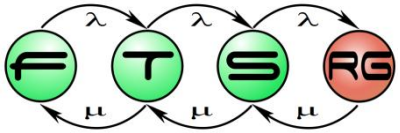
Rendszermodellezés

UML áttekintő



Tartalom

- UML bemutatása
- Use Case modell
- Osztálydiagram
- Egyéb diagramok

UML

- David Harel: Statecharts
- Rational Software: „The Three Amigos”
  - Ivar Jacobson: Object-Oriented Software Engineering
  - James Rumbaugh: Object Modeling Technique
  - Grady Booch: Booch Method
- 1997: Unified Modeling Language
- OMG 1997: befogadja
- OMG 2005: UML 2.0

UML modellek felhasználása

- Követelmények
- Analízis modell
  - Üzleti fogalmak, közvetlenül a követelményekből
  - Az implementációnál absztraktabb
- Implementációs modell
  - Részletes tervezés
  - Kódváz generálása
  - Adatmodell generálása
  - Interfész dokumentáció
  - Karbantartási célra

UML szabványok

Diagram Interchange

OCL - Object Constraint Language

Superstructure

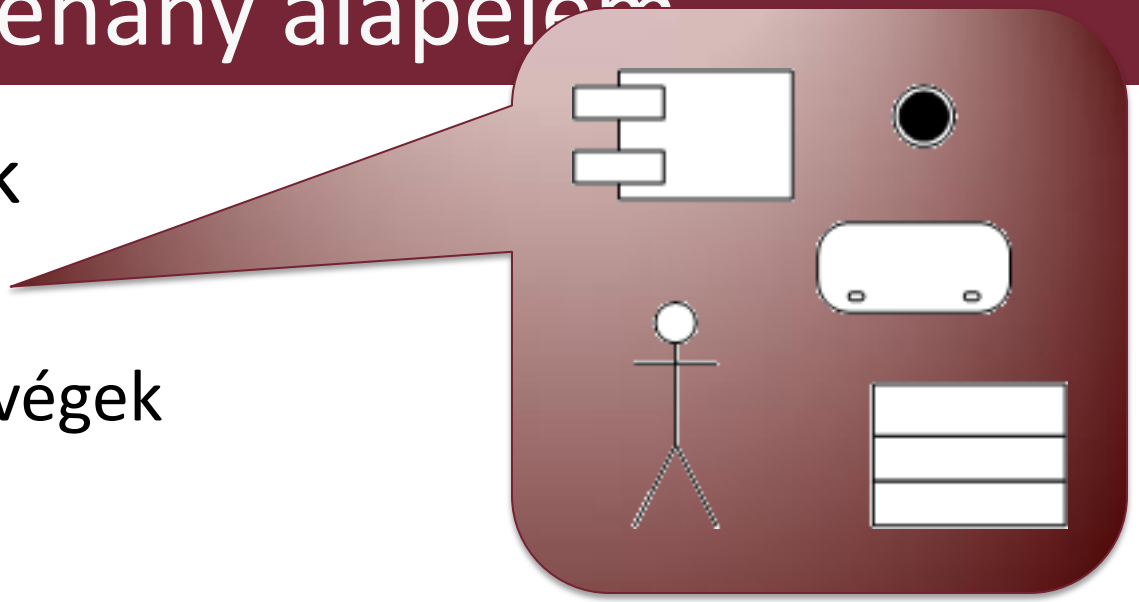
Infrastructure

Néhány alapelem

- Grafikus elemek
  - Csomópontok
  - Vonalak, vonalvégek
  - Címkék
- Általánosítás
- Sztereotípia
- Név:Típus/Szerep
- Jegyzet
- …

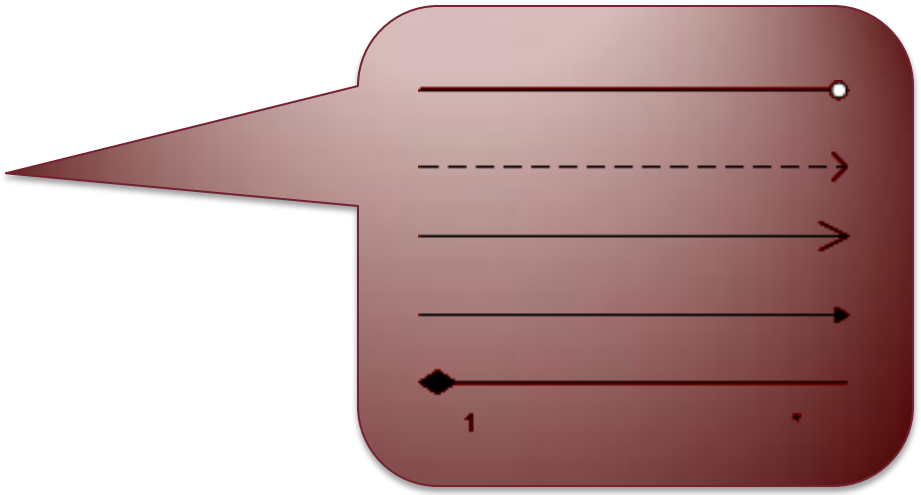
Néhány alapelem

- Grafikus elemek
  - Csomópontok
  - Vonalak, vonalvégek
  - Címkék
- Általánosítás
- Sztereotípia
- Név:Típus/Szerep
- Jegyzet
- …



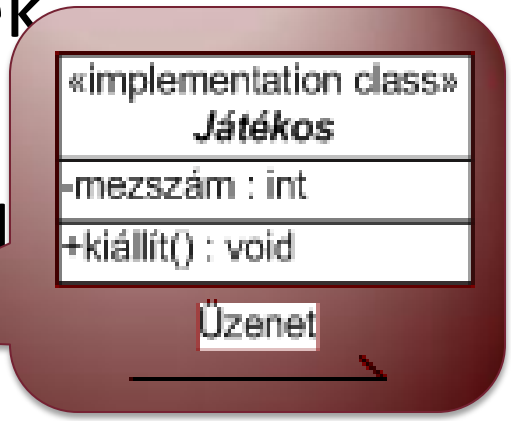
Néhány alapelem

- Grafikus elemek
  - Csomópontok
  - Vonalak, vonalvégek
  - Címkék
- Általánosítás
- Sztereotípia
- Név:Típus/Szerep
- Jegyzet
- …

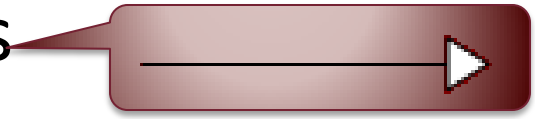


Néhány alapelem

- Grafikus elemek
  - Csomópontok
  - Vonalak, vonal
  - Címkék
- Általánosítás
- Sztereotípia
- Név:Típus/Szerep
- Jegyzet
- …



Néhány alapelem

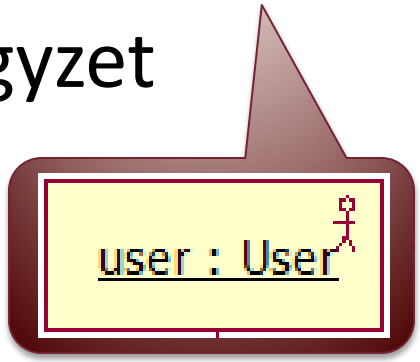
- Grafikus elemek
  - Csomópontok
  - Vonalak, vonalvégek
  - Címkék
- Általánosítás 
- Sztereotípia
- Név:Típus/Szerep
- Jegyzet
- …

Néhány alapelem

- Grafikus elemek
  - Csomópontok
  - Vonalak, vonalvégek
  - Címkék
- Általánosítás
- Sztereotípia
- Név:Típus/Szerep
- Jegyzet
- …

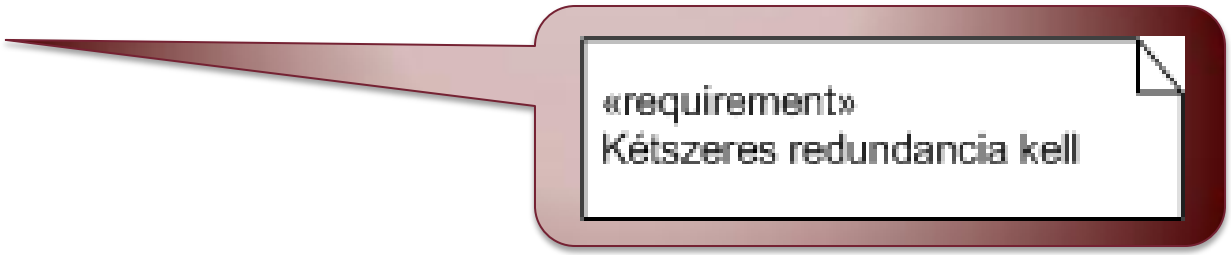
Néhány alapelem

- Grafikus elemek
  - Csomópontok
  - Vonalak, vonalvégek
  - Címkék
- Általánosítás
- Sztereotípia
- Név:Típus/Szerep
- Jegyzet
- …

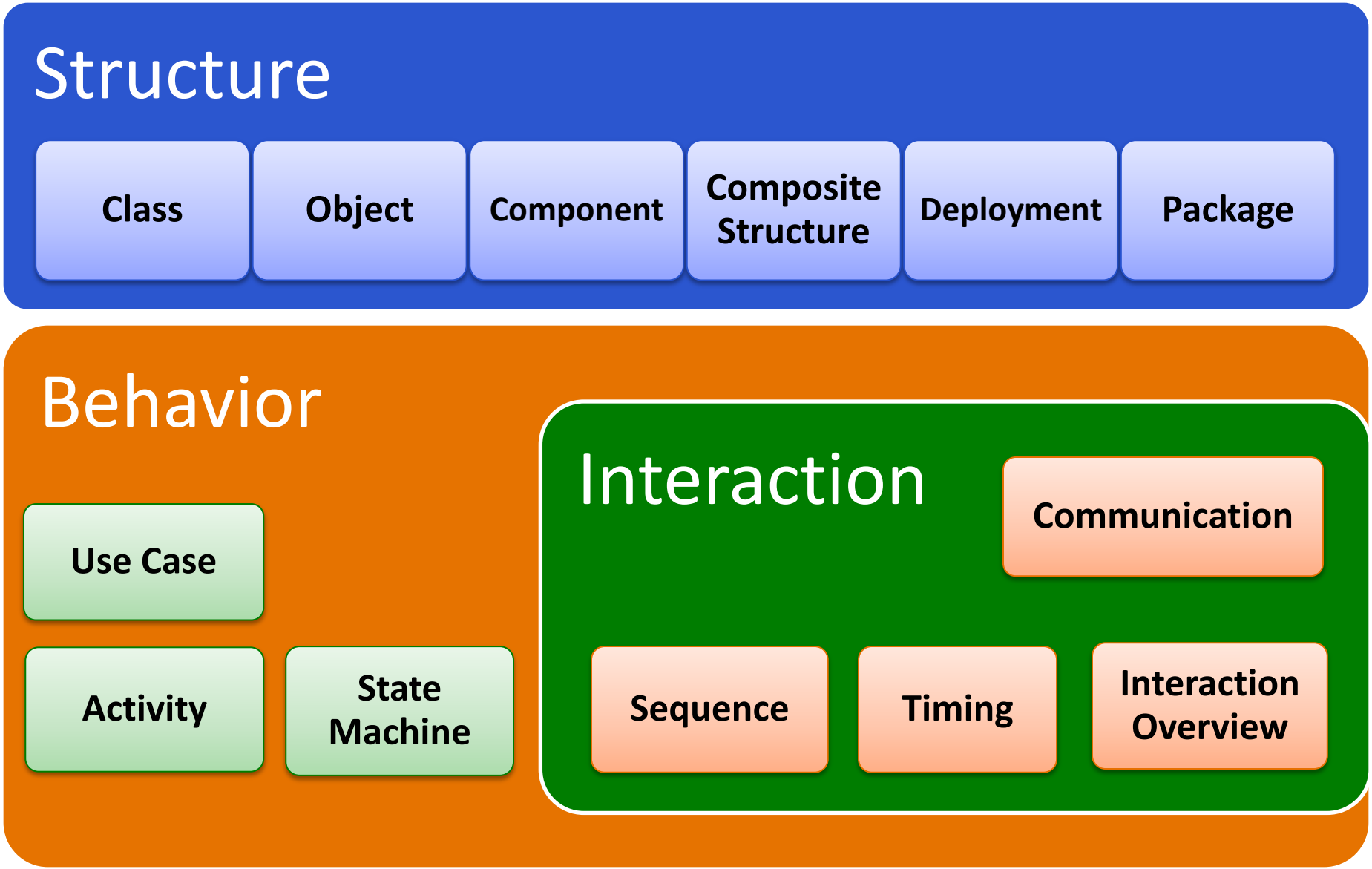


Néhány alapelem

- Grafikus elemek
  - Csomópontok
  - Vonalak, vonalvégek
  - Címkék
- Általánosítás
- Sztereotípia
- Név:Típus/Szerep
- Jegyzet
- ...

