OBJEKTUM ELVU INTERAKTÍV RÚDSZERKEZET SZÁMÍTÓ PROGRAM MODELLEZÉSE

Hunyadi Mátyás[*]

RÖVID KIVONAT

A cikk célja bemutatni egy objektum orientált elven tervezett és megvalósított interaktív rúdszerkezet számító program modelljét. A cikkben bemutatásra kerül a program célja, fobb elemeinek osztály- és állapotdiagramjai.

1. BEVEZETÉS

A manapság szerkezettervezésben használatos rúdszerkezet számító és véges elemes programok nagy része régen lett kifejlesztve, így elkerülhetetlenül csatlakoznak hozzá FORTRAN nyelven készített programrészek. A FORTRAN programozási nyelvet eloszeretettel használják még ma is mérnökök, mivel a mérnöki gyakorlatban használt matematikai fogalmak (mátrix, vektor) meg vannak benne valósítva. A nyel hátránya az adatorientáltság, vagyis, hogy FORTRANban megírt program a tárolt adatok ábrázolására erosen ráépül, ezáltal nehézkes a továbbfejlesztés. Például, megmaradva a mérnöki gyakorlatnál, egy síkbeli keretszámító program lineáris egyenletrendszer megoldó rutinja nagyon erosen kapcsolódik a feladat kétdimenziós voltához, háromdimenziós rúdszerkezet lineáris egyenletrendszer megoldására képessé csak nagy változtatások árán lehet tenni. Emiatt a FORTRAN nehezen felel meg az olyan elvárásoknak, mint a könnyu továbbfejleszthetoség és bovíthetoség. Ezeket az elvárásokat a fejlett piac igényli a programoktól, így a programfejlesztoktol is.

A procedurális nyelvekkel ellentétben, mint például a FORTRAN, a modern programtervezés másik technikája az objektum elvu tervezés és programozás. Az *objektum elvu* gondolkodás az emberi gondolkodás szerves része. A valós világot objektumok alkotják: állatok, gépek, emberek. Minden objektumnak vannak saját attribútumai (szín, méret) és metódusai, olyan eljárások, amelyekkel az objektum a külvilággal tart kapcsolatot (például madárnál: énekel, repül, sétál). Az objektumok között több fajta kapcsolat létezik. Ezek közül kettot emelnék ki:

- öröklodés: egyik objektum a másik leszármazottja, specializálása, konkretizálása. Ekkor a leszármazott objektum örökli az ostípus minden attribútumát, metódusát. Például a kutya, mint faj, leszármazottja az emlosöknek.
- reláció: két objektum közötti bármilyen kapcsolatot relációnak nevezünk. Reláció az öröklodés, a kommunikációs kapcsolat, testvéri kapcsolat.

Osztálynak nevezzük az azonos tulajdonságú objektumok halmazát (absztrakció), ekkor az objektum az osztály egy példánya. Az *osztálydiagram* egy olyan

[*] okl. építomérnök, doktorandusz, BME Hidak és Szerkezetek Tanszéke

diagram, ami osztályokat és a közöttük lévo kapcsolatokat írja le rajzi formában. Egy objektumnak vannak *állapotai*, amivel egy adott pillanatban jellemezhetjük az objektumot. Az objektum egyszerre csak egy állapotban lehet, de egyben lennie kell. Az *állapotdiagram* az objektum életét reprezentálja, állapotámeneteket ír le, milyen feltételek teljesülése esetén juthat egyik állapotból a másikba. Az objektum elvu modellezés egy olyan eszköz, amellyel könnyen és áttekinthetoen tudjuk leírni és megfogalmazni a feladatot és a megoldással szemben támasztott elvárásainkat. Egyik ilyen eszköz az UML modellezo nyelv (Unified Modeling Language) [1], amellyel a tárgybeli program modelljét készítettem. Az objektum elvu modellezéssel párhuzamosan alakultak ki az *objektum orientált* programnyelvek: C++, Java, stb.

