Komponens-alapú UML modellek fordításának vizsgálata

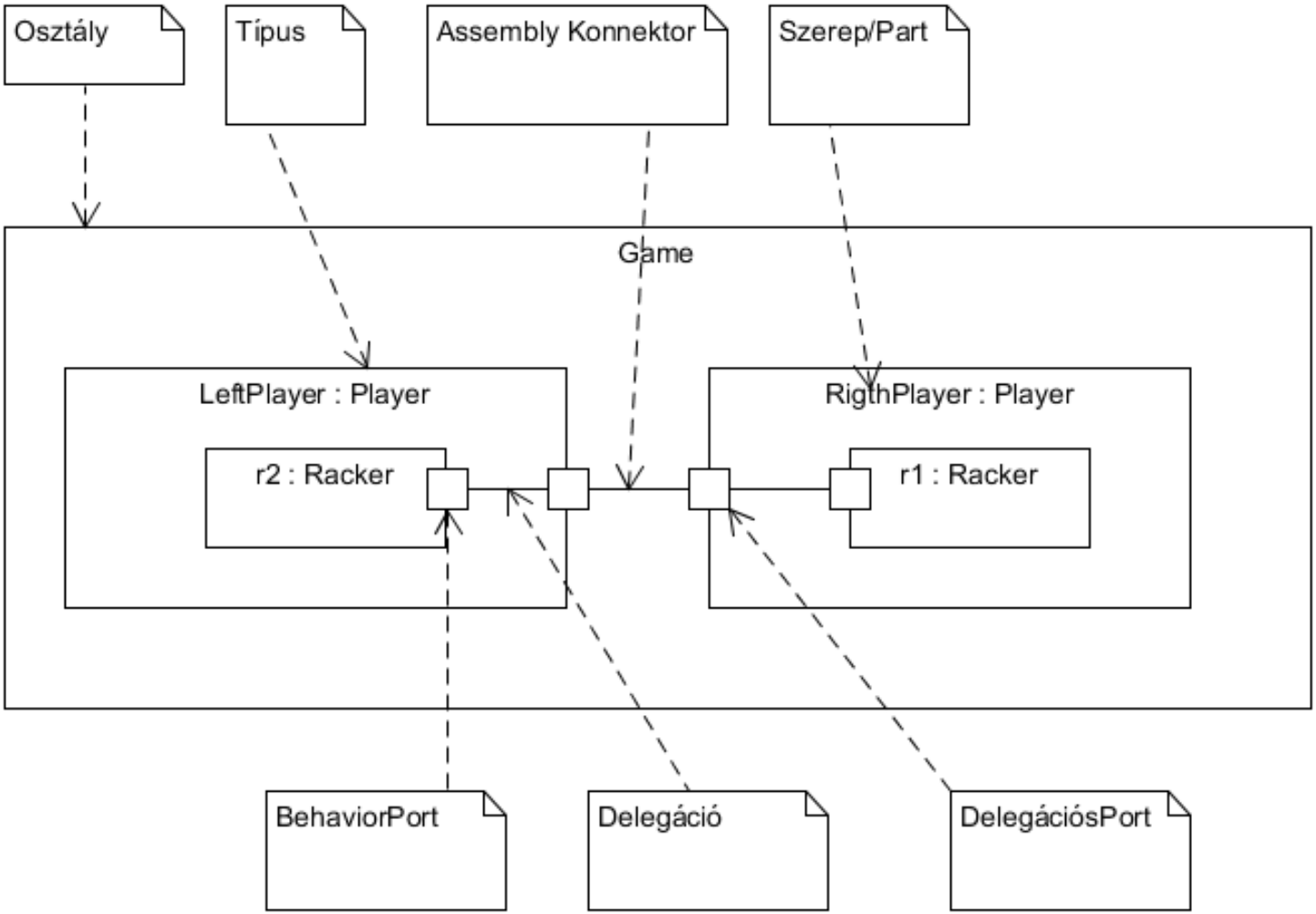
Nagy András

2019. január

Kompozíciós modell szöveges megadása

- *txtUML* keretrendszerben írjuk le a komponens-alapú modellt, Java-szerű nyelven, mely végrehajtható.
- Lefordítható egy szabványos UML2 modellre.
- A cél a kompozit struktúrák és akciók megfelelő UML2-es szabványának megtalálása, melyből hatékony C++ kódot szeretnénk generálni.

Példa egy konkrét kompozit struktúrára



Főbb UML reprezentációs problémák

- Interfész port reprezentálása: mi a típusa, hogyan fejezzük ki az elvárt interfészt.
- Két port összekapcsolása futási időben.
- Porta való üzenetküldés.
- Mi a szabványos modell szemantikája?