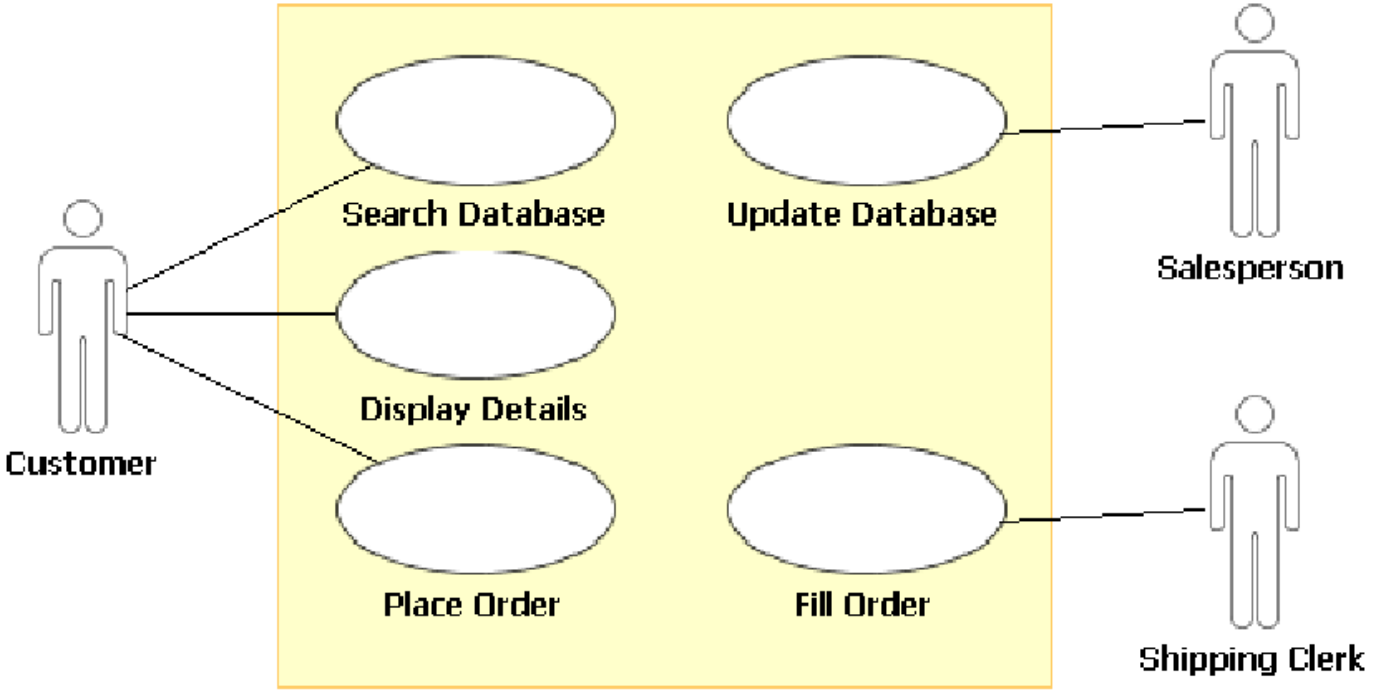


UML diagramok taxonómiája



Use Case modell

- Alapelemek
  - Use Case
  - Actor
  - Köztük „részt vesz” kapcsolat



Use Case modell

- Use Case („használati eset”)
  - ***funkcionális* követelmény**t rögzít
  - tipikus rendszer⇔külvilág(actor) **interakció**
  - tovább bontható: scenario („forgatókönyv”?)
  - tipp az elnevezésre: (angol) ige + főnév
- Actor („aktor”)
  - A UC-ek rendszeren kívüli **résztvevő**je (több-több)
  - interakcióbeli **szerep**et jelent, nem azonosítót / típust
    - Több-több összerendelés a külvilág tényleges elemeivel
  - Elsődleges (UC kezdeményező) / másodlagos

Use Case modell

- Szöveges követelményekből Use Case
- Internetes sakkbajnokság-szervező
  - Each registered player may announce a championship.
  - A player should register and log in to the system before using it.
  - Each player is allowed to organize a single championship at a time.
  - Players may join (enter) a championship on a web page
  - When the sufficient number of participants are present, the organizer starts the championship.
  - After starting a championship, the system must automatically create the pairings in a round-robin system.

Use Case modell

- Szöveges követelményekből Use Case
- Internetes sakkbajnokság-szervező
  - Each registered player may announce a championship.
  - A player should register and log in to the system before using it.
  - Each player is allowed to organize a single championship at a time.
  - Players may join (enter) a championship on a web page
  - When the sufficient number of participants are present, the organizer starts the championship.
  - After starting a championship, the system must automatically create the pairings in a round-robin system.

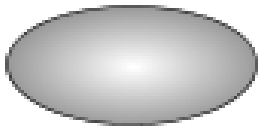
Use Case modell

- Szöveges követelményekből (folyt.)
  - If the championship is not started yet (e.g. the number of participants does not reach a minimum level), the organizer may cancel the championship
  - The actual game is played between existing clients, which is outside the scope of the system.
  - Both players should report the result and the moves after each game using a web form. A win scores 1 point, a draw ½, and a loss 0.
  - If players report contradicting results, the organizer should judge who the winner is, and penalize the cheating player by a 1 point penalty.
  - Finally, when all games are finished, the organizer should close the championship by announcing the winner. Then he or she may start organizing a new championship.

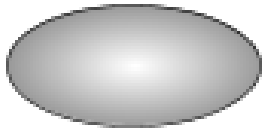
Use Case modell

- Szöveges követelményekből (folyt.)
  - If the championship is not started yet (e.g. the number of participants does not reach a minimum level), the organizer may cancel the championship
  - The actual game is played between existing clients, which is outside the scope of the system.
  - Both players should report the result and the moves after each game using a web form. A win scores 1 point, a draw ½, and a loss 0.
  - If players report contradicting results, the organizer should judge who the winner is, and penalize the cheating player by a 1 point penalty.
  - Finally, when all games are finished, the organizer should close the championship by announcing the winner. Then he or she may start organizing a new championship.

