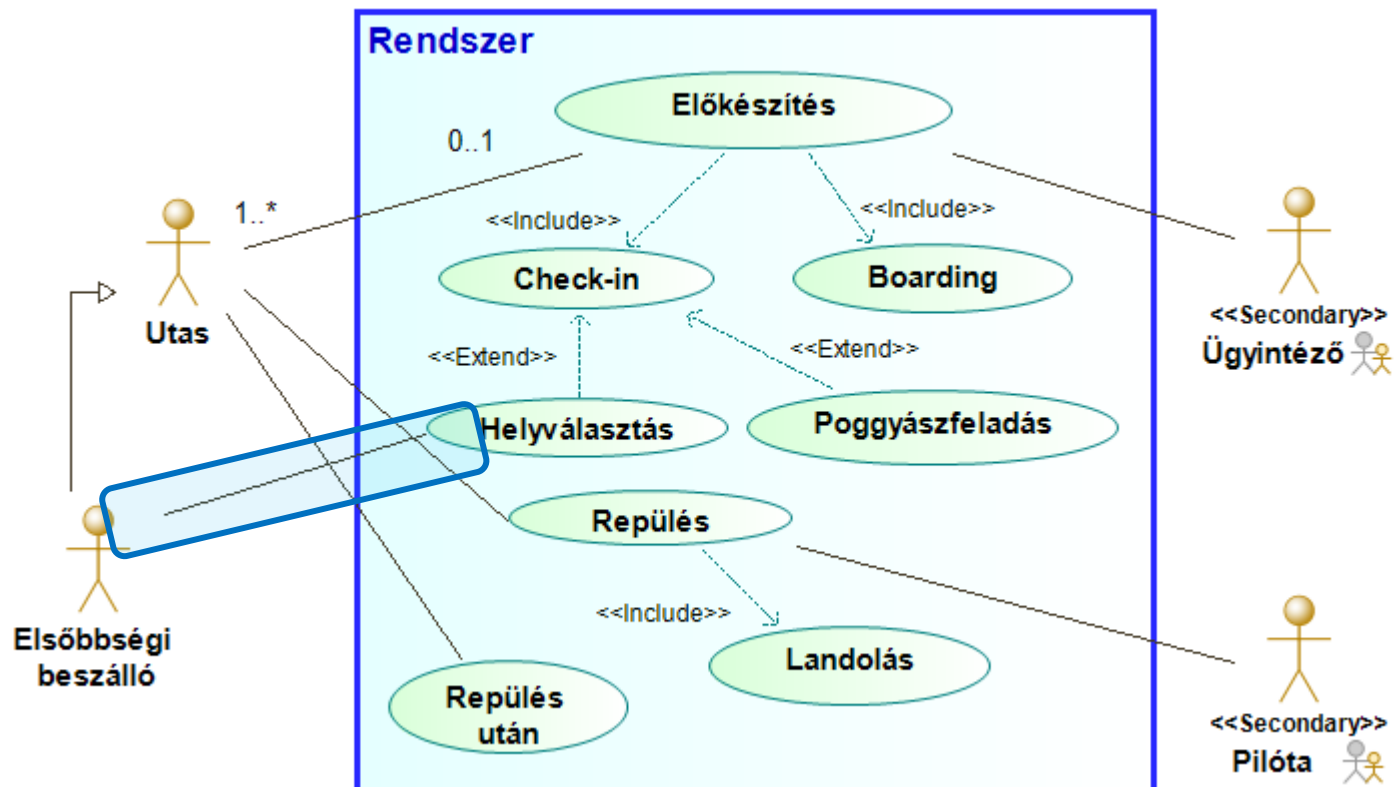
Kiegészítések

- A check-in folyamat néha kiegészül a poggyászfeladás és a helyválasztás folyamatával
 - > Nem mindig tartalmazza a funkciót, de a funkció önmagában nem értelmezett
 - > **Kiegészítés (extend)** sztereotípia
 - > Include vs extend vs inheritance
 - „Étteremben mindig fizetünk a végén” - include
 - „Étteremben fizetéskor néha adunk borralalót” - extend
 - „Két fajta éttermi fizetés van: kártyás és készpénzes” - inheritance