Az objektum elvuség szellemében készítettem el a BME Tartószerkezetek Mechanikája Tanszék támogatásával az alább bemutatásra kerülo interaktív rúdszerkezet számító program modelljét. Az *interaktivitás* a program és a felhasználója közötti kapcsolatra utal. Interaktív a program akkor, ha a felhasználói akciókra (adatbevitel / módosítás, egérgombnyomás) a program azonnal reagál. Az általam készített program célja, hogy a programba bevitt rúdszerkezetet ért változás (rúd inercia változás, csukló hozzáadás, támasz megszunése) azonnal megjelenjen az igénybevételi ábrákon. Természetesen nagy szerkezeteknél az eredményt (igénybevételi, lehajlás ábrák) hosszadalmas számítás után lehet csak megkapni. Ezt az idot a felhasználónak ki kell várnia, ami által a program veszít az interaktivitásából. Kis szerkezet vizsgálatánál viszont egy jól átgondolt modell garantálhatja nekünk a várt gyorsaságot.

Az így elkészülo programmal a Tanszék a mechanika alapelveit és összefüggéseit (pl. rúd inercia megváltozásának hatása a szerkezet viselkedésére) kívánja bemutatni a hallgatók számára eloadások alkalmával. Továbbá használható még a gyakorlatban kisebb szerkezetek vizsgálatára.

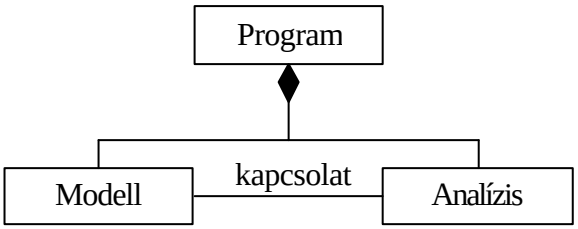
2. A MODELL

2.1. A program osztálydiagramja

Egy rúdszerkezet számító program az alábbi fobb részekbol épül fel:

- geometriai modell (Modell), mely a szerkezetet és a terheket írja le, az idevonatkozó objektumokat fogja össze,
- számító modul (Analízis), mely a rendszer matematikai oldalát képviseli, a lineáris egyenletrendszert megoldó algoritmus foglalja magában.

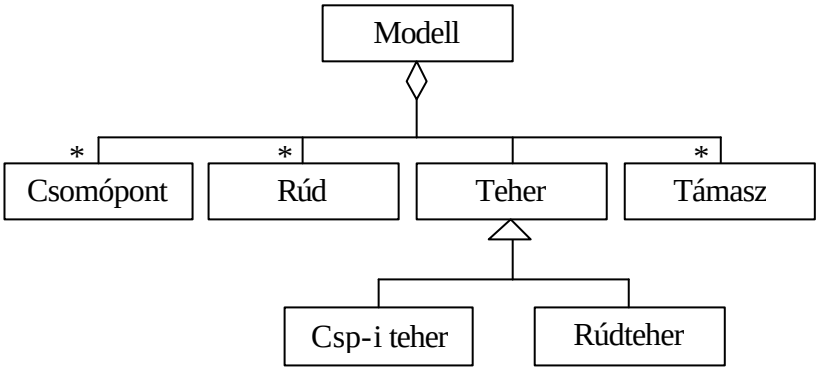
A felhasználó a modellen keresztül hozza létre a geometriát, módosíthat és kérhet le bármilyen adatot. Az analízis a modelltol szerzi be a számításhoz szükséges adatokat és jellemzoket. Ilyen adat például a probléma mérete (a lineáris egyenletrendszerben szereplo ismeretlenek száma), a rudak kapcsolati táblája (melyik rúd mely két csomópontot köti össze) és a támaszok megtámasztási módja. Ennek az összefüggésnek látható az osztálydiagramja az 1. ábrán.