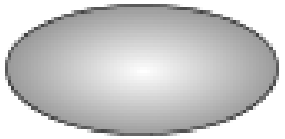
Kezdeti Use Case modell



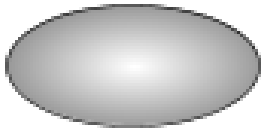
Announce Championship



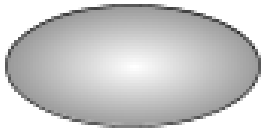
Enter Championship



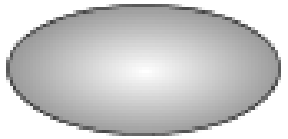
Judge Result



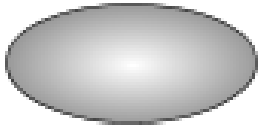
Cancel Championship



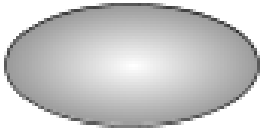
Start Championship



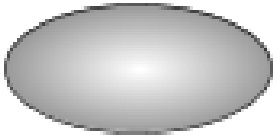
Report Result



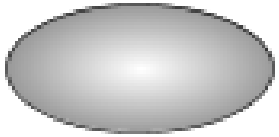
Close Championship



Create Pairings

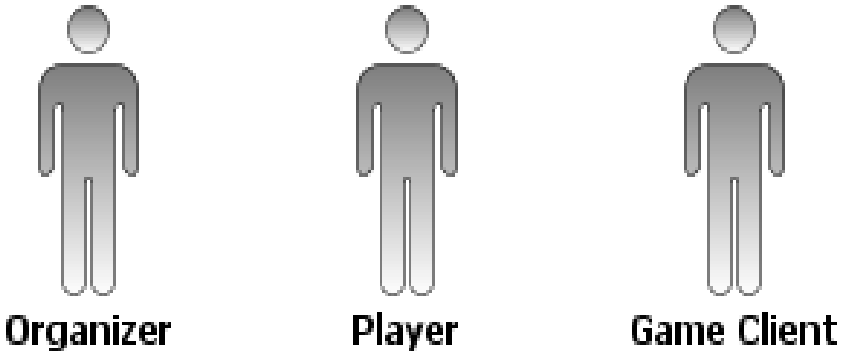


Register User

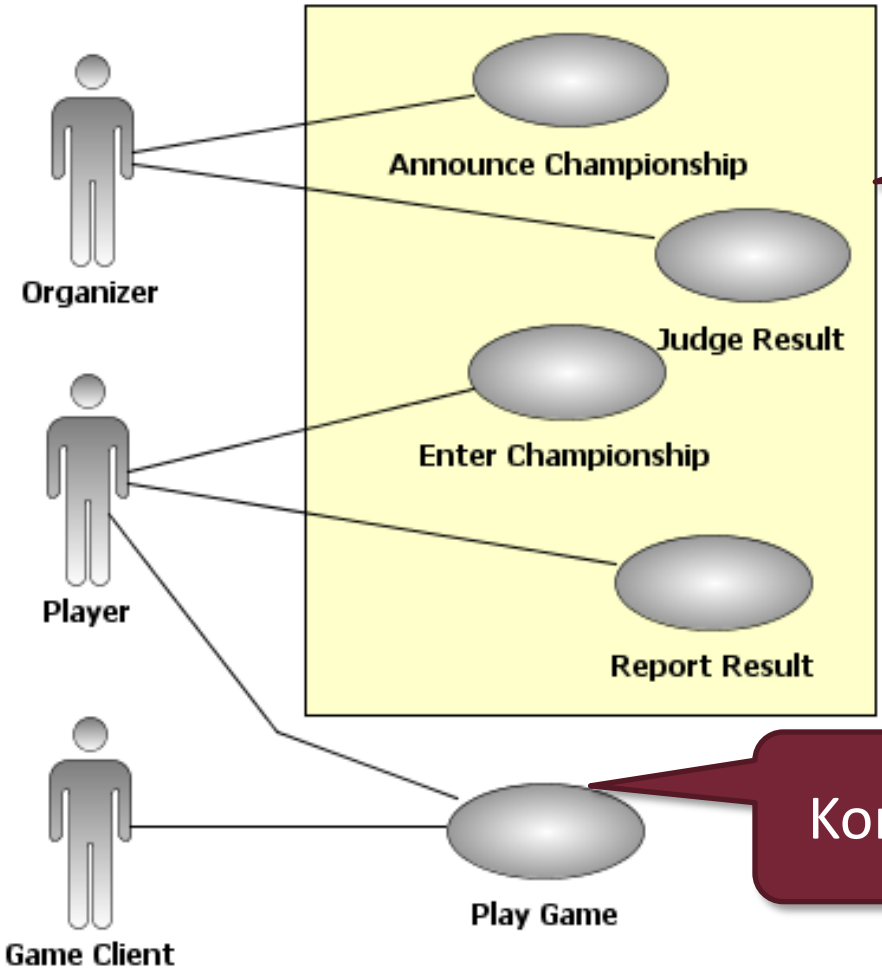


Login User

Kezdeti Use Case modell



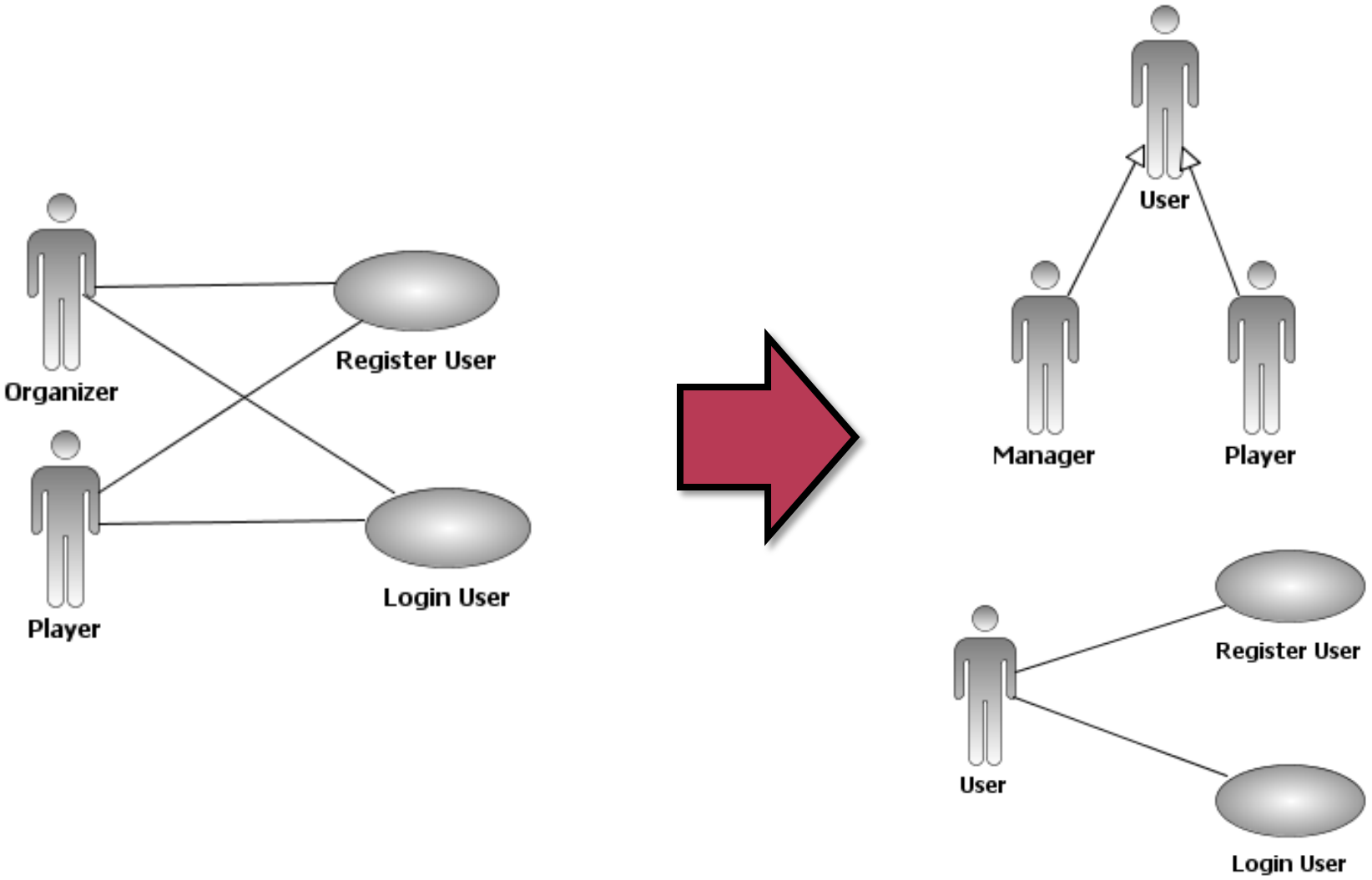
Kezdeti Use Case modell



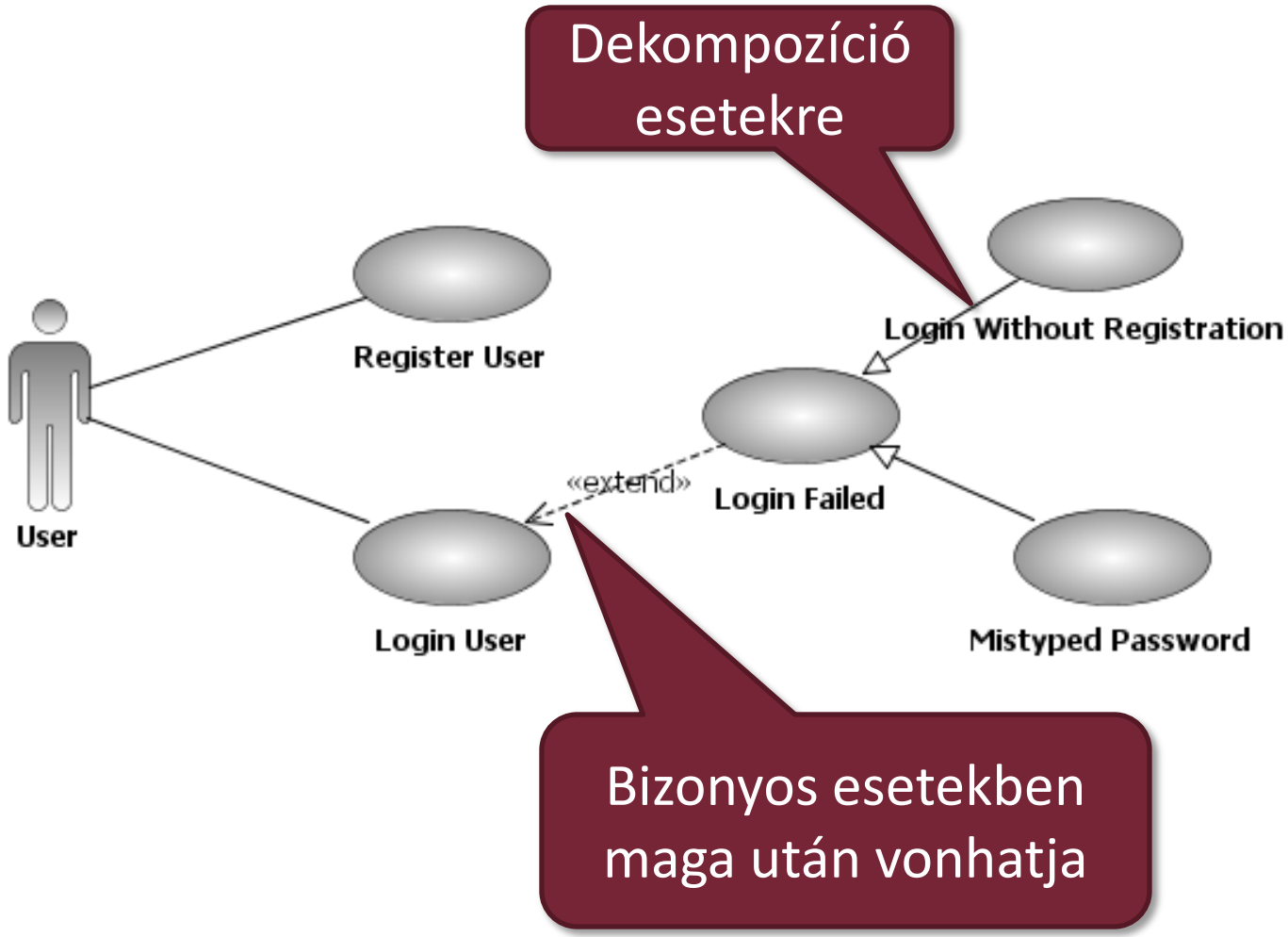
Rendszer határa

Kontextuális (rendszeren kívüli)

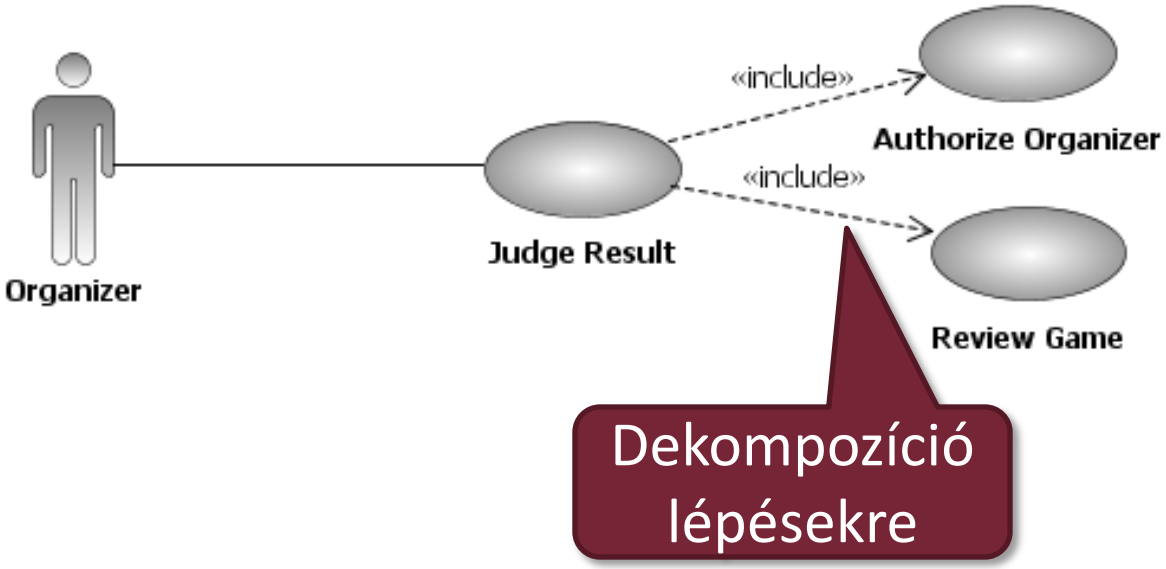
Use Case modell átalakítása



Use Case modell átalakítása



Use Case model átalakítása



Use Case diagram - élek

- Aktor – Use Case asszociáció
  - Részt vesz benne
- Aktor – aktor általánosítás
  - Nem hivatalos
- Use Case – Use Case általánosítás
  - Különböző esetek összevonása
- Use Case – Use Case «include»
  - Lépés azonosítása
- Use Case – Use Case «extend»
  - Opcionális része (választható vagy esetenkénti)