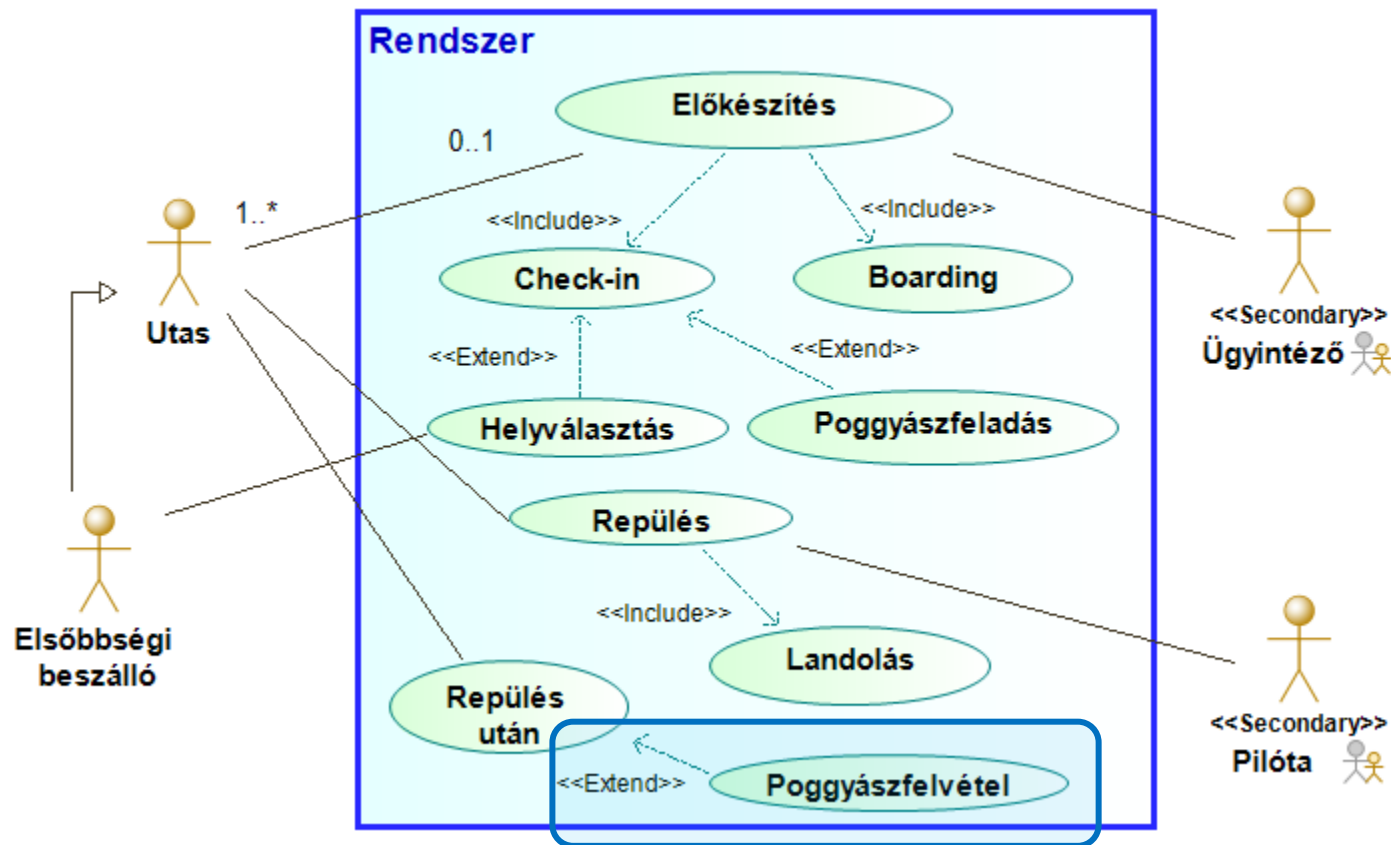
Helyválasztás

- A helyválasztás az elsőbbségi utasok számára elérhető
 - > Nem feltétlenül „öröklődik” úgy, mint az include esetén, külön jelöljük, hogy kik számára elérhető



