Komponens-alapú UML modellek fordításának vizsgálata

Nagy András

2019. január

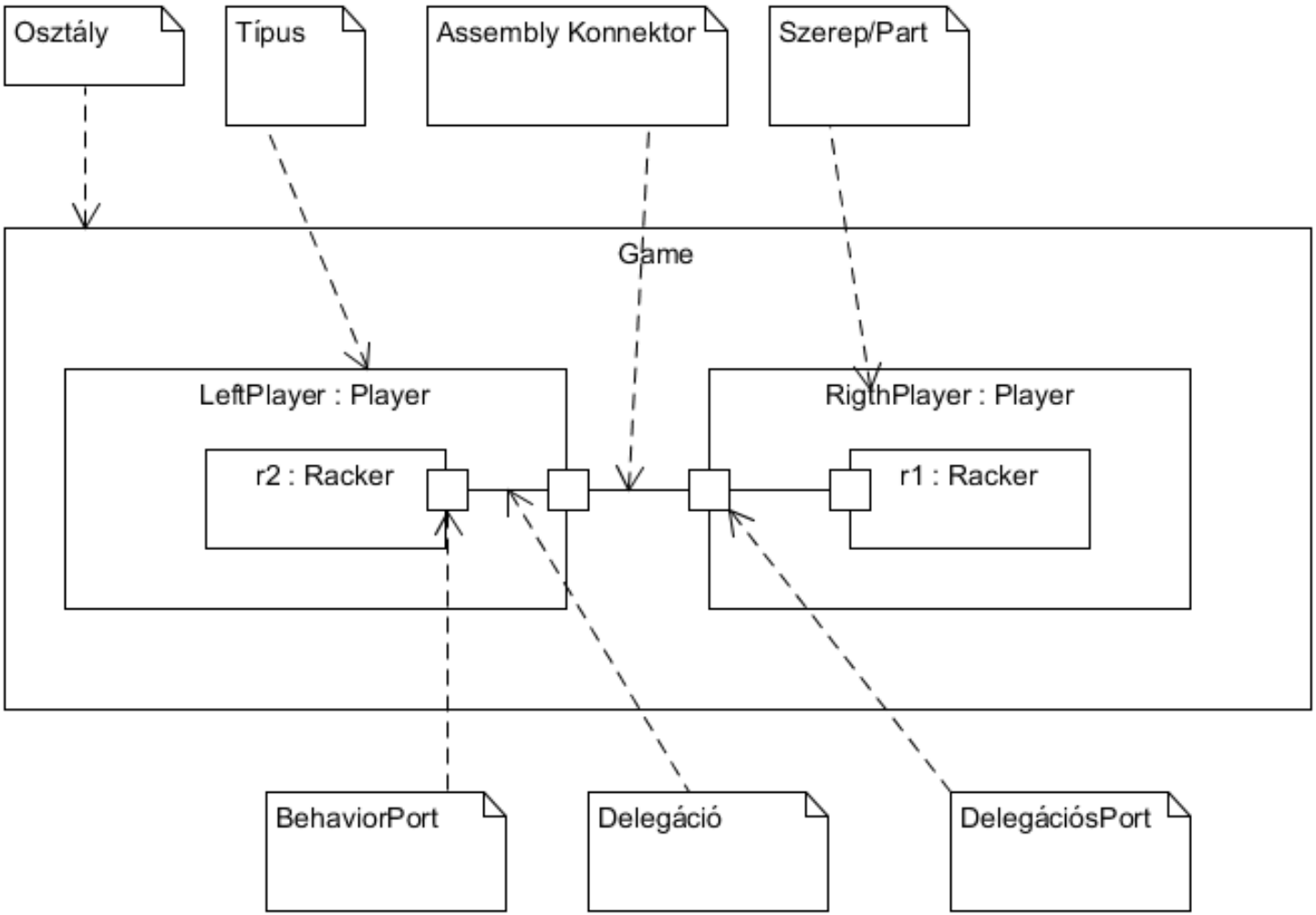
Komponens-alapú megközelítés röviden

- Program szereplőinek izolációja.
- A szereplők függetlenek a környezettől.
- Egymással interfész-portokkal kommunikálnak az egyes szereplők
- Előnyei: független telepíthetőség, explicit interfész függőségek..

Program szöveges modellezése

- *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++ kód generálása.

Példa egy konkrét kompozit struktúrára



Már az UML2-es reprezentáció sem triviális.

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 modell szemantikája??