Use Case diagram - gyakorlat

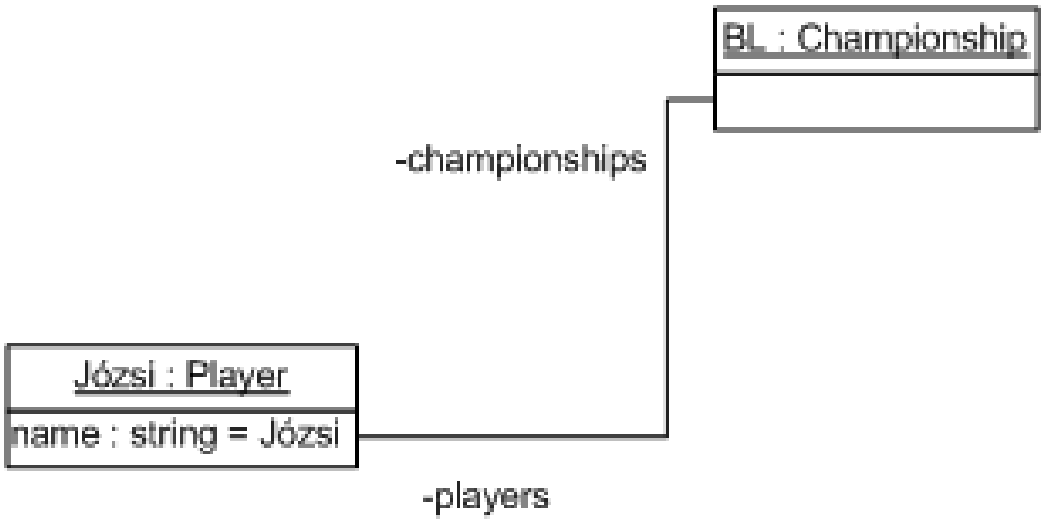
- A rendszer egy havidíjas többszereplős on-line játék.
- A játékba való belépés vastag kliensen keresztül zajlik, míg a befizetés és egyéb személyes adatok megadása egy weboldalon keresztül történik (mindkettő az adott felületen történő azonosítás után).
- A játékos miután a vastag kliensen keresztül azonosította magát és befizette a havidíjat (a weboldalon keresztül) játszhat meglévő játékkaraktereivel. Ha nincs ilyen, akkor új karaktert generál. Új karaktert akkor is lehet generálni, ha a játékosnak már van karaktere, de éppen nem játszik egyikkel se. Játék után a játékos új karaktert választhat, készíthet, vagy kiléphet.

Use Case diagram - gyakorlat

- A weblapon történő azonosítás után befizetés történhet hitelkártyával vagy feltöltőkártyával. A személyes adatok bekérése is itt zajlik, ezekre általában csak egyszer, a felhasználói fiók létrehozásakor kerül sor. A játékos továbbá beléphet a fórumba ahol üzenetet hagyhat.
- A játékos több egyéb szolgáltatást is elér a weblapon, de ezekhez nem szükséges belépés. Ezek tetszés szerint bővíthetők (karakterek elért eredményei és pontjai, szerverek státusza, stb).
- A játékvilágot adminisztrátorok töltik fel adatokkal. Kezelőfelületük nagyban hasonlít a játékosok által elérhető felülethez. Azonban nincs jogosultságuk bejelentkezni a weboldalra. Ugyanígy a weboldal adminisztrátorai biztonsági okokból nem férhetnek hozzá a játékvilág adataihoz. A weboldal adminisztrátorai egy megnövelt funkcionalitású kezelőfelületen keresztül dolgoznak, mely funkcionalitásában lényegesen különbözik a játékosok által megszokott webes felülettől.

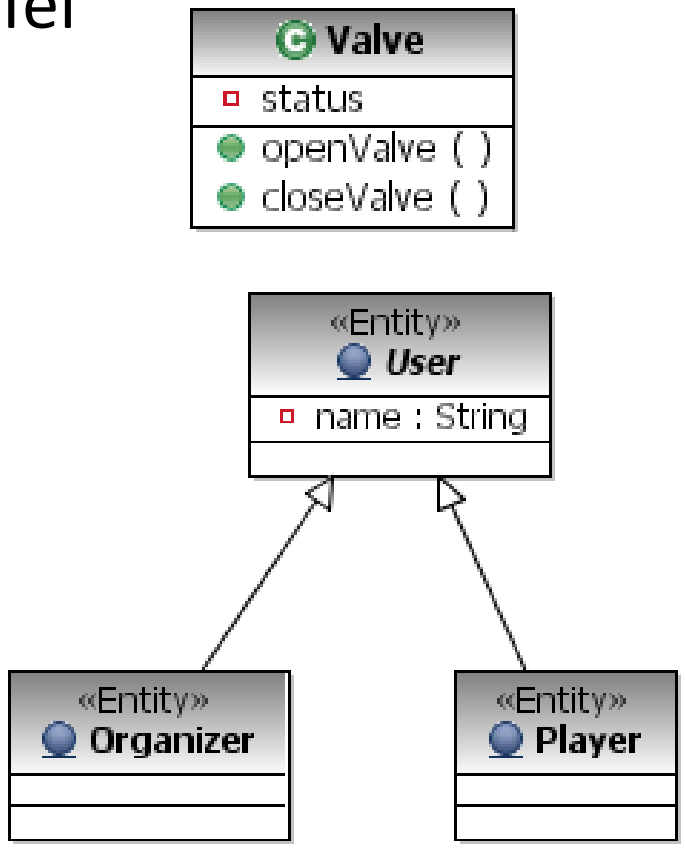
Objektumdiagram

- Hagyományos objektum-orientált szemlélet
  - A rendszer **objektum**okból épül fel
  - Tulajdonságok, műveletek
    - Értékek
    - Kapcsolatok



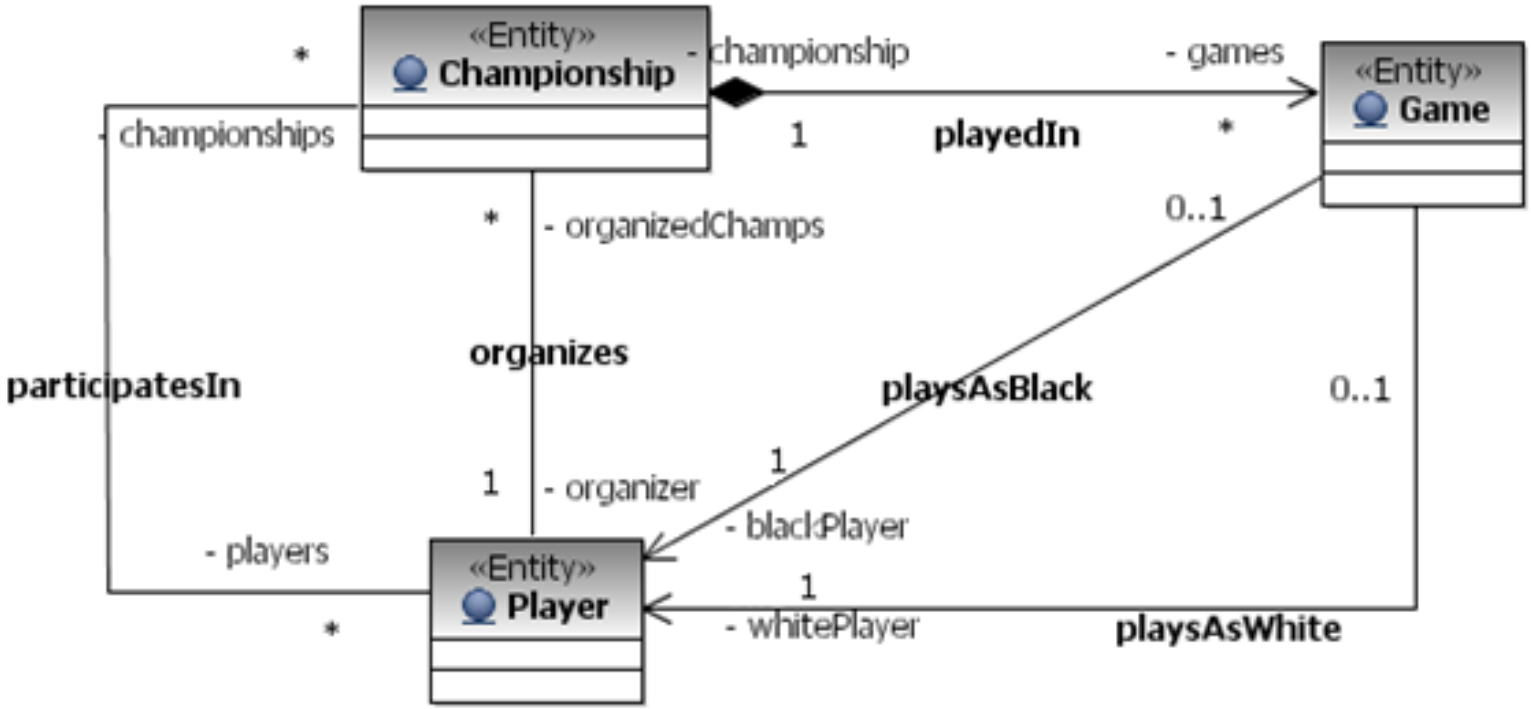
Osztálydiagram elemei

- Hagyományos objektum-orientált szemlélet
  - A rendszer **objektum**okból épül fel
  - Tulajdonságok, műveletek
- OO modell metamodellje
- Objektumok sablonja: **osztály**
  - **Attribútum**ok
  - **Operáció**k
  - Láthatóság, típus, stb.
- Osztályok közt általánosítás



