Megjelent: Magyar Tudomány Napja,
Doktoranduszok Fóruma,
Miskolc, 1997. nov. 6, pp. 11-15

NC program ellenorzo szimulátor fejlesztése objektumorientált módszerrel

Hornyák Olivér
I. éves doktorjelölt
Miskolci Egyetem, Alkalmazott Informatika Tanszék

A szoftver termék, sot egyre inkább tömegtermék, így tudományos ismeretek alkalmazásával történo gazdaságos eloállításának mikéntjét indokolt technológiának nevezni. Kezdetben programozási technikák, késobb módszerek aztán módszertanok, paradigmák jelentek meg. Az egyik legnépszerubb módszertan az objektum-orientáltság. Az objektum-orientált módszertan alkalmazásával kifejlesztendo rendszert együttmuködo objektumokkal modellezzük; a tervezés és az implementáció során ezeket az objektumokat megvalósító programegységeket alakítunk ki.

Az objektum a rendszer egyedileg azonosítható szereploje, amelyet a külvilág felé mutatott viselkedésével, belso struktúrájával és állapotával jellemezhetünk. Az objektum a rendszeren belül olyan szerepet játszik, amilyenre a rendszer feladatainak ellátásához szükség van. A többi szereplo nem lát bele az objektumba, a struktúrájára és állapotára csak a viselkedésébol következtethetünk. Ezt az információelrejtési formát egységbe zárásnak nevezzük.

Az egyes objektumok egymásnak küldött üzeneteken keresztül kommunikálnak. Minden objektum az üzenetek egy meghatározott készletét képes elfogadni és értelmezni. Az üzenet két fontos komponenssel rendelkezik: van neve és vannak paraméterei. Ezek a paraméterek az üzenet aktuális tartalmának tekinthetok. Az objektumok kölcsönhatásának vizsgálatakor eltekinthetünk a történés folyamatának részleteitol. Az ilyen pillanatszeru történést eseménynek nevezzük. Az üzenetek és események együttes alkalmazása is elképzelheto, amivel finom különbségeket tudunk tenni.

Az objektumhoz érkezo üzenet hatására az objektum valamilyen cselekvést hajt végre. Ha egy objektum képes fogadni egy adott nevu üzenetet, akkor erre az üzenetre az üzenet neve által meghatározott metódus végrehajtásával reagál. Az objektum viselkedésének pontos leírása, implementációja metódusainak kódjában található. Az üzenet tehát megmondja, hogy MIT kell csinálni, a metódus pedig azt, hogy HOGYAN. Az objektum azonos tartalmú üzenetekre különféleképpen tud reagálni az állapotától függoen. Az objektumba tehát a viselkedésén kívül bele kell foglalni az állapotinformációt. Az objektum állapotát csak az objektum által végrehajtott metódusok változtathatják meg, kívülrol ezek nem is láthatóak. Az objektum állapotát attribútumai tárolják. Az attribútumok értékei az objektum élete során változhatnak, ezért változókkal jelenítjük meg azokat.

Az objektum tehát olyan modellje a világ egy részének, amely a számára kívülrol érkezett üzenetekre reagálva valahogyan viselkedik. Az objektumnak kívülrol nem látható belso statikus struktúrája van, amely az állapotok értékét rögzíto attribútumokkal definiálható. Beszélhetünk az objektum állapotáról, amely a belso struktúrát egy adott pillanatban kitölto értékek halmaza. A megegyezo

viselkedésu és struktúrájú objektumok egy közös minta alapján készülnek, amit osztálynak nevezünk. Az objektum tehát az ot definiáló osztály egy példánya.

Az objektum-orientált tervezés az analízisnél kezdodik. Az analízis során a feladat definíciójából az alapveto objektumok, és a rájuk ruházott felelosségek összegyujthetok. Az analízis során eloállíthatóak absztrakt modellek, amelyeket le kell képeznünk a fizikai rendszerünk által biztosított szolgáltatásokra. Ezt a leképzést nevezzük tervezésnek. A tervezésnek három szintjét különböztetjük meg:

- architekturális tervezés: a létrehozandó program egészét érinto kérdésekben döntünk
- Külso interfész tervezése: a külvilággal való kapcsolattartás módját és részleteit írja le
- részletes tervezés: az osztályok és az objektumok specifikációja olyan módon, hogy azok együtt tudjanak muködni.