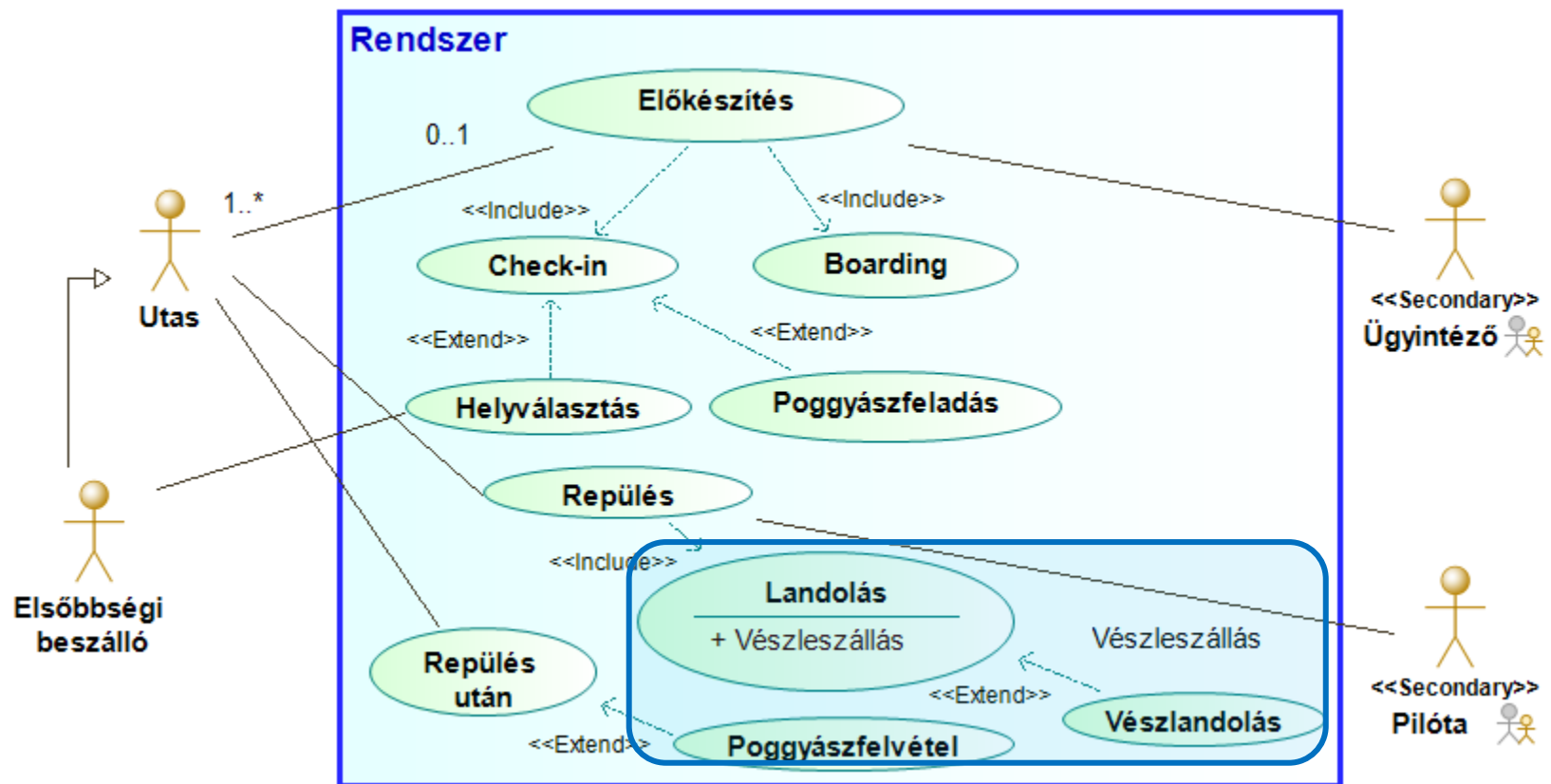
Poggyászfelvétel

- A poggyászfelvétel kiegészíthető a repülés utáni teendőket



Vészlandolás

- A landolás speciális esete a vészlandolás
 - > Kiterjesztési pont (extension point)