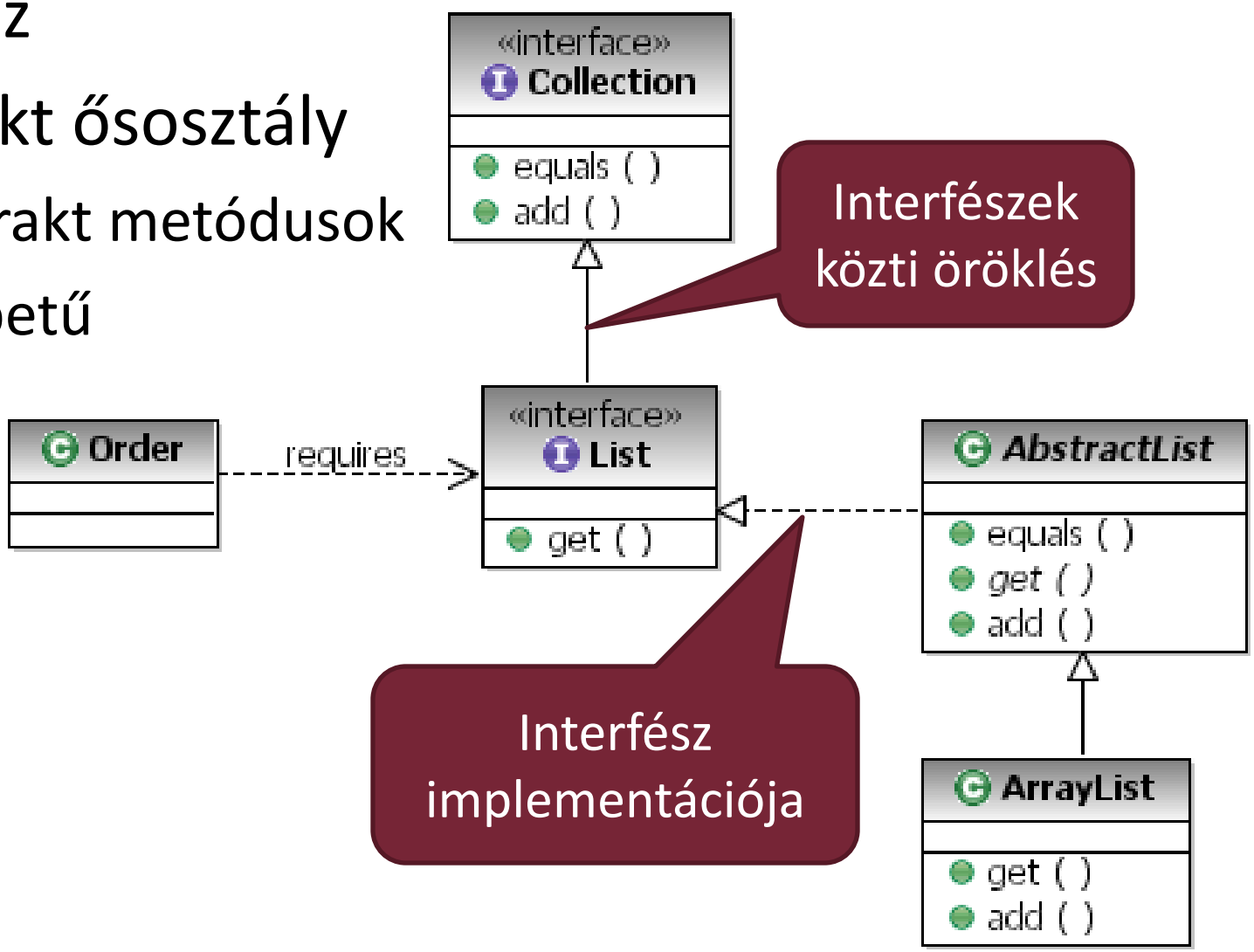
Osztálydiagram elemei

- Kapcsolat sablonja: **asszociáció**
  - Végeken: multiplicitás, navigálhatóság, szerepnév
  - Tartalmazás: aggregáció / kompozíció
  - Bináris asszociáció ⇔ attribútum



Osztálydiagram elemei

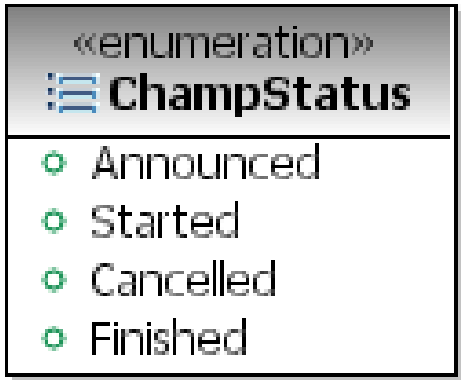
- Interfész
- Absztrakt ősosztály
  - Absztrakt metódusok
  - Dőlt betű



Osztálydiagram elemei

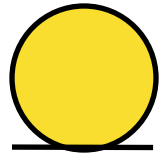
- Classifier Scope (≃ Java static member)
- Többágú asszociáció
- Származtatott tulajdonságok

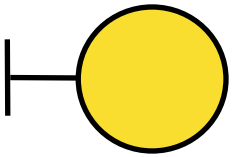
- Láthatóság
  - +public, -private, #protected, ~package
- Szimbolikus felsorolás típus
- …

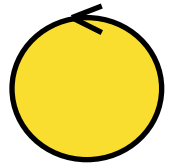


Analízis célú osztályok

- RUP javaslata: analízis modell osztályai
- Entitás osztály (Entity class)
  - Perzisztens adat
  - Túléli a UC végét
- Határoló osztály (Boundary class)
  - Felület az aktorok és a rendszer közt
  - Pl. webes űrlap, ablak, SOAP webszolgáltatás…
- Vezérlő osztály (Control class)
  - Az üzleti logikát koncentrálja
- Sztereotípiaként jelölhető







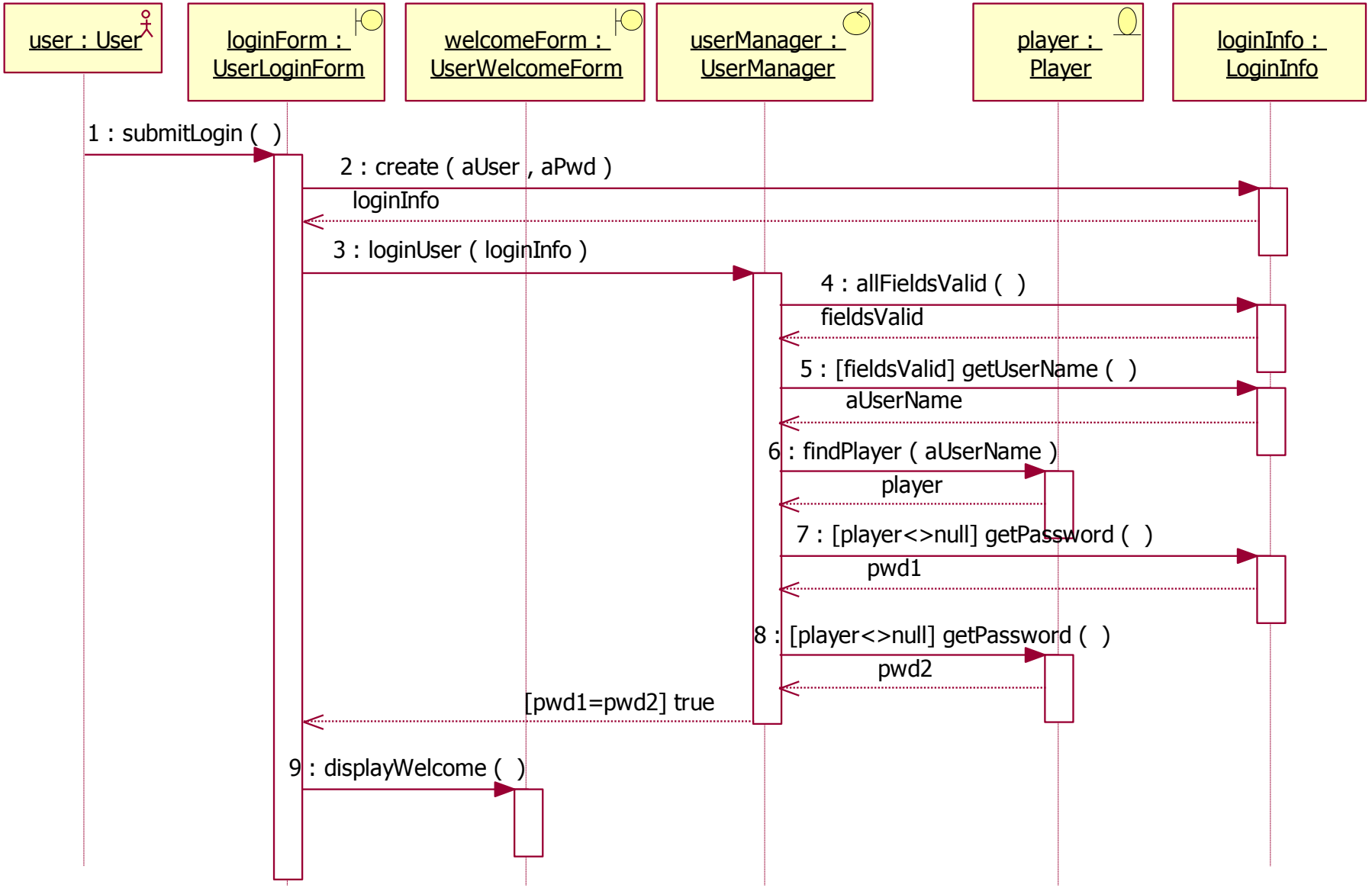
Szekvencia diagram - felhasználás

- Use Case tipikus végrehajtása
- Scenario tipikus végrehajtása

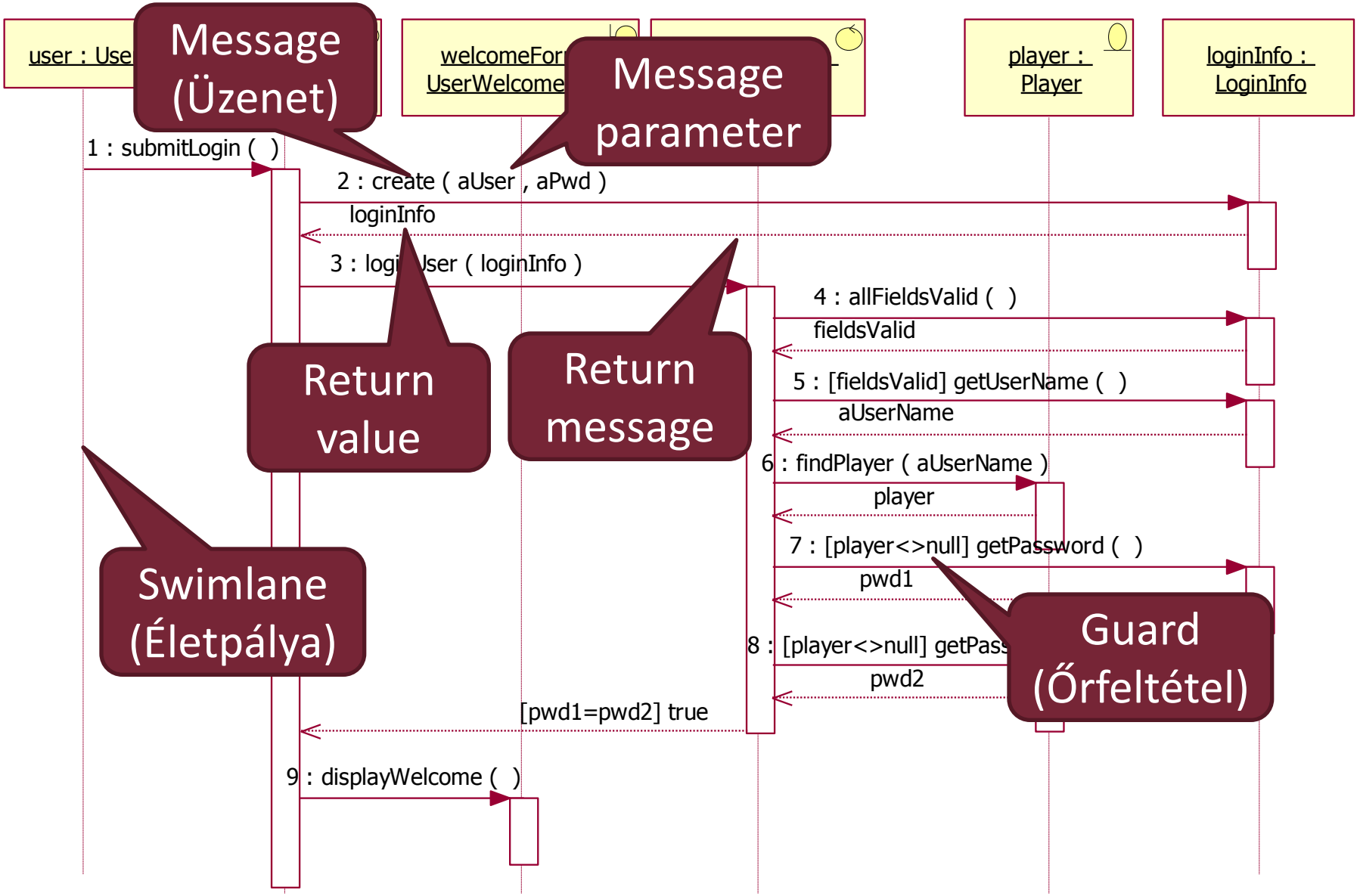
} Kollaboráció

- Komponensek, objektumok kommunikációja
  - Analízis, tervezési fázisban egyaránt
- State Machine, Activity Diagram egy lefutása
- Teszteset specifikációja