Példa: Konnektor, konnent UML-ben

UML specifikáció szerint

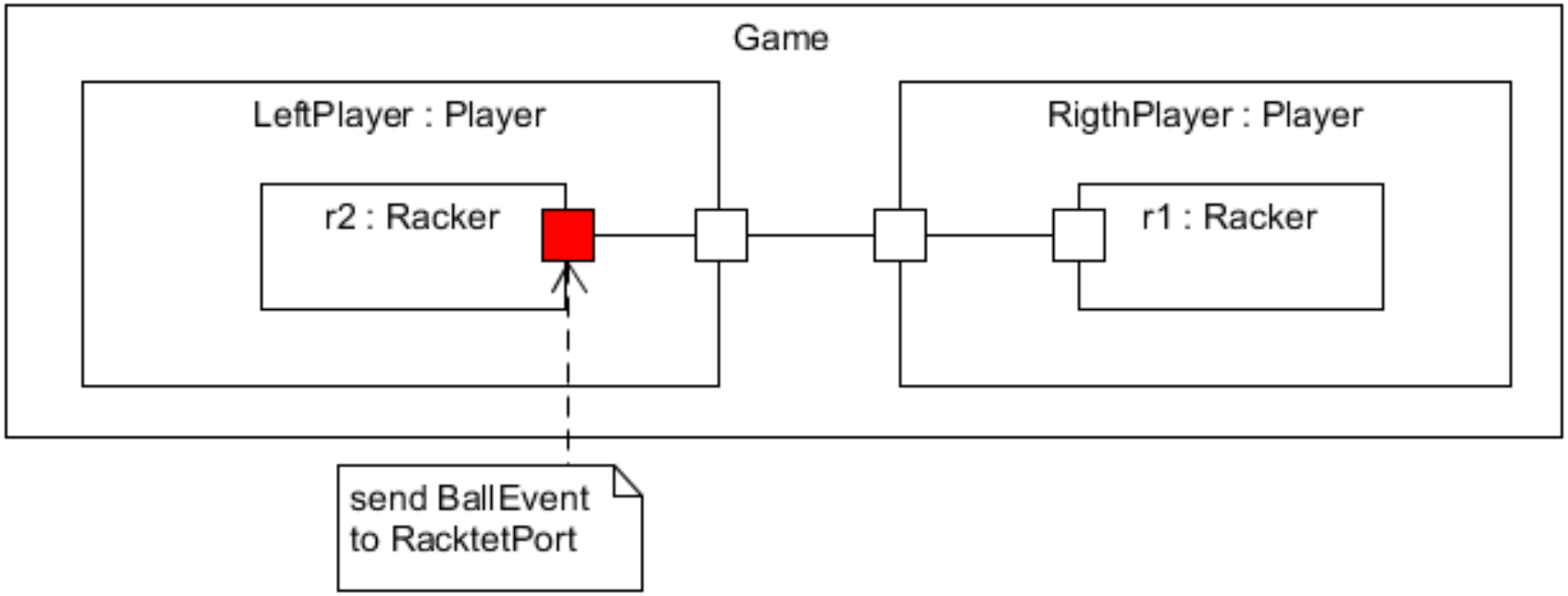
A Connector specifies links (see 11.5 Associations) between two or more instances playing owned or inherited roles within a StructuredClassifier." "A CreateLinkAction is a LinkAction for creating links."

- A *Connector* értelemszerű (de mi az a *type* referencia?)
- A problémás a *connect* művelet
  - Nincs *connect* akció UML-ben
  - *DefaultConstructionStrategy* (de az a szerkezet nem mindig egyértelmű..)
  - A *CreateLinkAction* segítségével összeköthetünk két portot a konnektor típusa mentén (Itt jön be a *type* referencia, mely egy asszociációnak felel, ezt még ki kell generálni.).

C++-ra való fordítása

- Sok alternatíva (típusbiztonság, adatreprezentációs különbségek, stb.)
- Különböző elvárások a generált kóddal szemben (hatékonyság, olvashatóság, külső kóddal történő biztonságos illesztés, stb.)
- Ez főleg a megfelelő kertrendszerek kialakításából és az UML szemantika értelmezéséből áll.
- Ezeket a szempontokat figyelembe véve a munka elemzi az egyes elemek kompozit kódgenerálását.

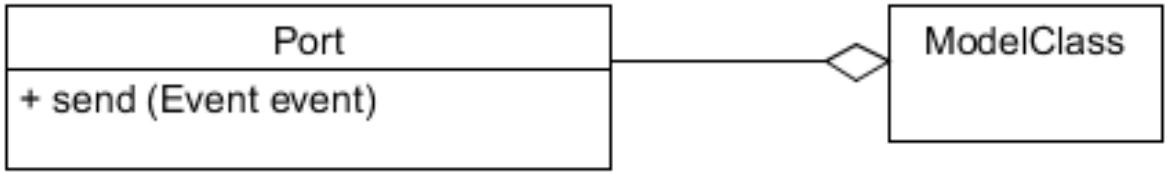
Üzenetáram eleje



Kérdések:

- Hogyan reprezentáljuk a szabványban a portra való üzenetküldést, hogyan kell ezt értelmezni?
- Interfész portok reprezentálása, generálása?

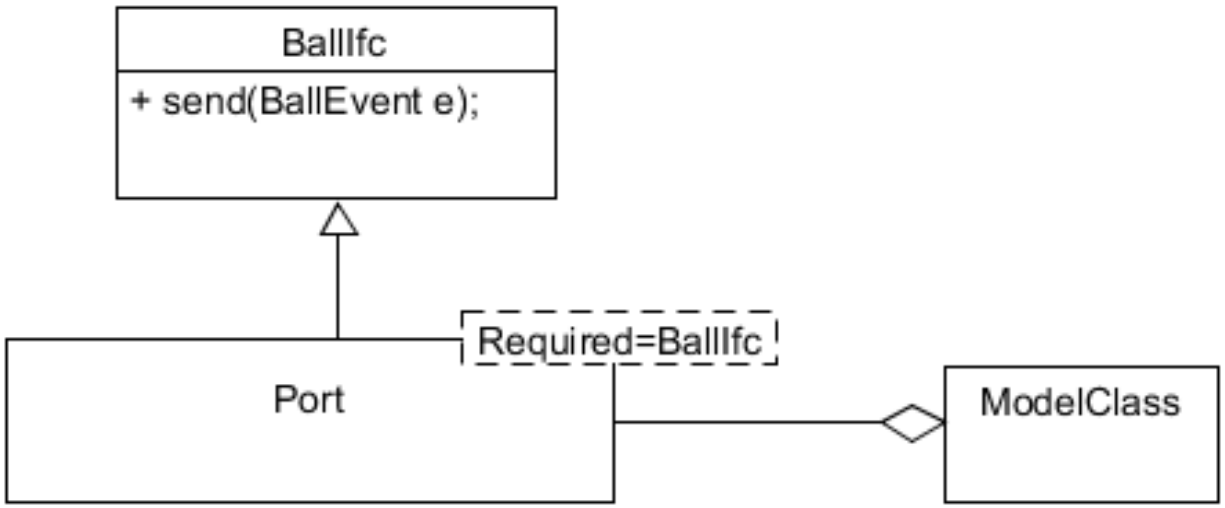
Portok naiv ábrázolása



Problémák:

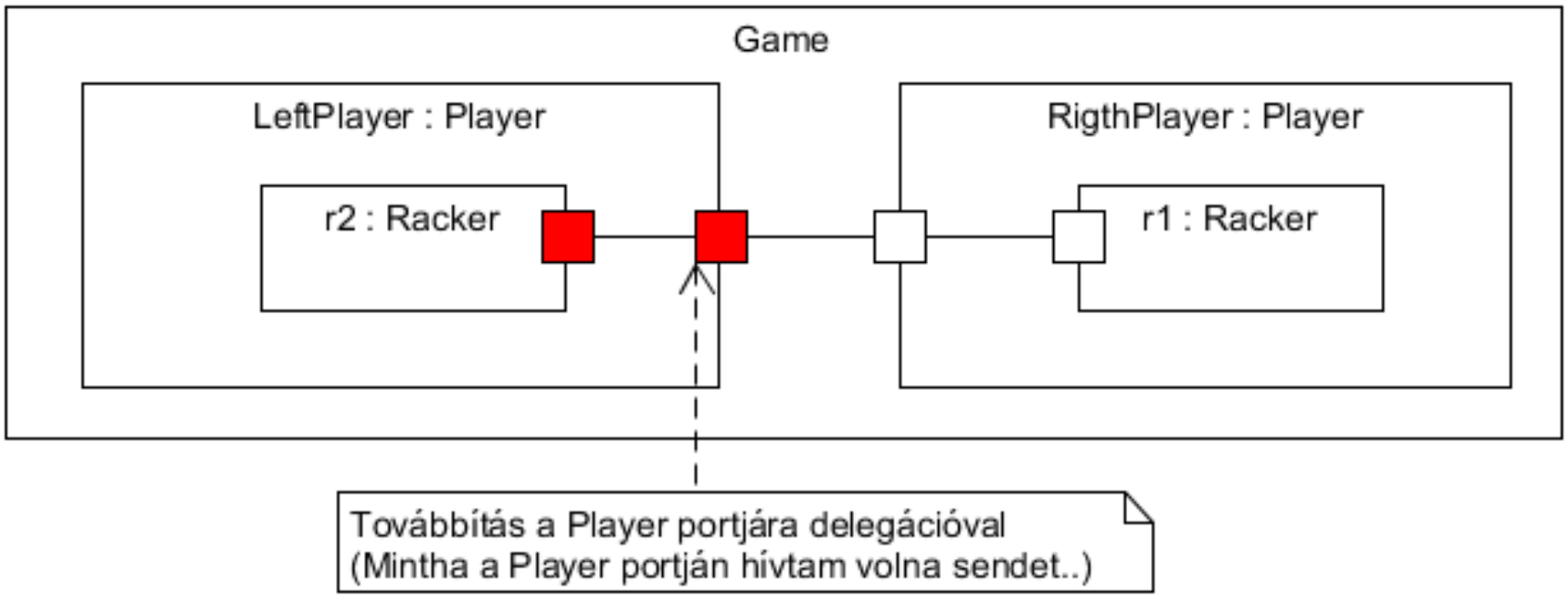
- A port bármilyen üzenetet fogadhat, hol vannak az interfészek?
- Mi a send szemantikája, milyen UML akciónak felel meg?

Interfész portok üzenetküldéssel



- A típushelyességet biztosítottuk.
- A továbbiakban a *send* szemantika érdekes.

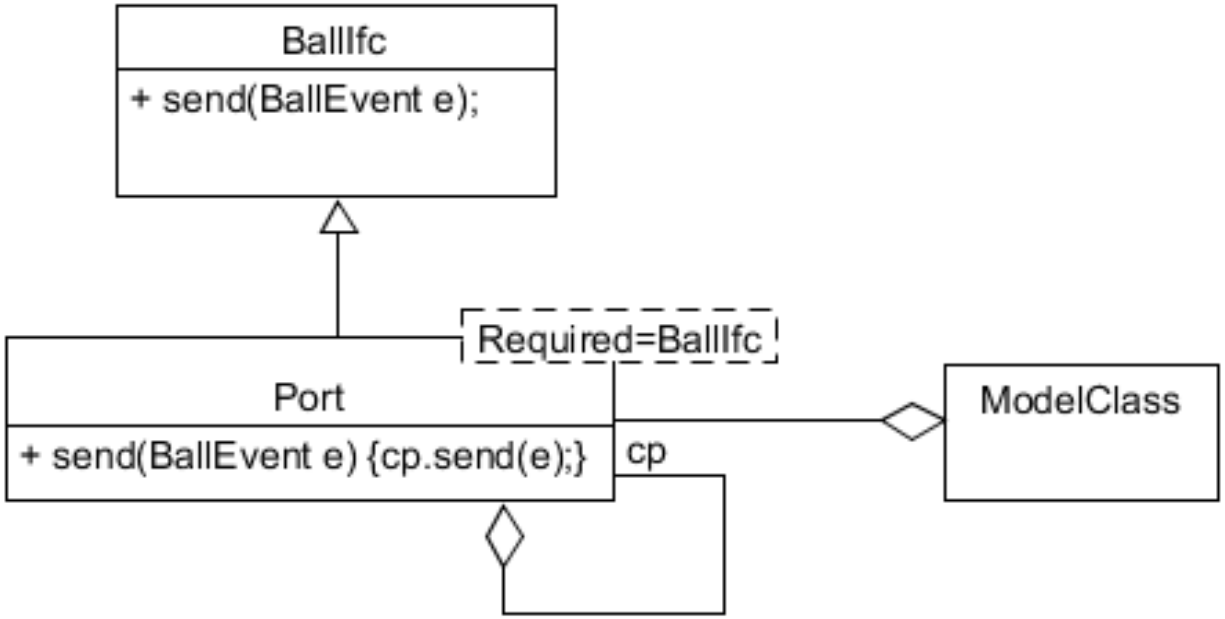
Üzenet továbbítása a szülő komponens felé - Delegáció