Forgatókönyv

- Szöveges forgatókönyv írható a diagramból
 1. Előkészítés
 - 1.1. Check-in elvégzése
 - 1.1.A. Helyválasztás
 - 1.1.B. Poggyászfeladás
 - 1.2. Boarding
 2. Repülés
 - 2.1. Landolás
 - 2.1.A. Vészleszállás
 3. Repülés után
 - 3.A. Poggyászfelvétel
- Nem része az UML szabványnak

Használati eset diagram

- Rendszer
- Aktor
- Használati eset (use-case)
- Asszociáció
- Általánosítás
- Tartalmazás (include)
- Kiterjesztés (extend)

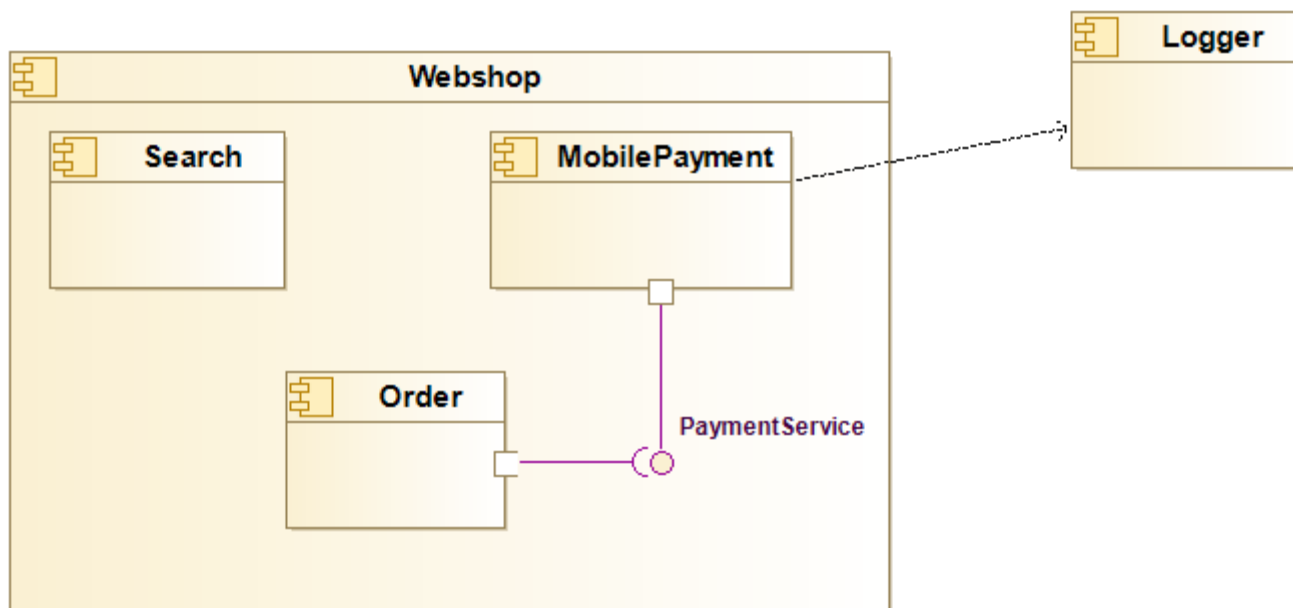
UML – diagramok

UML diagramok

- Rendszer modellezése
 1. Használati eset diagram
 - Követelmények, funkciók, résztvevők
 2. Aktivitás diagram
 - Folyamatok
 3. Állapotgép
 - A rendszer állapotai
 4. Osztálydiagram
 - Kódstruktúra
 5. Szekvencia diagram
 - Függvényhívások