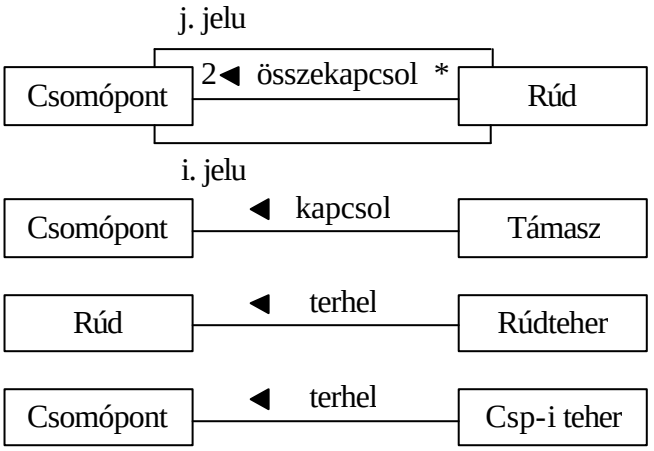
1. ábra: Program osztálydiagramja

2.2. A geometriai modell osztálydiagramja

A modell csomópontokból (Csomópont), az azokat összekapcsoló elemekbol, esetünkben rudakból (Rúd), teherbol és támaszokból áll. A terhek (Teher) felbonthatók két fajtára: csomóponti teher (Csp-i teher), melyek a csomópontokra hatnak, és a rudakat terhelo terhekre (Rúdteher). A szerkezet támaszokra támaszkodik fel, ami csak csomóponti támasz lehet (Támasz). Ezt fejezi ki a 2. és a 3. ábrában vázolt osztálydiagramok. Mivel egy bemutató program modelljérol van szó, ezért kikötést tehetünk arról, hogy csak egy terhet engedélyezünk a modellben, aminek támadási pontja és intenzitása változtatható, bemutatandó a hatásábra jelentését.



2. ábra: Modell osztálydiagramja



3. ábra: Osztálydiagramok

2.3. Analízis

Az eddigi objektum elvu véges elemes modellek [2,3] sajátosságai, hogy a számítási modul használati szempontból teljesen el van választva a geometriai modelltol; azaz szigorúan csak a következo sorrendben lehet használni az alábbi részeket: geometria bevitele / módosítása – számítás – eredmény megjelenítése. Jun Lu [4] egy kis méretu, rugalmas, „pihe könnyu” véges elemes programot mutat be, melynek magja a különbözo koordinátarendszerek közötti áttérést lehetové tevo transzformációs objektum. Ez utóbbi egyben bijektív megfeleltetést rendel a geometriai térbeli csomópontok szabadságfoka és a matematikai térben a lineáris egyenletrendszer ismeretlenjei között, ezáltal teljesen eltakarva a számító programrész elol rúdszerkezet mechanikai jelentését, így az analízisben csupán az egyenletrendszer megoldására lehet koncentrálni. Itt érezni is lehet az objektum orientált szemlélet erejét.