Problémák:

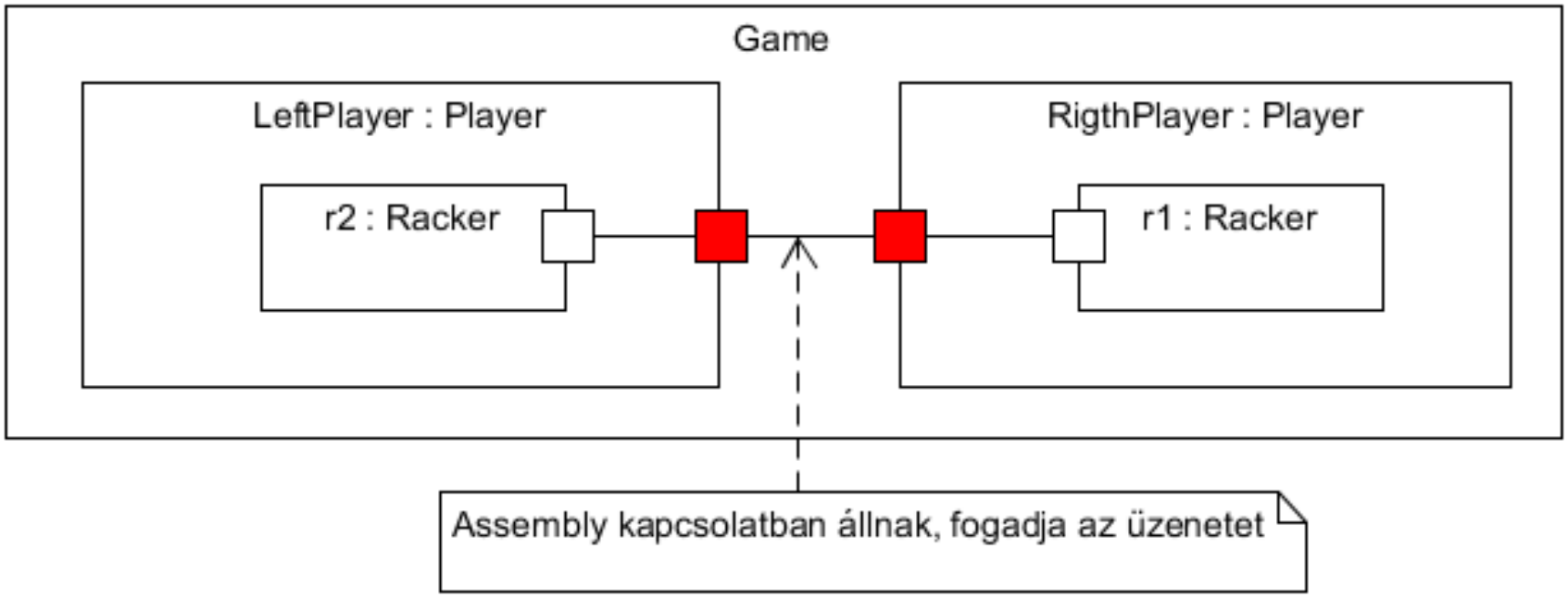
- Hogyan adom tovább az üzenet a szülő felé?
- Hogyan kapcsolom össze a gyerek és a szülő portját?
- Honnan tudom, hogy delegációs kapcsolat áll a két port között?

Naiv kapcsolat referencia tárolás



- Delegációs kapcsolat esetén működik csak.
- A referenciát összekapcsolásnál állítom be, ami már UML-ben is nehezen reprezentálható és értelmezhető.

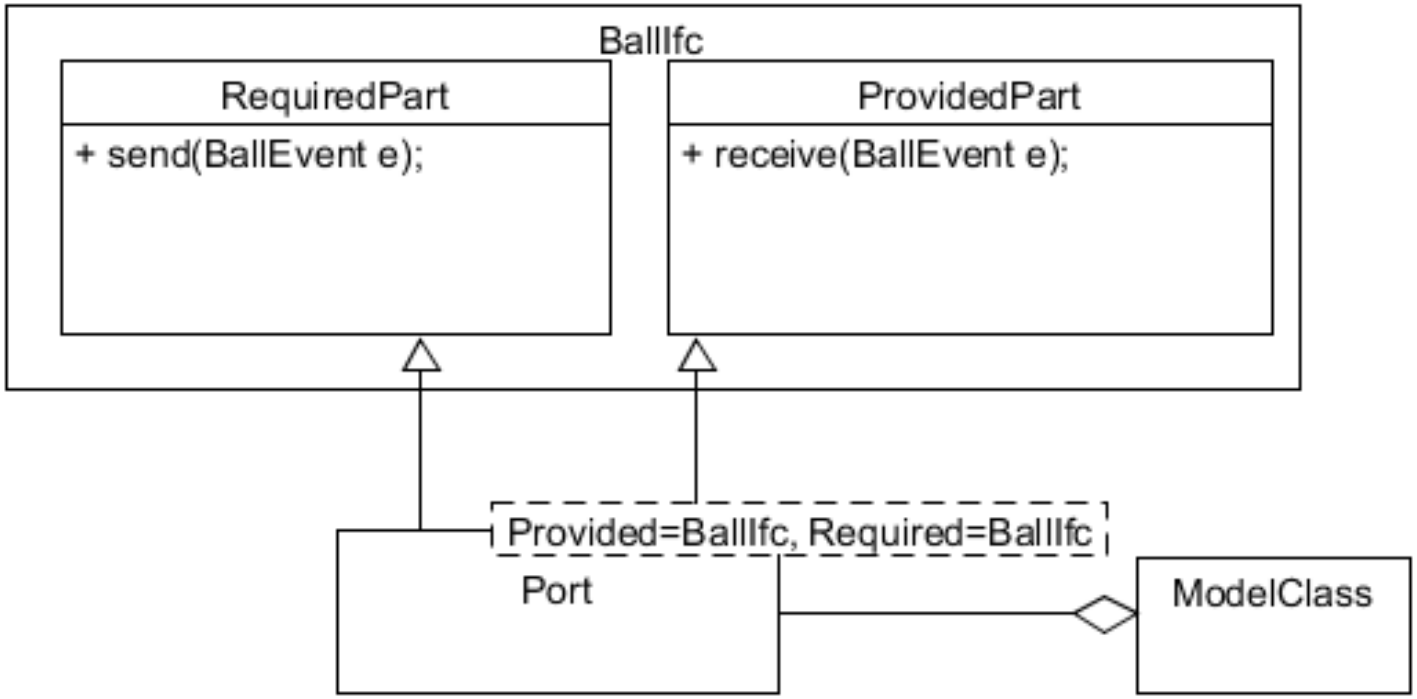
Üzenet átadása testvér komponensek - Assembly