További diagramok: komponens diagram

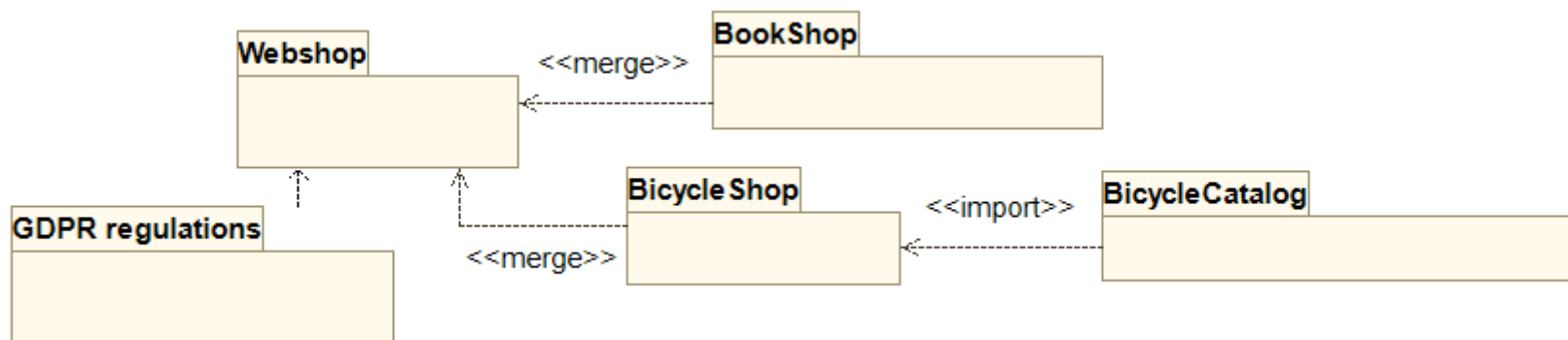
- Komponens diagram
 - > Komponens: összefüggő osztályok
 - > Hierarchikus felépítés
 - > Komponensek és függéseik
 - > Interfészek (biztosított ill. elvárt)



További diagrammok: csomagdiagram

- Csomagdiagram

- > Csomag: összefüggő információk (nem csak kód!)
- > Rendszer felbontása modell szinten
- > Csomagok közti függőségek ábrázolása
 - Függőség (dependency) – nem meghatározott kapcsolat
 - Importálás (include, import, using, ...) – újra felhasználás
 - Összevonás (merge) – egyesítés



UML diagramok

- Osztálydiagram (Class diagram)
- Szekvenciadiagram (Sequence diagram)
- Aktivitásdiagram (Activity diagram)
- Állapotgép (State machine)
- Használati eset diagram (Use case diagram)
- *Komponensdiagram (Component diagram)*
- *Csomagdiagram (Package diagram)*
- Objektumdiagram (Object diagram)
- Telepítési diagram (Deployment diagram)
- Összetett struktúradiagram (Composite Structure diagram)
- Profildiagram (Profile diagram)
- Kommunikációs diagram (Communication diagram)
- Interakció áttekintő diagram (Interaction overview diagram)
- Időzítő diagram (Timing diagram)

UML - Kényszerek

Kényszerek

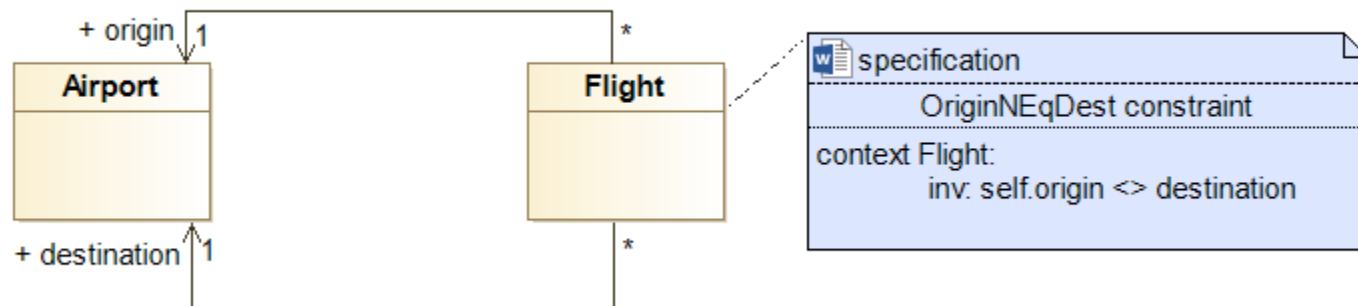
- UML: alapvetően grafikus leírás/modellek
- Grafikus leírás
 - > Könnyen átlátható
 - > Könnyen tanulható (megrendelő is érthetik)
 - > Megkötések/összefüggések nehezen írhatóak le vele
- Pédák
 - > “Mindenki fiatalabb a szüleinél.”
 - > “Senki nem lehet a saját szülője.”
 - > “Minden BME-n végzett üzemmérnök informatikus fizetése nagyobb 300 ezer Ft-nál.”