Példa: Konnektor, konnent UML-ben

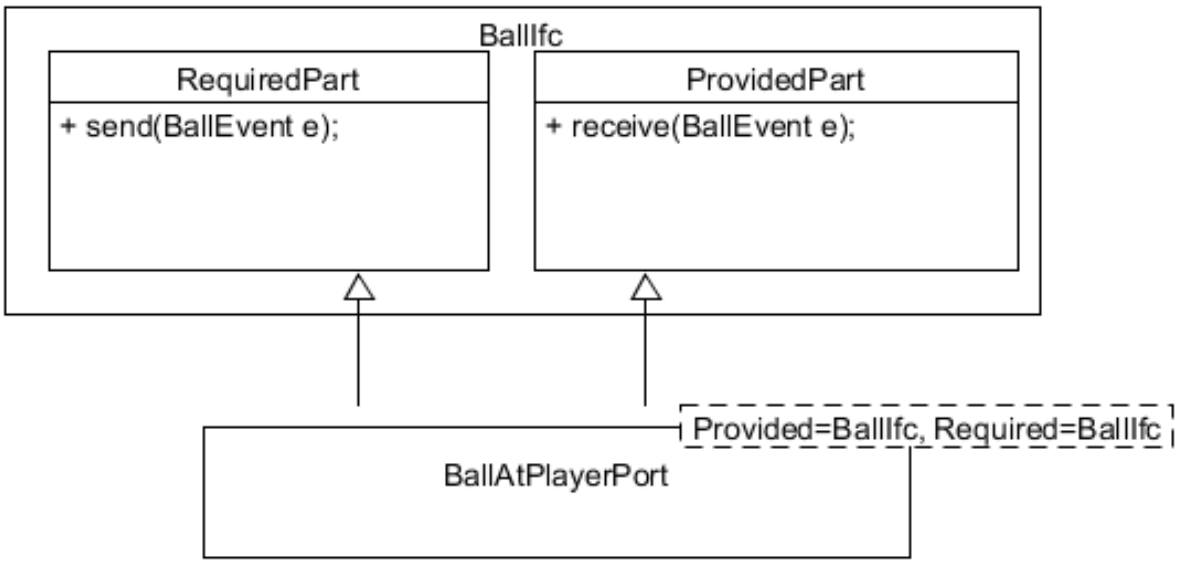
- A *Connector* értelemszerű (de mi az a *type* referencia?)
- A problémás a *connect* művelet
  - Nincs *connect* akció UML-ben
  - *DefaultConstructionStrategy* (de az a szerkezet nem mindig egyértelmű..)
  - A *CreateLinkAction* segítségével összeköthetünk két portot a konnektor típusa mentén (Itt 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 elemezni az egyes elemek generálását (interfész: 3+1, különböző port típusok: 2, konnektor struktúrák: 2, kapcsolódási végpontok tárolása: $2 + 1$, portok összekapcsolásának kifejezése $2 + 1$, üzenetküldés/fogadás: 2, üzenetfeldolgozás: 2).

Példa: Interfész implementált kódgenerálási stratégiája

- Az interfész tartalmazzon annyi *send* műveletet, ahány fogadó művelete van.
- Szedjük szét két részre az interfészt.
- +1: C++ sablon metaprogramozással leszűkíthetjük a kódot. (lásd diplomamunka)



Interfész portok keretrendszere C++-ban

- Bevezetünk egy port osztályt, sablonparaméterei az interfészek.
- Felveszünk minden porta egy port típusú adattagot a megfelelő interfészekkel.
- A portoknak két típusa lehet, a normál port, illetve az állapotgéppel összekapcsolt port.
- A normál port a befutó üzenetet egy gyerek felé továbbítja, az állapotgéppel összekapcsolt port a tartalmazó osztály állapotgépe felé.
- Ez más viselkedést és referencia tárolásokat jelent, melyeket leszármazással a legtisztább megoldani.

Üzenetküldés C++

- Interfész felosztása kintről illetve bentről jövő üzenetek megkülönböztetésére.
- A fent említett *PSCS* szabvány szerinti szemantikának megfelelő művelet generálása. (*send* vagy *receive*)
- Mi ennek a szemantikája, mit jelent ez C++-ban?

Portról jött üzenet feldolgozása C++-ban

- Az üzenet az objektum üzenetsorához fut be.
- Azonban megjelöljük, hogy melyik portról érkezett.
- Állapot-átmeneteknél megadhatjuk, mely portokról érkezett üzenetek érdekesek számunka.
- Feldolgozáskor (esemény, állapot) alapján megkeressük a megfelelő átmenetet, és vizsgáljuk, hogy az átmenet lehetséges portjai között között ott van-e az üzenet portja.

Portok összekapcsolása C++-ban

- A referencia tárolását választottuk.
- Probléma: assembly vagy delegációs kapcsolatban áll a referenciával?
- Megoldás: Csomagoljuk be a port referenciát egy kapcsolat osztályra, mely eldönti, hogy a kapcsolatban álló portnak mely műveletét kell meghívni üzenetküldés/üzenetfogadás esetén. Ennek leszármazással két típusa van, a *DelegationConnect* illetve az *AssemblyConnect*
- *connect* műveletnél ezeket a referenciákat kell kölcsönösen kitölteni, ahol megadhatjuk annak a konnektornak a típusát, mely szerint összekapcsolunk, ez validálja az összekapcsolás helyességét.

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!