A tervezés során nehézséget jelent, ha a megvalósítandó program több párhuzamos folyamatból, taszkból áll. A különbözo folyamatok közötti kommunikációt lényegesen nehezebb implementálni, mint a taszkon belüli párbeszédet.

Hasonlítsuk össze az objektum-orientált módszertant a strukturált programozással ! A strukturált más néven funkcionális stratégiák funkcionális szempontból tervezendok. A munka felülrol indul, és a részletek feltárásával halad lefelé. A rendszer állapota oszthatatlan egész, egyetlen helyen van számontartva, és ehhez a helyhez (állapotleíráshoz, adatbázishoz) minden funkció hozzáférhet. A tiszta strukturáltság betartásának a hatékonyság csökkenése lehet az eredménye. Elofordulhat, hogy egy magasabb szintu rétegben olyan egyszeru muveletekre is szükség van, amelyre az alacsonyabb rétegekben is számítanak. Ha ilyenkor következetesen betartjuk a strukturált programozás elveit, akkor több rétegen keresztül is csupán közvetíto szerepet betölto funkciókat veszünk fel. A rendszer állapotváltozóihoz való közvetlen hozzáférés rendkívüli veszélyeket rejt, és nehezen felfedezheto hibák forrásává válhat. Az objektum-orientált rendszereknél a rendszer állapota részekre tagolódik, és egy-egy ilyen állapotrész a megfelelo objektum hatáskörébe tartozik.

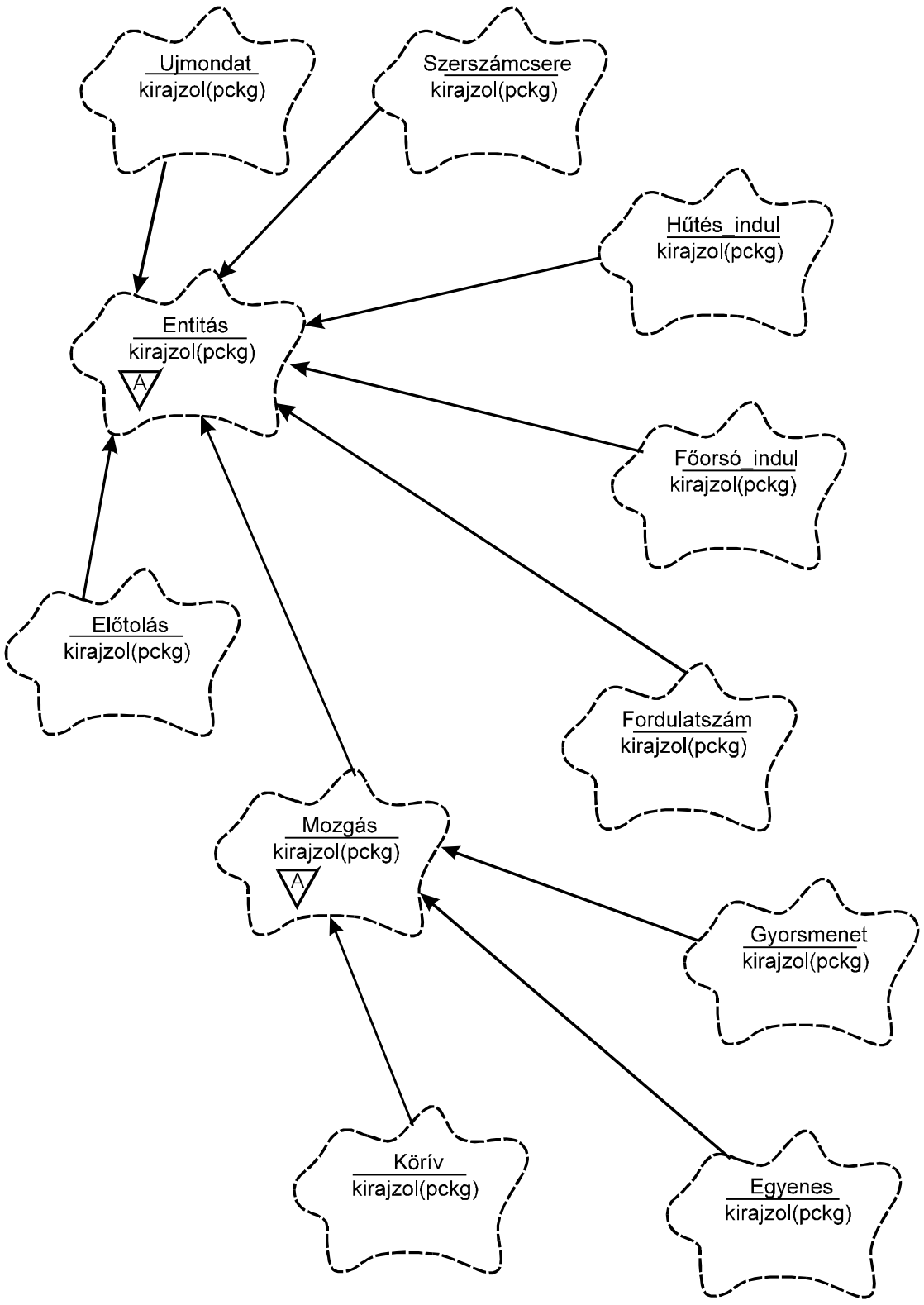
Az általam kifejlesztett NC program ellenorzo szimulátor az ISO 6983 szabványnak megfelelo NC programokat képes ellenorizni. A program két részre bontható: az NC program feldolgozó és az így feldolgozott programot grafikus animációval megjeleníto részre.