Kényszerek

- Megoldás: szöveges nyelv a grafikus modellhez illesztve
- Object Constraint Language (OCL)
 - > Deklaratív (“Mit?” a “Hogyan?” helyett)
 - > Programozási nyelvtől független
 - > UML modellekhez illeszkedik (pl. navigáció)
 - > Mellékhatás-mentes végrehajtás (lekérdez, nem változtat)
 - > Kényszertípusok
 - Invariánsok (mindig igaz feltétel)
 - Elő-, utófeltétel
 - Metódustörzs
 - Számított érték
 - Kiindulási/Alapértelmezett érték

Kényszerek

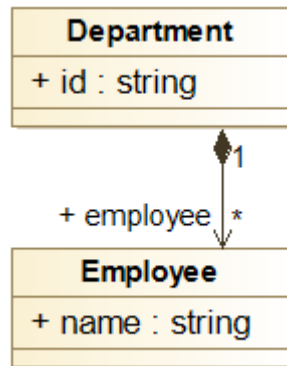
- Repülő esetén a kiindulási és célreptér nem lehet azonos!



UML – közös formátum

XMI

- UML – szabvány a modellező nyelvekhez
 - > Több UML- kompatibilis eszköz létezik
 - > Hogyan lehet adatokat cserélni köztük?
- Megoldás: XML Metadata Interchange
 - > XML-alapú
 - > Szabványos modell-leíró formátum



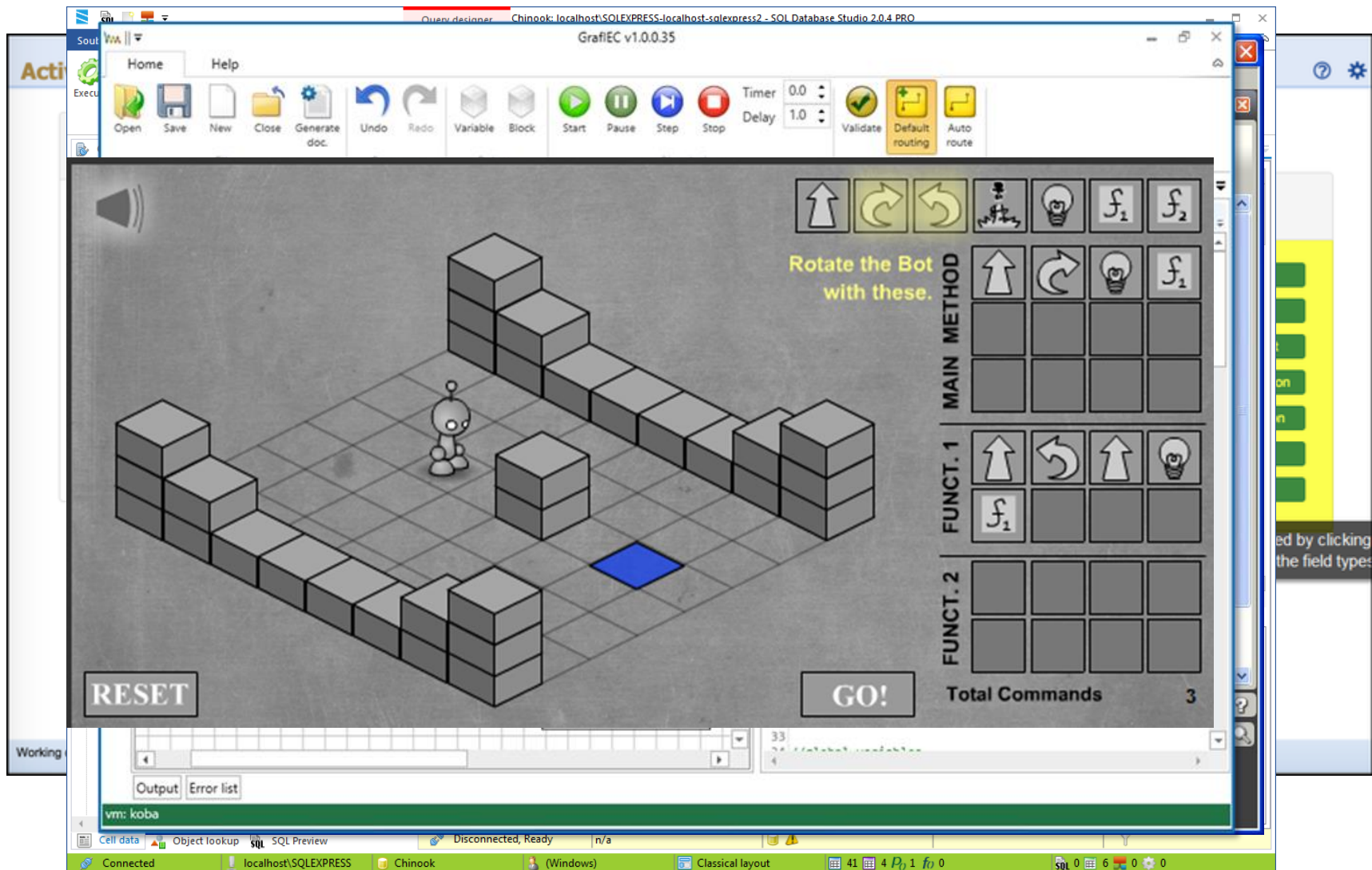
```
<Classifiers xsi:type="Class" name="Department">
  <StructuralFeatures xsi:type="Attribute" name="id" Type="String"/>
  <StructuralFeatures xsi:type="Reference" name="member" upperBound="-1"
    Type="#//Employee" containment="true"/>
</Classifiers>
<Classifiers xsi:type="Class" name="Employee">
  <StructuralFeatures xsi:type="Attribute" name="name" Type="String"/>
</Classifiers>
```