Újabb problémák:

- A *send* másképp viselkedik, mint delegációs kapcsolat esetén.
- A jobb oldali célportnak nem delegálok, hanem fogadja az üzenetet, ami befele áramol tovább. (Az elvárt interfész érdekes számomra)

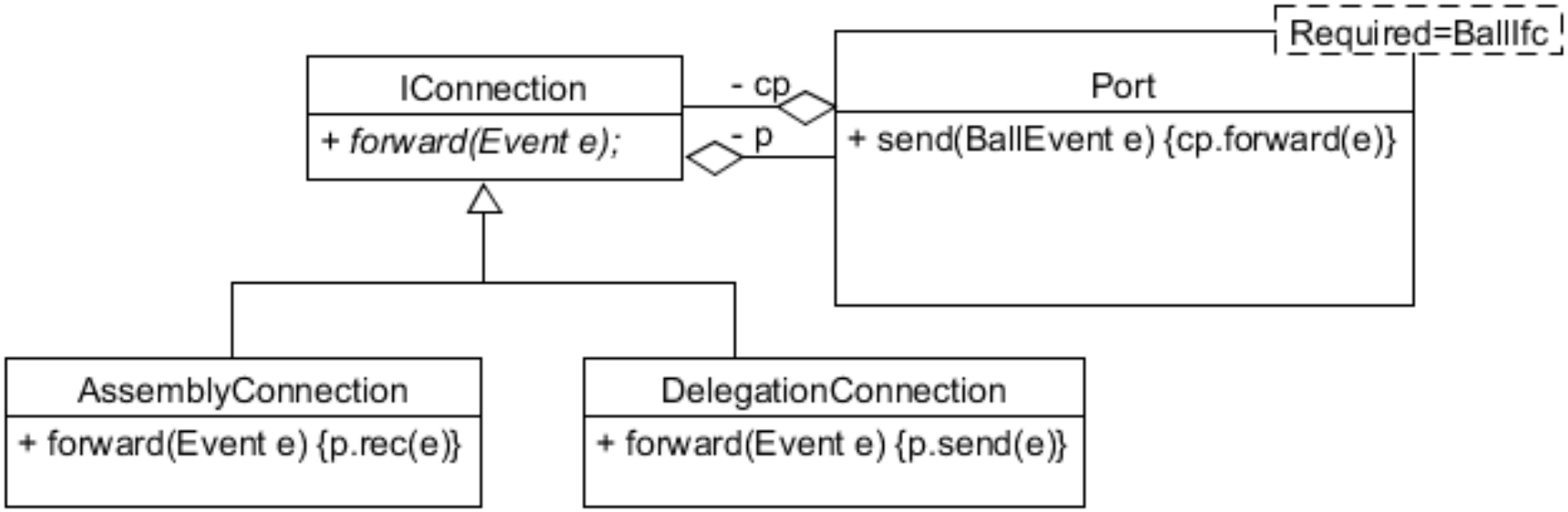
Interfészek szétbontása



- Az PSCS szemantika szerint az üzenetküldés kontextusától függ az üzenetáram iránya (send - belső, receive - külső).
- *TYPELIST* pattern

