Komponensek

Portok

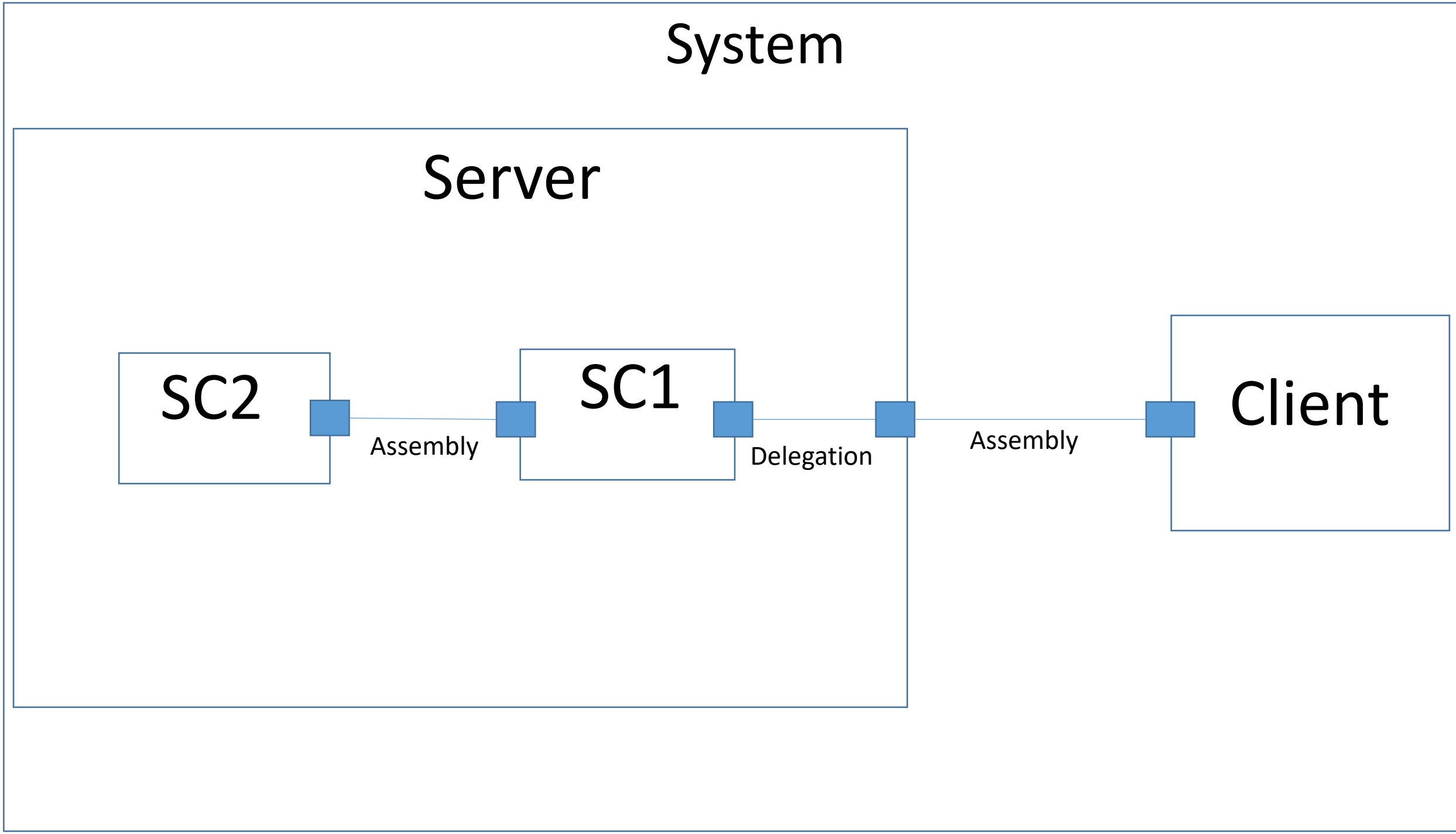
- Logikai egység (pl. class) adattagja, kommunikációs pontot reprezentál
- Lehet egy primitív típus is akár (pl. int), de lehet típusozni valamilyen interfésszel is -> ez van a mi esetünkben is
- A szolgáltatott interfésszel típusozzuk, az elvárt interfészt pedig használja (using reláció) az adott port.
- A szolgáltatott interface azt mondja meg, milyen üzeneteket fogadhatunk (receve), az elvárt, hogy milyeneket küldhetünk (send)
- Jelnleg a c++ exportbe a required imterfacből származik le egy port, így kap olyan receptionöket, melyekre szüksége van