A feldolgozás elso lépése a szintaktikai ellenorzés. Szintaktikailag hibás program megadása esetén hibalista készül, ami alapján a hibák kijavíthatóak. Szintaktikailag helyes NC programnál a feldolgozás az interpretálással folytatódik. Az interpreter megírása bonyolult feladat, például az NC programbeli X szó jelenthet abszolút vagy relatív elmozdulást, illetve várakozási idot a programsortól függoen. Az interpreter egy entitásokból álló listát készít. Ilyen entitás például az egyenes és a kör interpoláció, a gyorsmenet, a szerszámcsere, a fordulatszámváltás, stb.

A szimuláció ezeknek az entitásoknak a megjelenítése grafikus animációval. A szimuláció nem feltétlenül jelent animációt, például egy termelési folyamatot Gant-diagrammokkal is jellemezhetünk. Az 1. ábrán az entitások öröklodési hierarchiája

látható.(A dolgozatban a Booch-féle [1] jelölésrendszert használtam.) A központban egy olyan osztály található, melynek minden metódusa virtuális, azaz nincs „munkavégzo” függvénye. Az ilyen osztályokat absztrakt osztályoknak nevezzük. Az effajta származtatás elonye az, hogy az egyes entitások mindegyike a közös *Entitás* osztályból származik, és így egy közös láncolt listán tárolhatók. Az ábrán megfigyelheto a többszörös öröklodés a *Mozgás* osztály esetében.

Az egyes entitás osztályoknál csak a kirajzol metódust tüntettem fel, az animáció implementálása ezekben a függvényekben valósul meg.