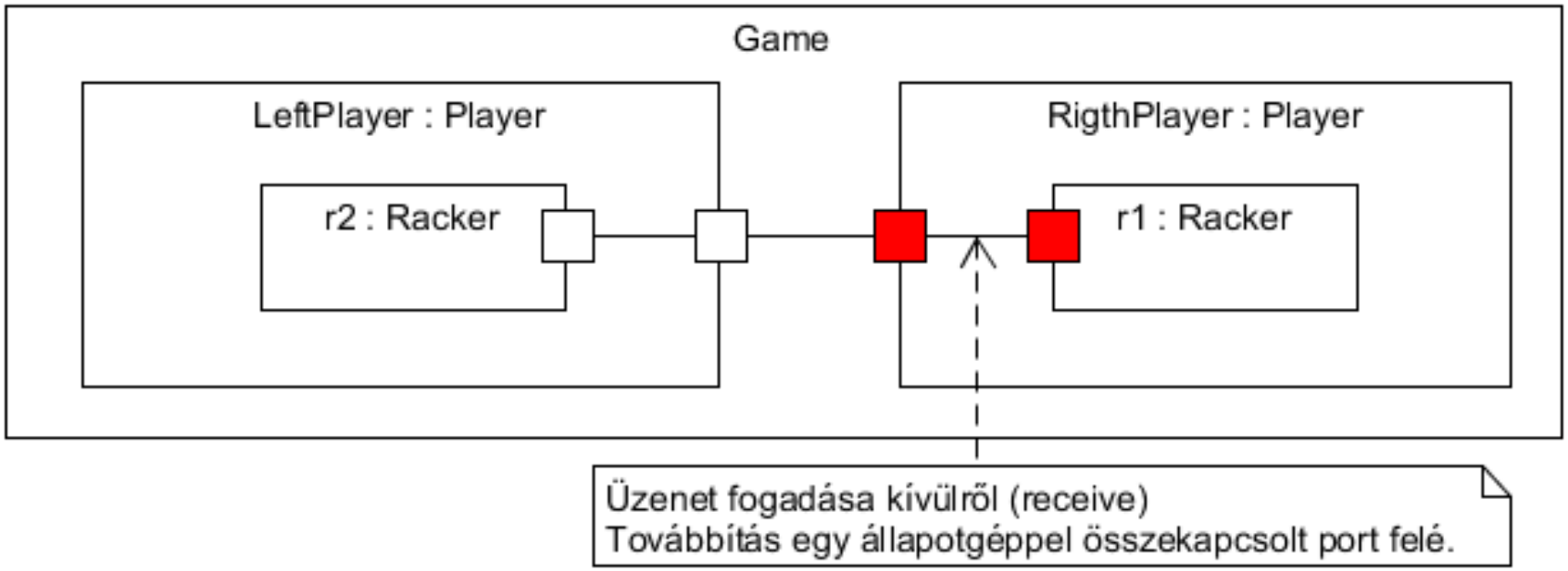
```
GeneralInterface<Ball,..>
```

Polimorf kapcsolat



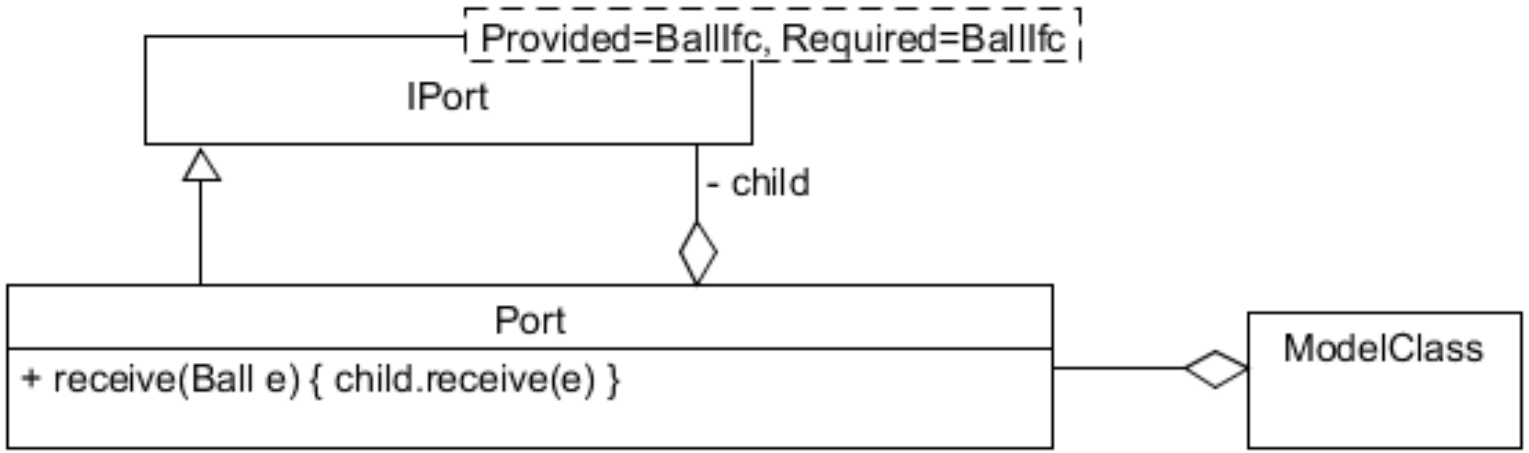
- Tisztább reprezentáció, mint az ömlesztett adattárolás.
- Sablon interfésszel lehet javítani a hatékonyságon (lásd diplomamunka).
- A típuskonzisztenciát az portok összekapcsolásakor kell biztosítani.

Üzenet továbbítása a gyerek komponensek



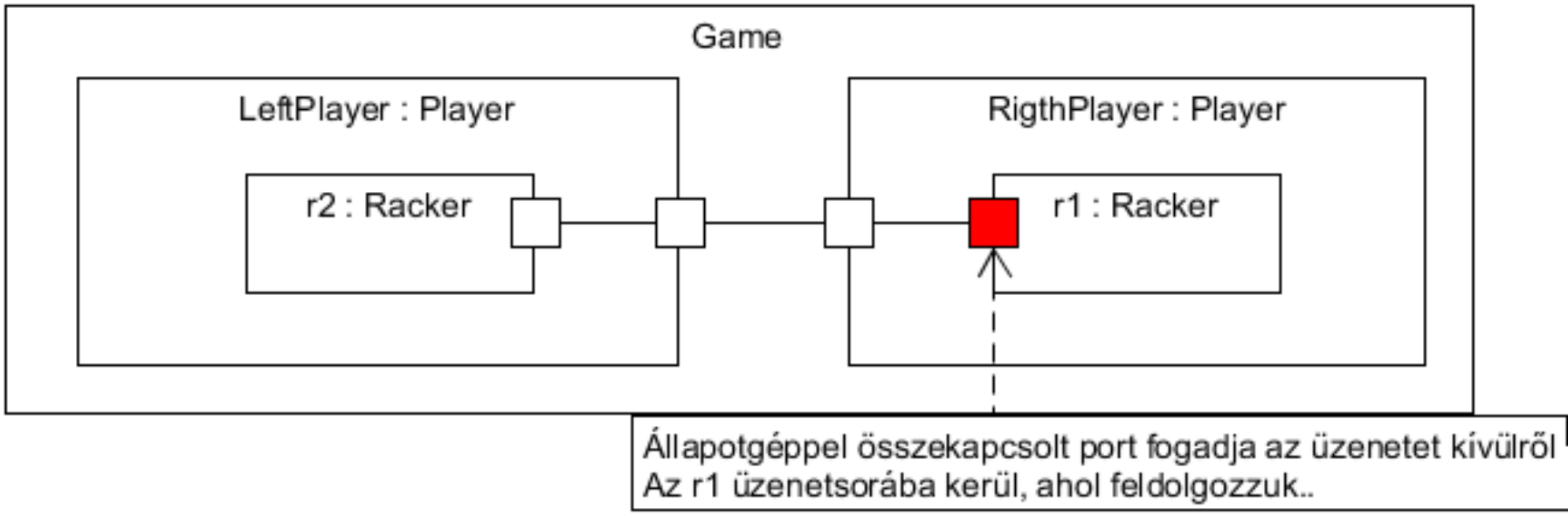
- A delegáció miatt továbbítani kell az üzenetet a belső komponens felé.
- Ez a kapcsolat azonban különbözik az általános kapcsolat referenciától.
- Mintha a gyerek *receive*-jét hívtuk volna..