Port típusai

- Behavior port:
- Össze van kötve az objektum állapotgépével
- Végső soron egy behavior portba futnak be az üzenetek
- Átmeneteknél egy új dimenzió, hogy melyik portról jött az üzenet. (UML-ben többet is felsorolhatunk, txtUML-ben csak egyet..)
- Port: Nincs összekötve az objektum állapotgépével
- Általában két magasabb szintű komponens között tartja a kapcsolatot, és lefele delegál. (Mivel üzenet fogadásnál nem tud állapotgép felé közvetíteni üzenetet, így csak egy belső komponens portjára tud)

Connectinök

- Kétféle kapcsolat létezik: Delegation, illetve assembly
- Delegation: Azonos interfészű portokat köt össze, amikor egy portra küldünk egy üzenetet, azt delegálja a a kapcsolat másik végén lévő port számára. Szülő és gyerek között állhat fent.
- Assmebly: A szolgáltott és elvárt interfészeknek azonosnak kell lennie. (Ami az egyik oldalon az elvárt, a másik oldalon a szolgáltaott interfész) Kölcsönös kommunikáció valósul meg. Azonos szinten lévő komponensek között állhat csak fent.
- Az exporterben két külön osztályként jelenik meg, így virtualizáció segítségével döntjük el, hogy üzenetküldésnél delegációra vagy fgadásra van-e szükség


System
Server
SC2
Assembly
SC1
Delegation
Assembly
Client

UML-es reprezentáció nehézségek

- Nincs olyan akció, amely a portra küld üzenetet. Azonban a send object action-nek megadhatunk egy portot, amin keresztül üzenetet szeretnénk küldeni. (Minek is felel ez meg, send vagy receive – kifele vagy belefe küldünk üzenetet?) Másik alternatíva, hogy a tartget maga a port, mint property.
- Connect: Több lehetséges implementáció techicailag, kérdéses, hogy melyik a helyes
- Értékül adjuk egymásnak a két portot, amit össze akarunk kötni (mivel a port property, így megtehetjük): Techinkailag lehetséges, de helytelen, nem vesszük figyelembe a connectort. (ez van most implementálva)
- Automatikusan generáljuk a connectecet a connectorok és struktúra alapján, ergo UML-be meg sem jelenik explicit a connect. (Nem mindig egyértelmű ez a struktúra nálunk..)
- A szabvány szerint egy connector tipusozható egy asszocicióval (Probléma: mi ez az asszociűció?) -> Ekkor gond nélkül megy a connect, mivel a creat link akció működik
- Jelenleg 1-1 port asszociációkat támogatunk

Miért jók a portok?

- Teljesen függetlenít egy komponenst a külvilágtól
- Explicit asszociációk helyett connectorok, melynek a végpontja változhat
- Nincs közvetlen referencia egy objektumra: Az álom az volt, hogy kommunikációs protokoll legyen akár testre szabható (pl. socet)
- Problémák: több probléma merült fel, nagyon nehéz garantálni a szálbiztonságot, a szülőnek mindig van referenciája a gyerek objektumra a szabvány szerint, így szinkronhívások lehetnek (Gábor ttdk munkája ilyesmiről szólt, de végül nem oldotta meg a problémát..)