Túl az UML-en?

UML – egy nyelv(család) mind felett?

- Mire jó az UML?
 - > Szabványos
 - > Szoftvermérnökök közti közös nyelv
 - > Struktúrák, folyamatok ábrázolása
 - > Követelmények rögzítése a megrendelő felé
- Mire nem jó az UML?
 - > Nem szoftveres projektek
 - > Egyedi jelölések
 - > Szakterület-függő szabályok

Nem UML, mégis modell



UML?

- Az UML
 - > Elterjedt (mindenki ismeri a szoftver világban)
 - > Általános (minden modellezhető vele)
 - > Támogatott (rengeteg eszköz van hozzá)
 - > De nem az egyetlen modellező nyelv!



Szakterületi nyelvek

- Szakterületi nyelv
 - > Speciális nyelv egy adott szakterületre
 - > Korlátozott elemkészlet
 - > Erős, specializált szabályok és jelölés
 - > Egy adott termék(családhoz) készül
 - > Egyedi modellezőeszközök

UML vs. szakterületi nyelvek

UML

- Egységes, szabványos
- Minden szoftver-mérnök ismeri
- Jó eszköztámogatás
- Korlátozott kódgenerálás
- Megrendelő számára érthetetlen

Szakterületi nyelvek

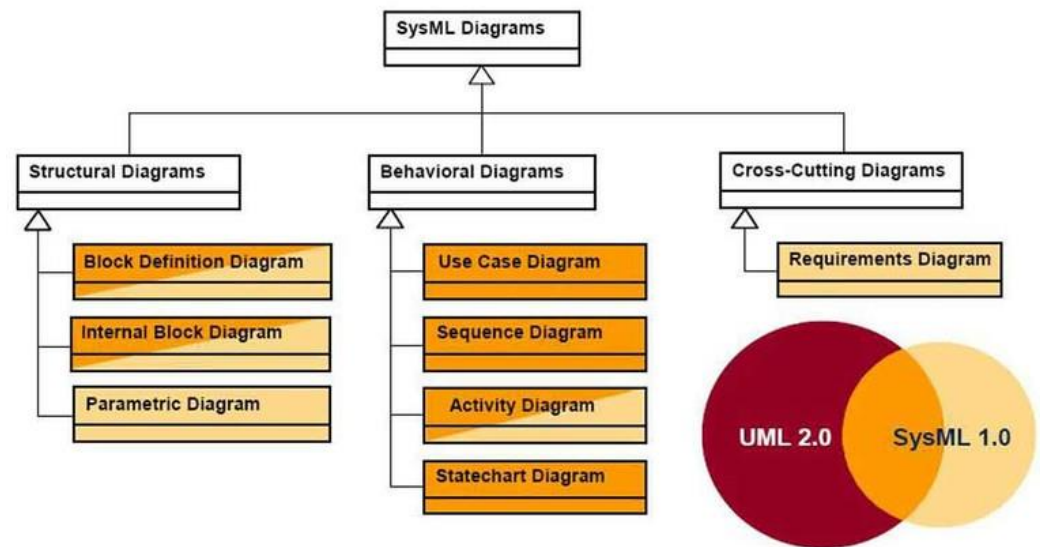
- Egyedi
- Csak a szakértők ismerik
- Egyedi eszközök kellenek
- Akár teljes kódgenerálás
- Megrendelő is jól érti

UML Profile

- Alapgondolat: terjesszük ki az UML-t
 - > Közös alapokon nyugszik
 - Könnyű megtanulni
 - Eszköztámogatás is lehetséges
 - > Szakterületi szabályok leírhatóak
- UML Profile
 - > OMG szabvány
 - > Jól formált UML részhalmoz, kiegészítésekkel
 - > Sztereotípiák, kényszerek, tag-ek
 - > Nem mond ellent az eredeti specifikációnak!

UML Profile

- UML profile for XML
- MARTE – valósidejű rendszerekhez
- SysML – rendszermodellezés
 - > Kevésbé szoftver-centrikus
 - > Elhagy és definiál diagram típusokat



Metamodellezés

- UML