Gyerek felé továbbítás delegáció esetén



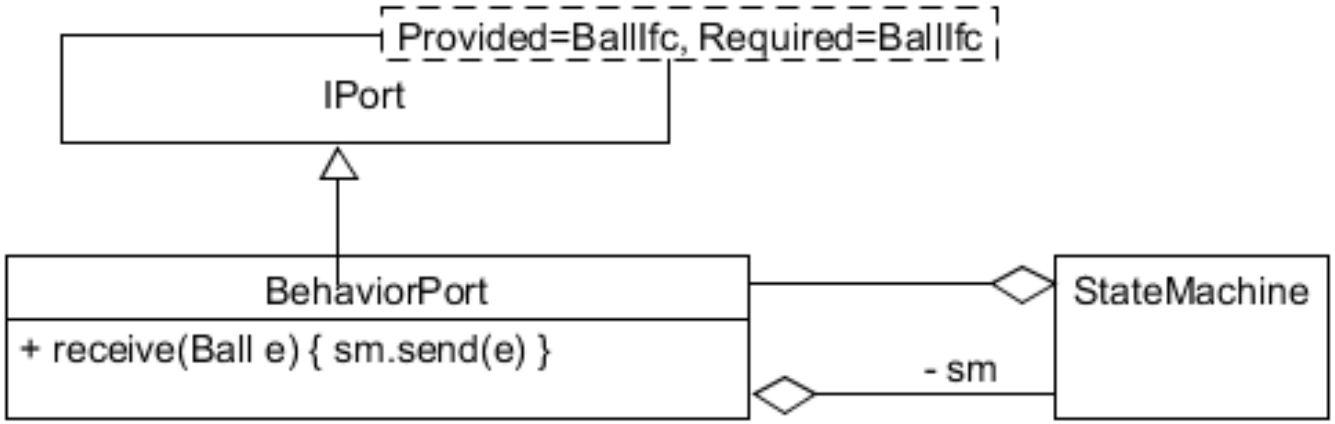
- A gyerek referenciát egy delegációs összekapcsolás állítja be.
- Állapotgéppel összekapcsolt esetén más adatok tárolására és viselkedésre lesz szükségünk, ezért vezessünk be egy általános *IPort* interfészt.

Gyerek felé továbbítás delegáció esetén



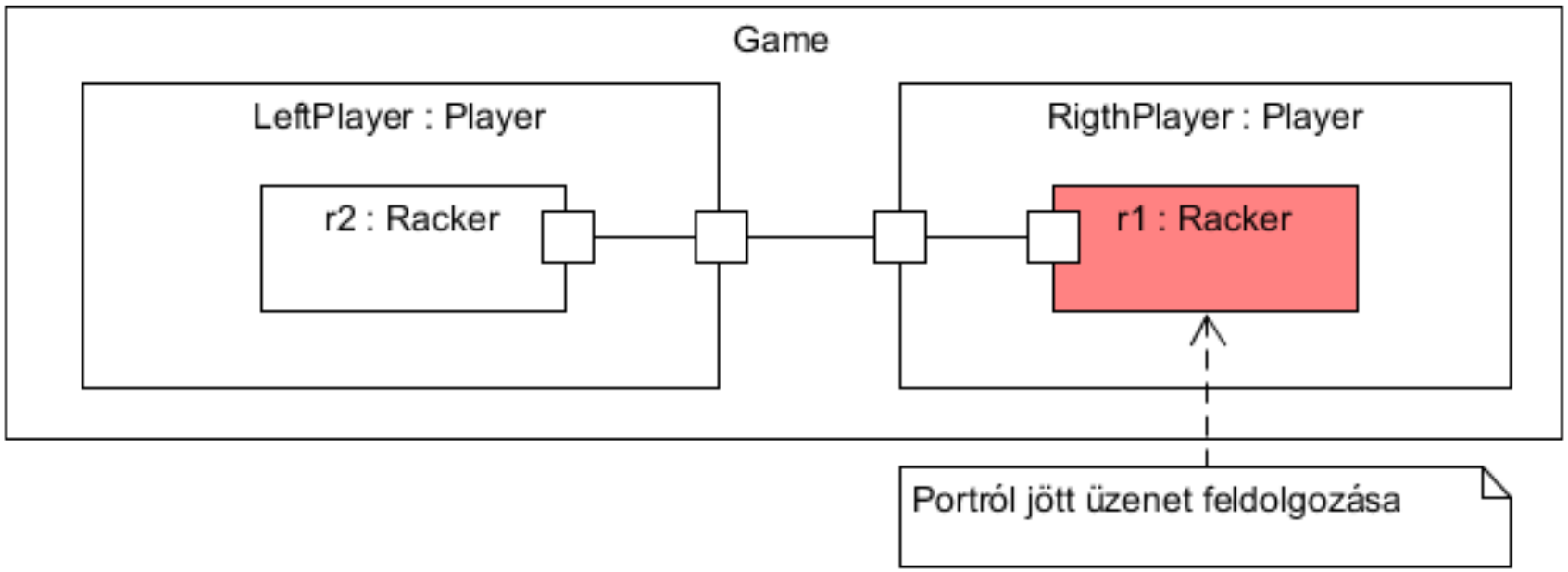
- Gyerek komponens referencia helyett az állapotgépre kell referenciát birtokolnunk.
- Más viselkedés, más adattagok.