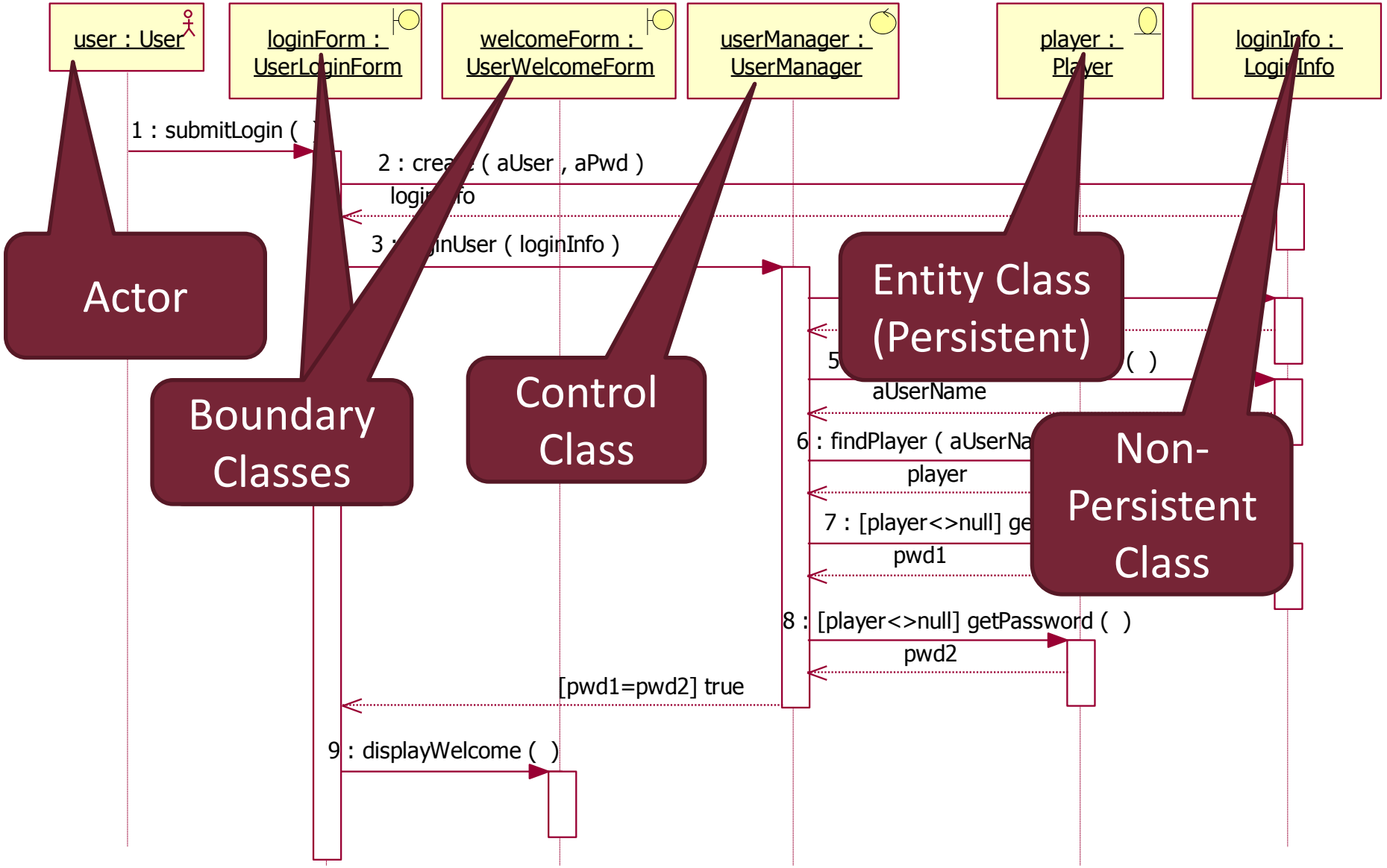
Szekvencia diagram - példa



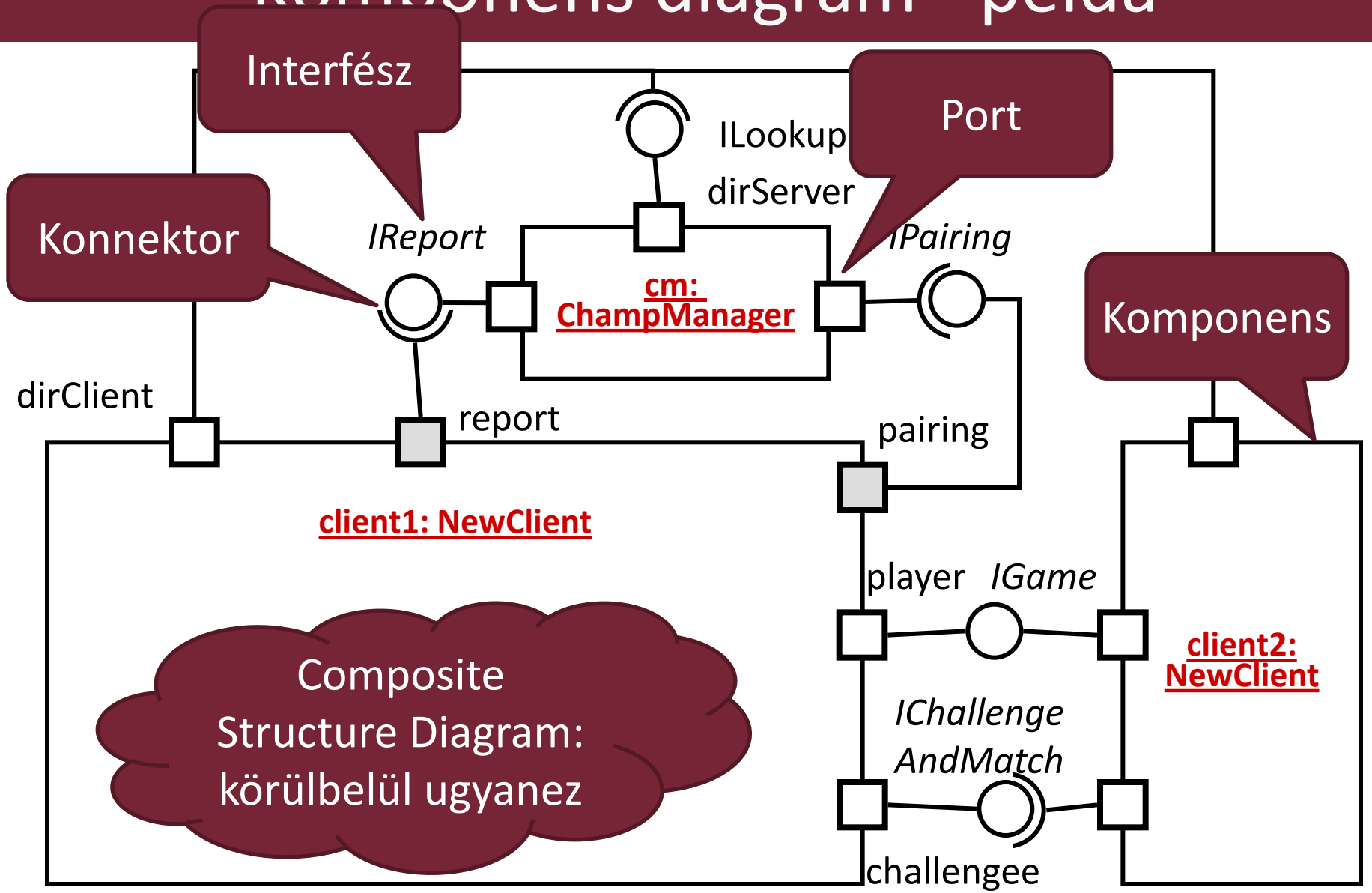
Szekvencia diagram - példa



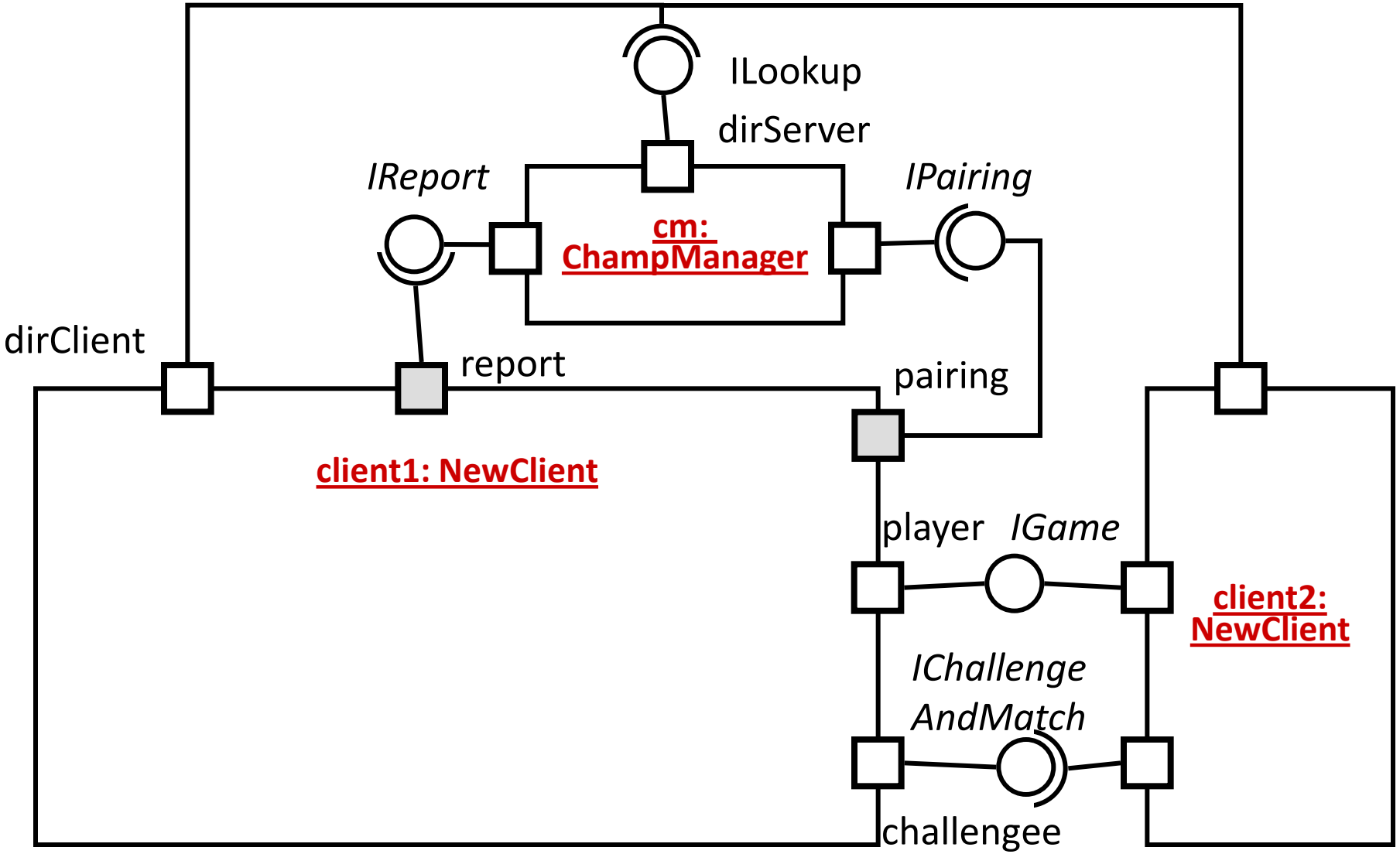
Szekvencia diagram - példa



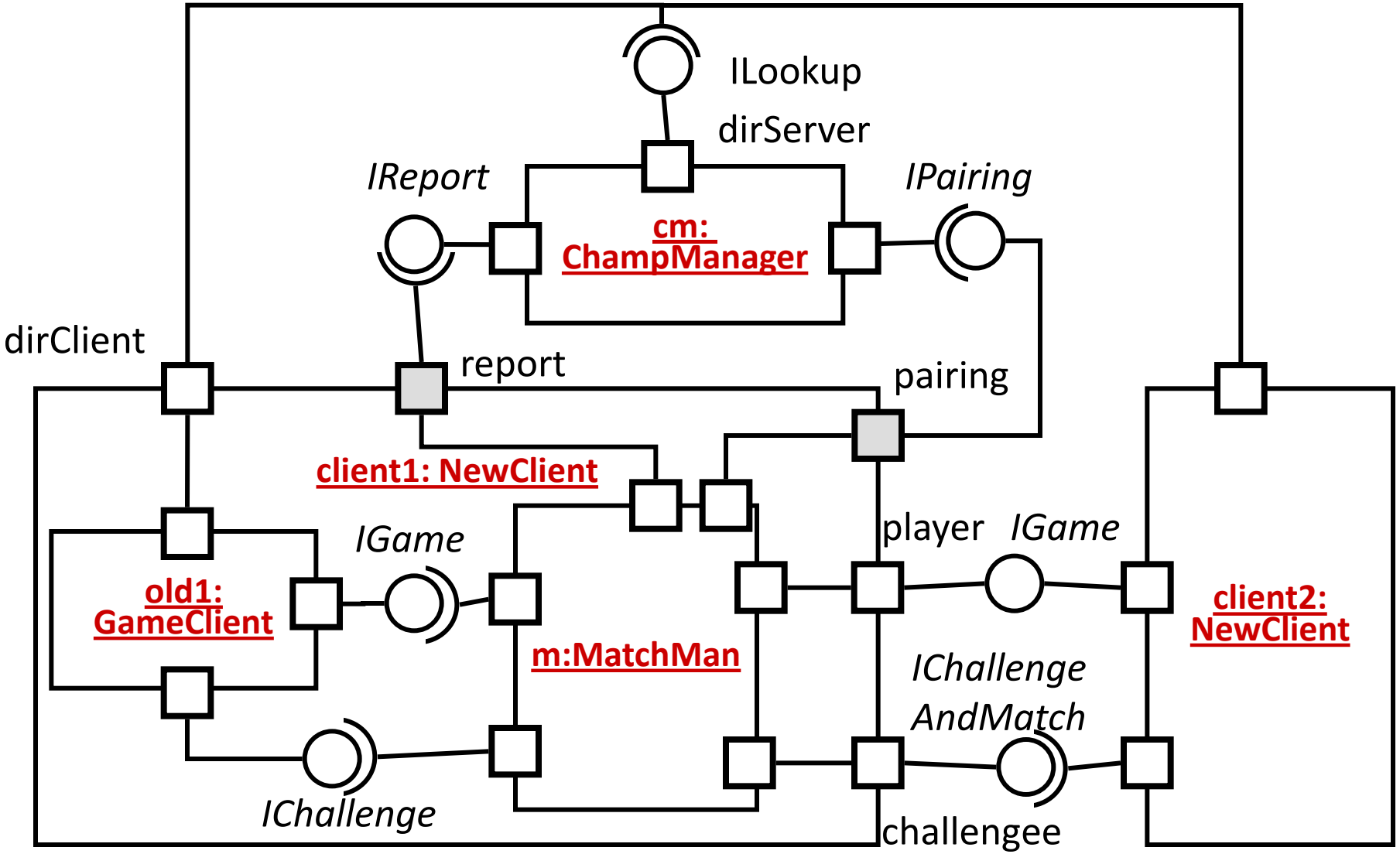
Komponens diagram - példa



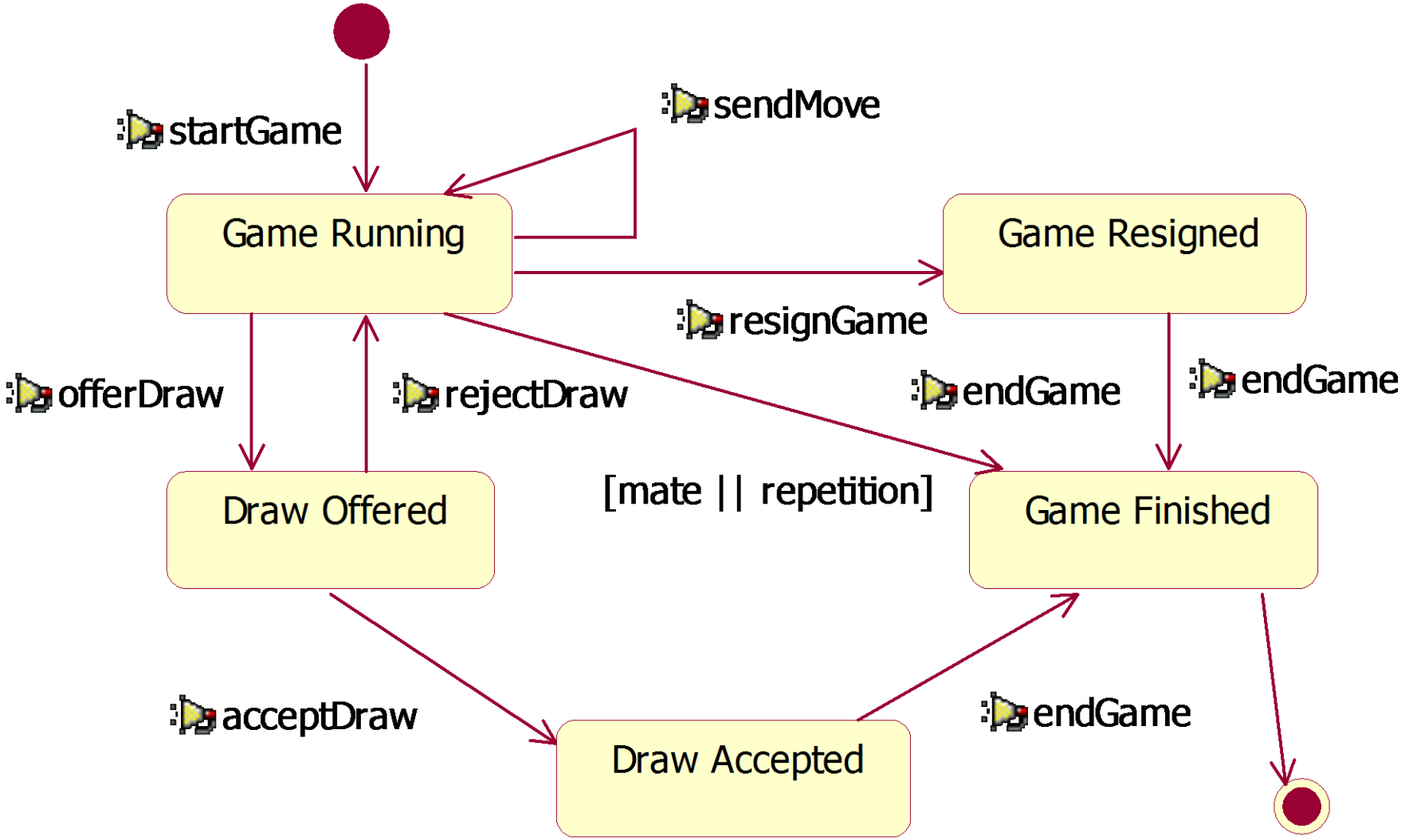
Komponens diagram - példa



Komponens diagram - példa

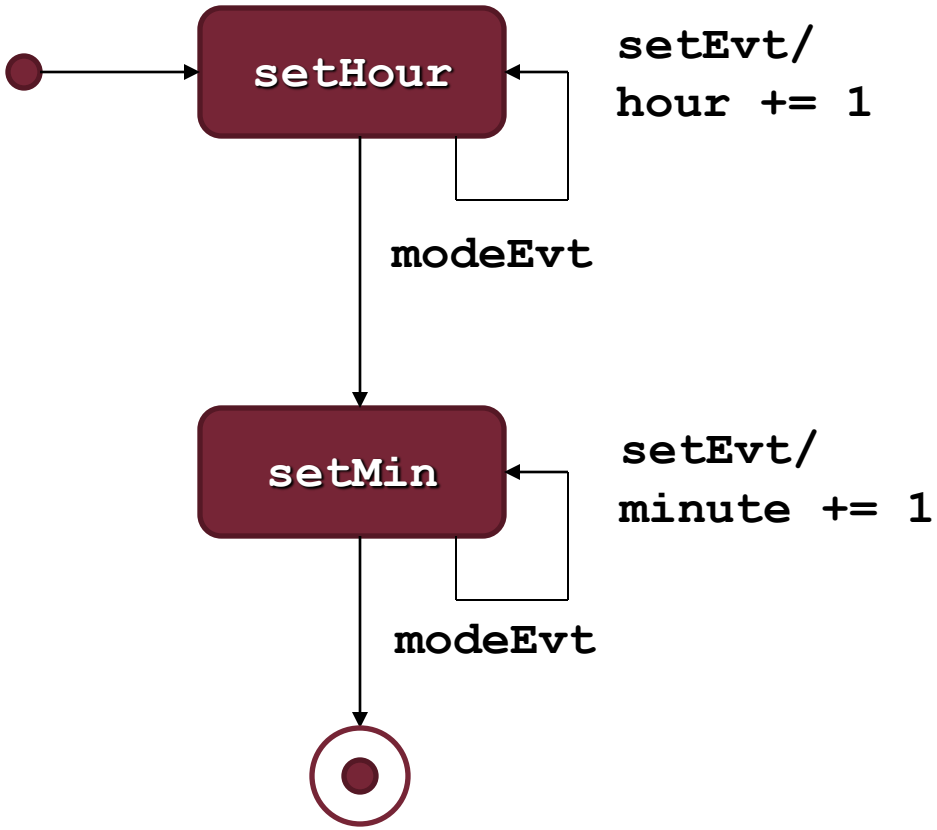


Protocol Statechart of IGame



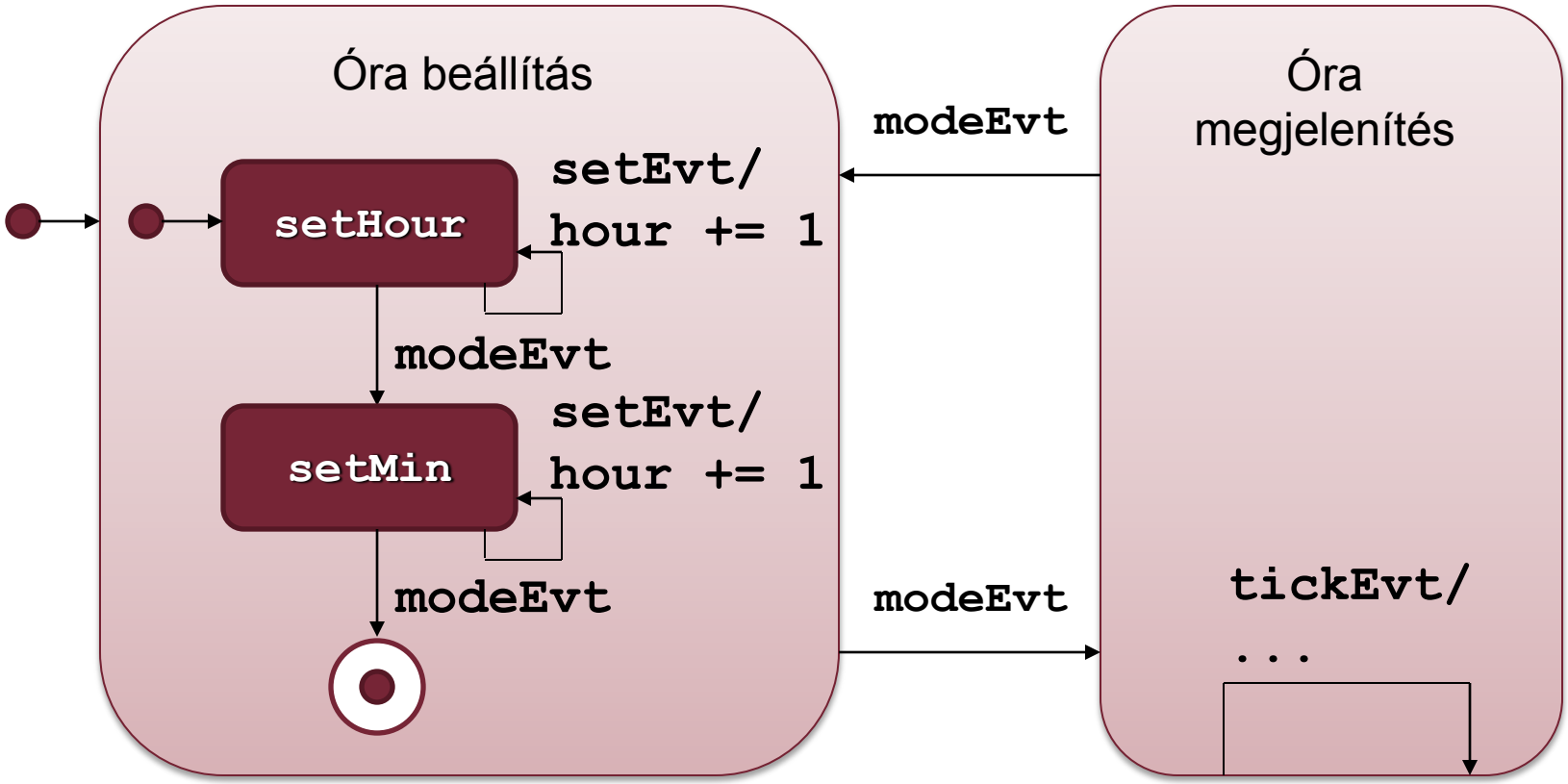
State Machine Diagram

- Állapotok
  - Egyszerű állapotok
  - Kezdő- és végállapot
  - History (ld. később)
  - Összetett (ld. később)
