Komponens-alapú UML modellek fordításának vizsgálata

Nagy András

2019. január

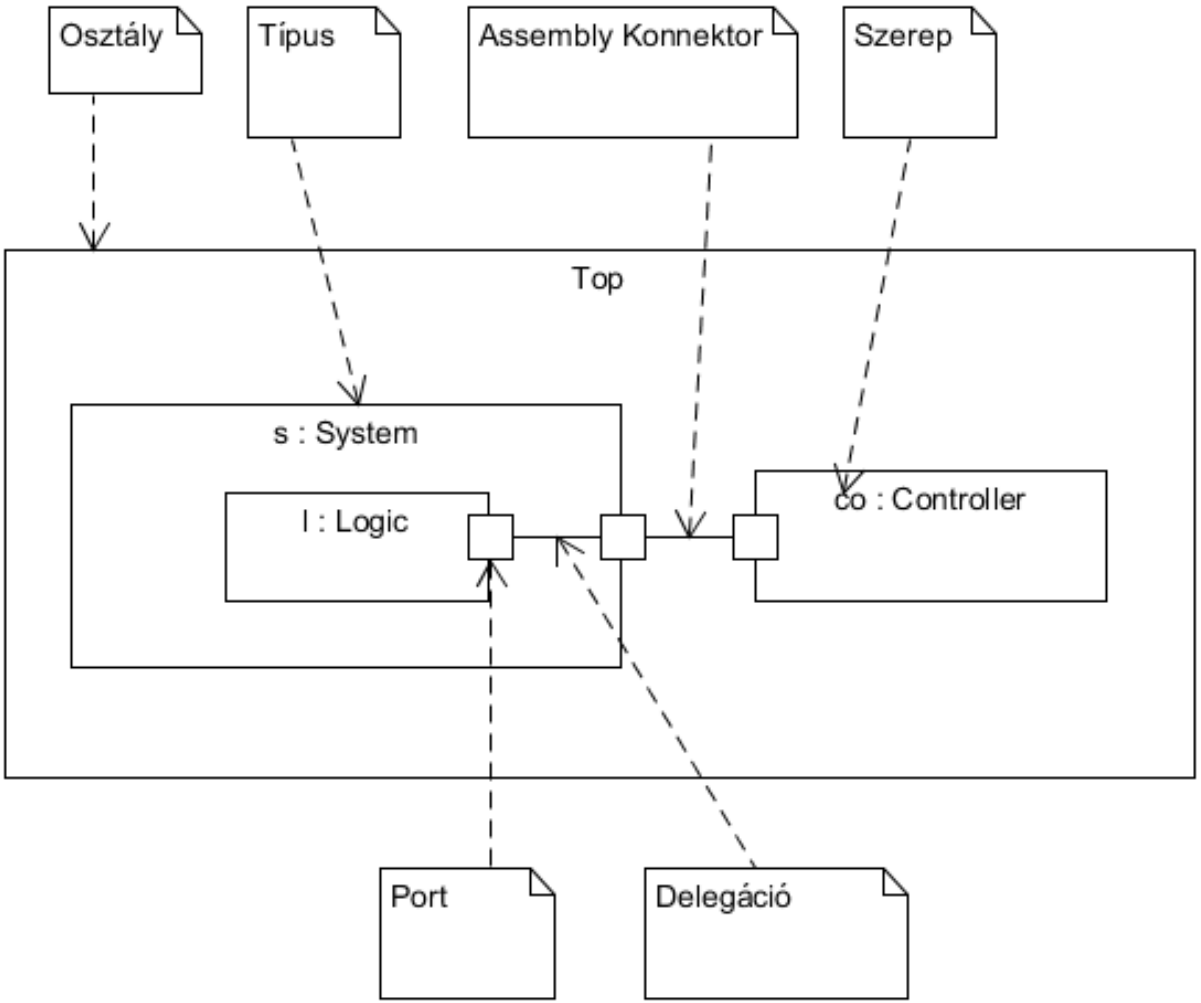
Komponens-alapú megközelítés

- Program szereplőinek izolációja.
- A szereplők függetlenek a környezettől.
- Egymással interfész-portokkal kommunikálnak

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.

Kompozit elemek bemutatása



Már az UML2-es reprezentáció sem triviális.

Főbb reprezentációs problémák

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

Interfész kódgenerálás stratégiái

- Nincs interfész.
- A forráskód validációja kiszűrni a nem megengedett műveleteket.
- Triviális megoldás, nem kell kódgenerálási stratégia.
- Külső kommunikáció esetén olyan üzeneteket küldhetünk a porra, amit nem szabadna.

Interfész kódgenerálás stratégiái

- Jelző osztály generálása az interfész nevével.
- A fogadó műveletek szignáljai leszármaznak a jelző osztályból.
- A port *send* művelete a jelző osztály leszármazottjait várja.

Interfész kódgenerálás stratégiái

- Előbbi megoldás nem elég általános, portokra korlátozódik.
- Az interfész ne üres osztály legyen, hanem ő tartalmazza a *send* műveleteket.
- Pontosan annyi különbözőt, ahány fogadó szignálja van.
- C++ sablon metaprogramozással leszűkíthetjük a kódot. (lásd diplomamunka)

Interfész portok reprezentálása 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 UML-ben

- Használjuk a *SendObjectAction OnPort* referenciáját.
- A szemantika a kontextustól függ: Ha az akció végrehajtója megegyezik a célobjektummal, akkor üzenet küldésről beszélünk. (belülről kifele továbbítjuk az üzenetet)
- Egyébként pedig üzenet fogadásról. (Kívülről befele továbbítjuk az üzenetet)

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.

Konnektor, konnent UML-ben

- A *Connector* értelemszerű
- A problémás a *connect* művelet
  - Többféle megoldás, nincs *connect* akció UML-ben
  - A kapcsolatok az osztályszerkezet konstruálásakor jönnek létre.
  - *PSCS* szabvány szerint, ha nem adunk meg konstruktor művelet, az alapértelmezett konstrukciós stratégia *DefaultConstructionStrategy* fog életbe lépni.
  - A szerkezet nem mindig egyértelmű.
  - A *CreateLinkAction* segítségével összeköthetünk kért portot a konnektor típusa mentén (A típus asszociációt portok típusai alapján generálhatjuk).

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.

Köszönöm a figyelmet!