Gyerek felé továbbítás delegáció esetén



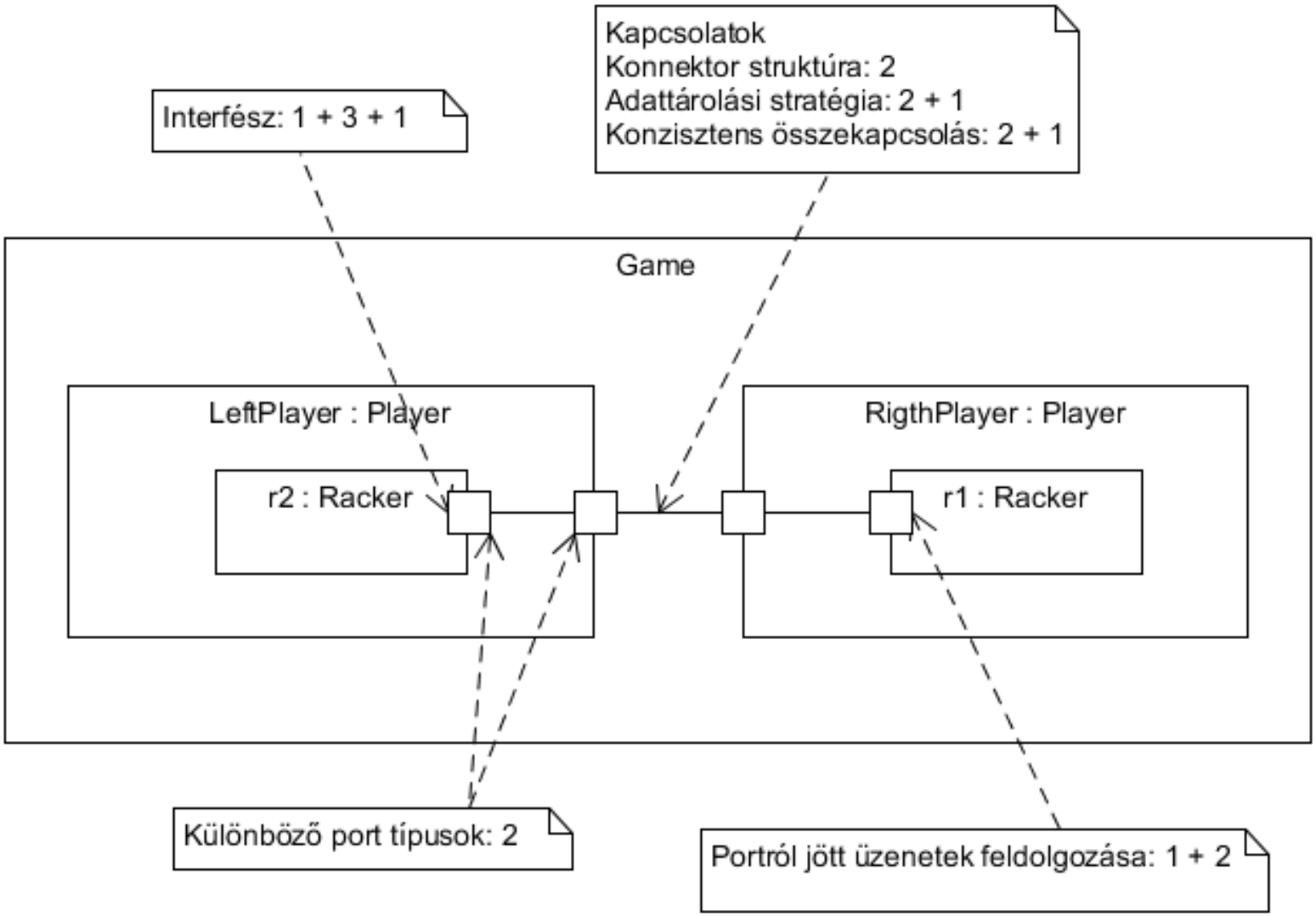
- A *send* hatására bekerül az állapotgép üzenetsorába az *Ball* üzenet.
- Tudnunk kell, melyik portról érkezett az üzenet, amikor átmenetben dolgozzuk fel.

Gyerek felé továbbítás delegáció esetén



- Átmeneteknél (esemény,állapot) mellett megadhatjuk, mely portokról jött üzenetek relevánsak az átmenet szempontjából.
- Állapotgép reprezentálása mint eddig, figyelembe véve, honnan érkezett az üzenet (átmenet tábla bővítése, maszkolás).

Elemezett kódgenerálási stratégiák száma



A kertrendszer kompozíciós elemek támogatásának előnyei

- Generált modell beburkolása egy komponensbe, így a külvilággal interfészekkel tud kommunikálni.
  - Generált kód és külső elemek teljes szétválasztását teszi lehetővé.
  - PI: *FMU* export során a modell egy funkcionális komponens, mellyel portokon keresztül köthetjük hozzá az *FMI* szabványt megvalósító szereplőt.
- Komponensek egyszerűbb párhuzamosítása (környezetfüggetlenség, nem birtoklunk referenciát, kommunikációs csatorna testre szabása).
- Komponens-alapú fejlesztésről szóló előadáson hasonló egy ehhez demonstrációt lehet készíteni a kompozit elemek tisztázásra..

Összefoglalás, eredmények

- Az UML kompozit szabvány alapos értelmezése, a megfelelő reprezentációk és szemantika megtalálása.
- C++ kódgenerálási stratégiák készítése.
- Szabványos modell generálása txtUML modell alapján, egy stratégia implementációja.
- A diplomamunka összefoglalja, hogy miben segítenek a portok a modellezésben (könnyebb párhuzamosíthatóság, jobb FMU wrapper, példamodellek készítése oktatási célokra komponens-alapú oktatáson, stb.).

Köszönöm a figyelmet!