- Állapotátmenetek
  - Kiváltó esemény
  - Akció
  - [Guard]
- Példa: óra beállítása



State Machine Diagram

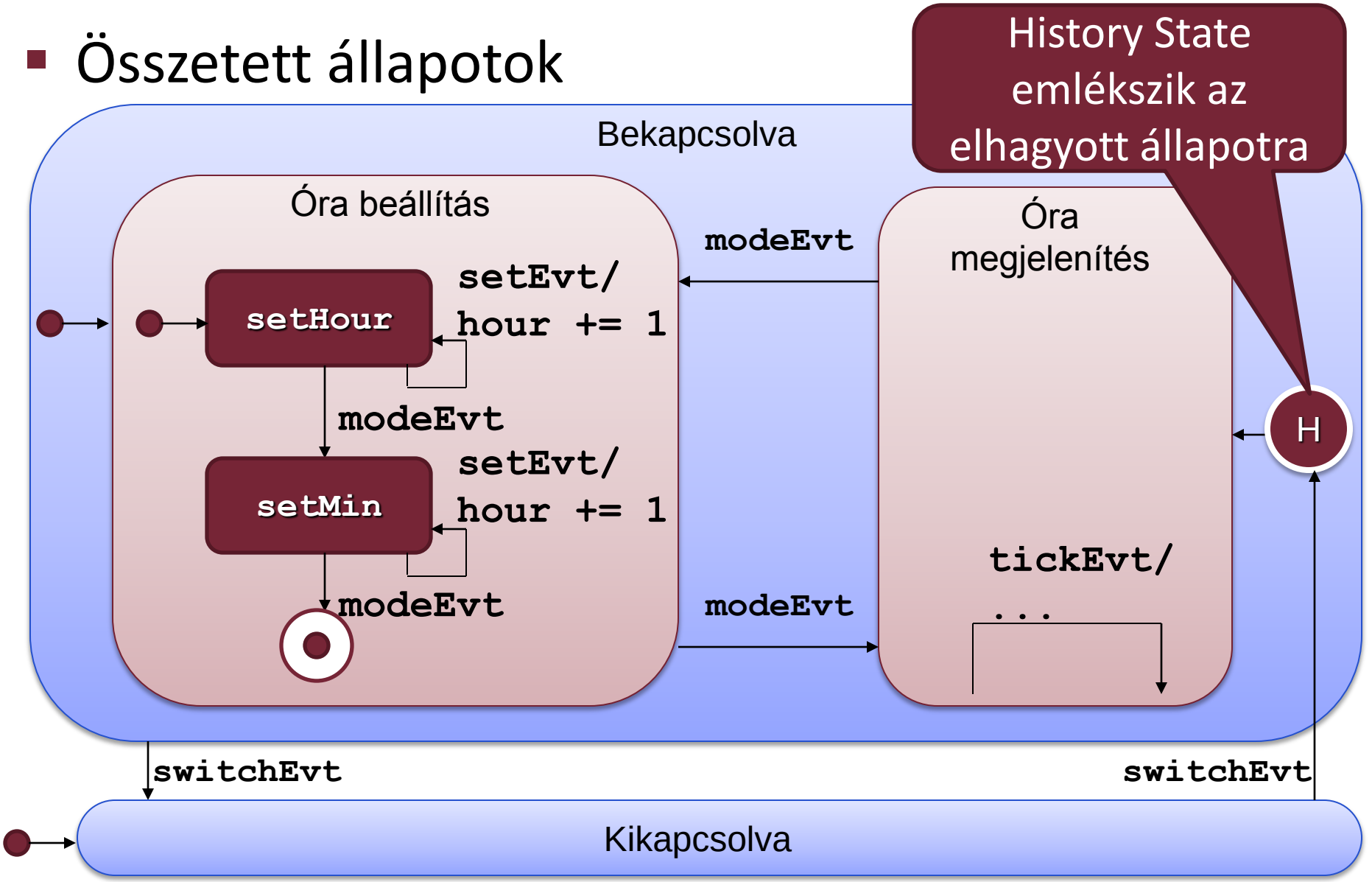
- Összetett állapotok



- Konkurens régiók

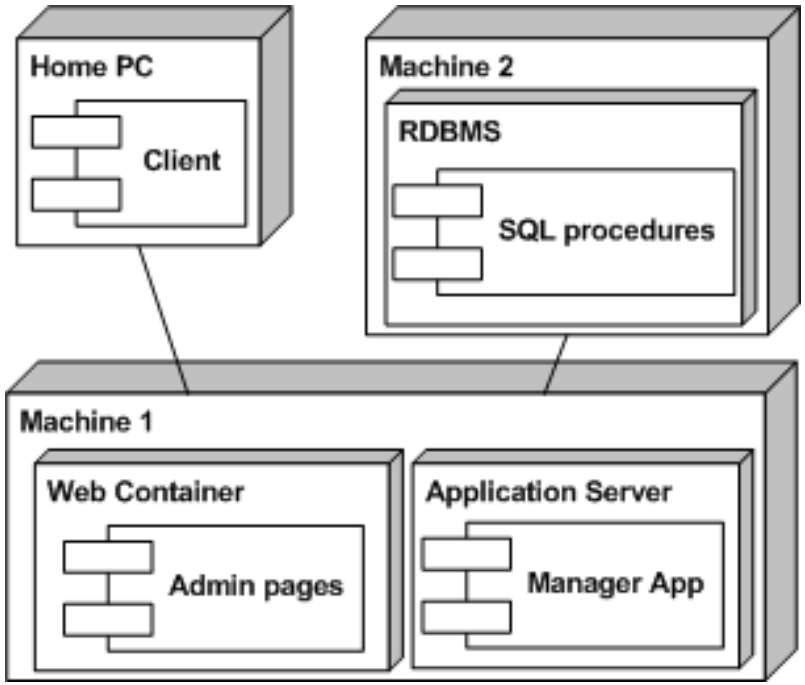
State Machine Diagram

- Összetett állapotok



További strukturális diagramok

- Package Diagram
  - Az UML modell elemei csomagokba gyűjthetőek
  - Függőség, tartalmazás, névtér import
- Deployment Diagram



További viselkedési diagramok

- Activity Diagram: folyamatmodell
  - erről még külön lesz szó
- Communication Diagram
  - Szekvencia diagram üzenetei, de objektum diagramon
  - Jól be tud mutatni egy *kollaborációt*
- Timing Diagram: állapotok időtengelyen
  - Állapotgép egy lefutása
- Interaction Overview Diagram
  - Activity Diagram összetett csomópontokkal
  - Csomópont lehet: Sequence, Timing, Communication

UML hibái, hiányosságai

- A specifikációjában mindig van redundancia, többértelmű elem, ellentmondás.
- Túl általános akar lenni…
  - …keveset fog
  - Emiatt szükségtelenül bonyolult
- Nem csak információs rendszerek vannak
  - Szoftveresek értik, más nem
  - SysML és szakterületi (Domain-Specific) modellezés…
- Jim Rumbaugh: The Preacher at Arrakeen (2001)