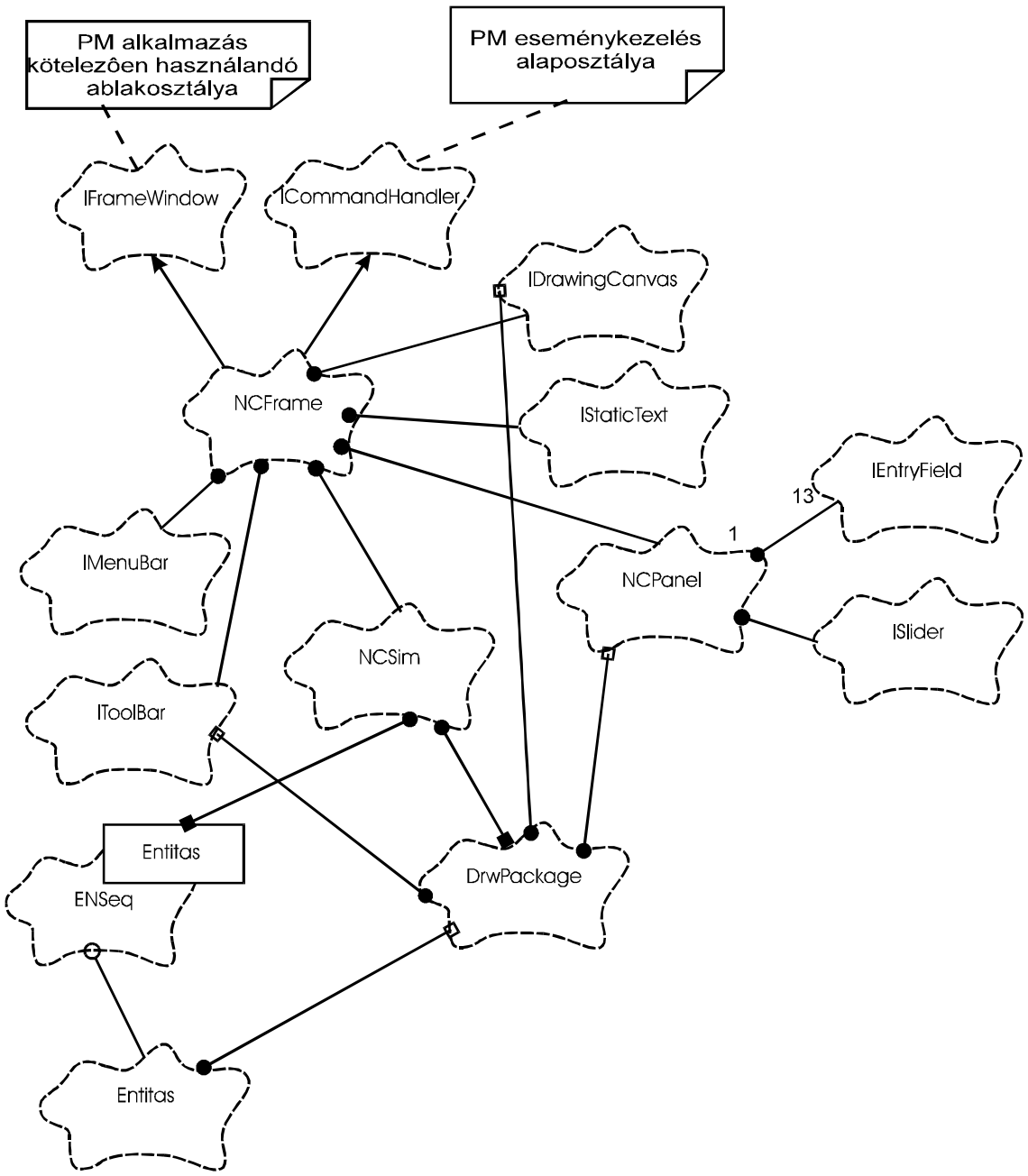


1. ábra

Az entitások osztályhierarchiája

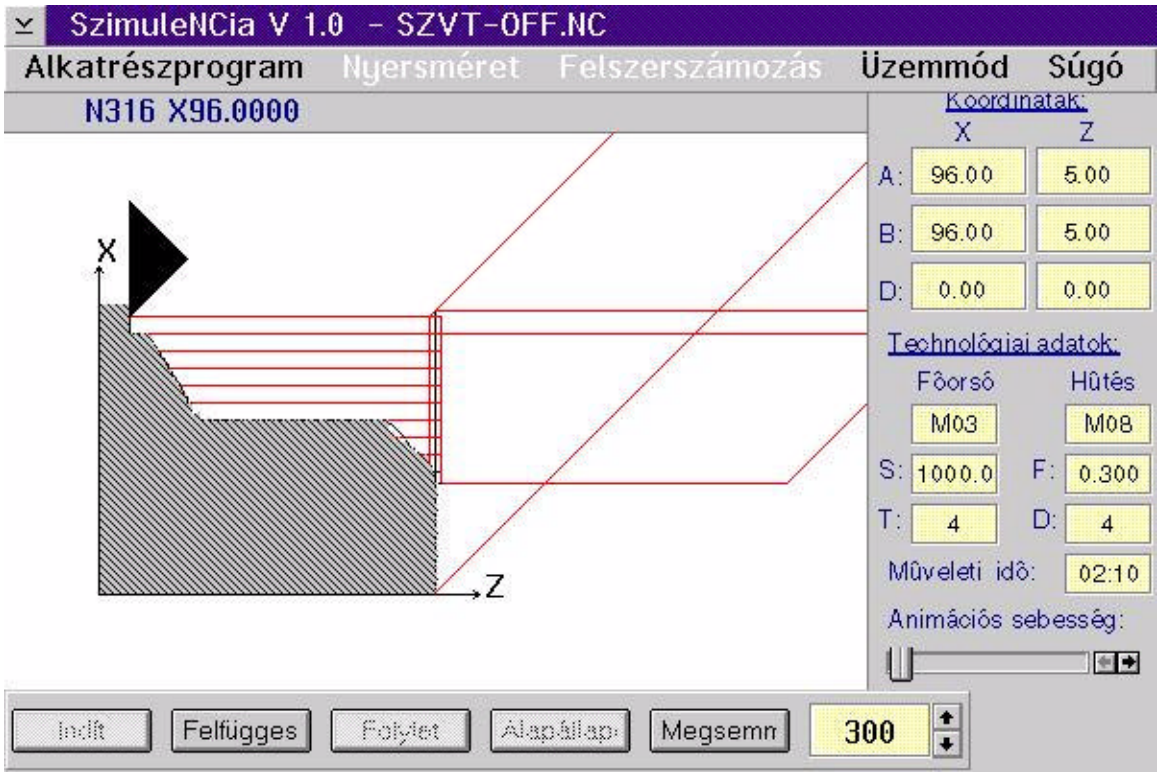
Az objektumorientált módszerekkel kifejlesztett rendszert együttmuködô objektumokkal modellezünk. A 2. ábrán a fôbb objektumokat és azok kapcsolatrendszerét láthatjuk. Az egyes objektumok csak meghatározott interfészeken keresztül kommunikálhatnak egymással, ezért a kapcsolatrendszer megtervezése általában bonyolultabb feladat, mint a strukturált programozásnál. A

befektetett többletmunka azonban a karbantartásnál, a szoftver továbbfejlesztésénél megtérül.Ez azért különösen fontos, mert szoftvertermékek esetében a karbantartás költsége elérheti a teljes költség 70 %-át is !



2. ábra
A fôbb objektumok

A 3. ábrán a szimulátor látható muködés közben. A forgácsoló szerszámot a háromszög alakú lapka jelképezi. Látható a szerszám vezérelt pontja által bejárt út, illetve a munkadarab pillanatnyi állapota. A jobboldali panelen a vonatkozó kapcsolási információk leolvashatók, és a menü alatti státuszsorban megjelenik az éppen végrehajtás alatt álló NC programsor is.



3. ábra
Az NC szimulátor

A rendszer továbbfejlesztésével a tervem az, hogy a kinyert geometriai információk felhasználásával olyan szimulátort fejlesszek ki, amely a forgácsolásnál fellépô erôhatásokat szimulálja. Ilyen jellegu szimulátorok fejlesztése ma még kutatások tárgya.

Irodalomjegyzék:

[1] Booch G.: Object oriented analysis and design with application; Second Edition, Benjamin/Cummings, 1994.
[2] Kondorosi K.-László Z.-Szirmai-Kalos L.: Objektum-orientált szoftverfejlesztés;ComputerBooks Budapest, 1997.