 - > A nyelvek szabványban leírtak
 - > Nem lehet megváltoztatni őket
- Modellezzük a modellező nyelveket!
- Metamodellezés
 - > A nyelvek leírását is egy modellben adjuk meg!
 - > Rugalmasan összerakhatóak “saját” nyelvek
 - > Amire szükség van: nyelvleíró nyelv

Meta-Object Facility (MOF)

- OMG szabvány
- UML általánosítása
- Önleíró, négyszintű metamodellezési hierarchia
 - > 0. szint: objektumok
 - Konkrét személy objektum (pl. Nagy Lajos)
 - > 1. szint: UML modellek
 - A személy osztály definíciója (osztálydiagram)
 - > 2. szint: UML modellek modellje
 - Az osztálydiagram definíciója (és a többi diagrammé)
 - > 3. szint: MOF önleíró alapstruktúra
 - Általános modelldefiníció nyelv

MOF a gyakorlatban

- MOF – két változat
 - > Complete MOF – nem használt
 - > Essential MOF – gyakorlatban is használt
- Eclipse Ecore
 - > 95%-ban szabványos EMOF megvalósítás
 - > Az Eclipse Modeling Framework alapja
 - > Általános szakterület-, és metamodellezési keretrendszer
 - > XMI támogatással

Összefoglalás: Modellezés

Köszönöm a figyelmet!

Feladat: Use case diagram

- A könyvtárban a tagok könyveket tudnak kölcsönözni, visszahozni, valamint Internetezni tudnak
- A kölcsönzés a könyv kiválasztásával kezdődik, majd a könyvet a könyvtárhoz kell vinni, aki bejegyzi a kölcsönzést a rendszerbe
- Egyszerre maximum 5 könyvet lehet kölcsönözni
- A könyv visszahozatalakor a könyvtáros veszi vissza a könyvet és késedelem esetén büntetést szab ki a tag számára
- Az internetezéshez el kell kérni a Wifi kódot a könyvtárostól, de azt csak a főkönyvtáros tudja megadni
- A régi tagok számára lehetőség van előjegyzéseket is megadni, a normal tagoknak erre nincs lehetősége