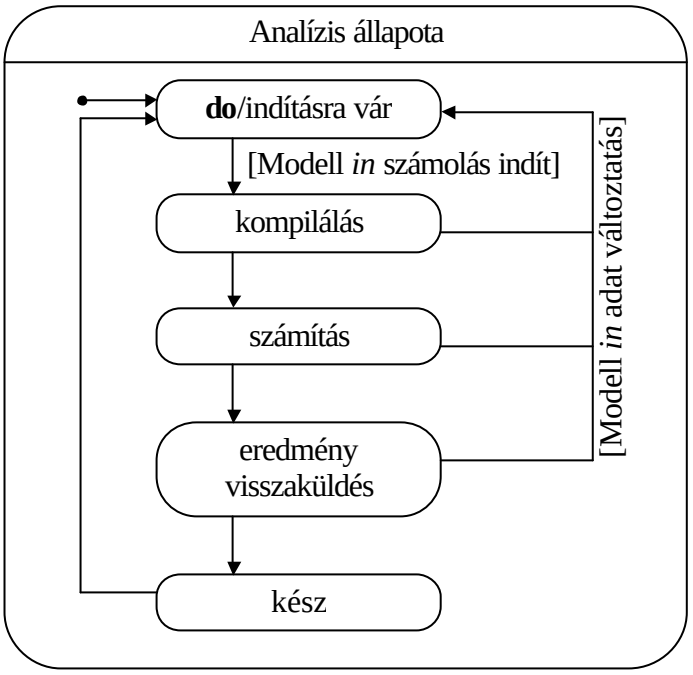
Fentiektol eltéroen egy interaktív program fo jellemzoje, hogy az eredmény és a geometria egyszerre jelenítheto meg. Az eredménynek azonnal meg kell jelenítenie a geometriát ért változtatásokat, amiket a felhasználó hajtott végre. Ezt az interaktivitást két módon lehet elérni.

Egyrészt a bevitt szerkezet kicsi méretébol fakadóan az elmozdulások kiszámítása gyorsan elvégezheto, így egy gyorsabb számítógépen a „hagyományos” adat bevitel – számítás – eredmény megjelenítés sorrenddel u.n. kvázi interaktivitás érheto el. Ennek magyarázata, hogy amíg a számítás modul fut, akár másodpercekrol is legyen szó, addig a program nem reagál a felhasználói eseményekre, majd végeztével ezeket az eseményeket rögtön feldolgozza, egy kicsit „szaggatott” muködési érzést keltve a felhasználóban. Ennek megvalósítása, azaz hogy milyen adatváltoztatások után vagy milyen idoközönként végezze el a számítást, nem jár nagy programozás technológiai fáradsággal.

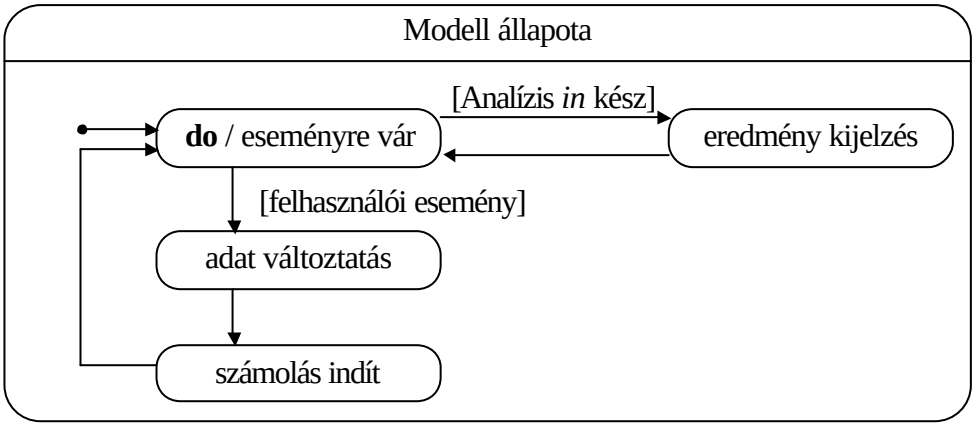
Másik módszerként megoldható a feladat többszálú programmal. Ez esetben a számítási modult külön folyamatszálként, a felhasználói kapcsolattartó szállal párhuzamosan is meg lehet tervezni. Ekkor a két folyamatot eros szinkronban kell futtatni egymás mellett. Ennek egyik lehetséges kivitele, hogy az adatokat ért bármilyen

módosuláskor a számítási szálat le kell állítani, az új adatokkal pedig újra indítani és visszatérni a felhasználói kapcsolattartásra, miközben háttérben fut a számítás. A külön folyamatszálon futtatott számítási modul, indulása után, végzi a feladatát és a szinkron által biztosított koherens adatokkal kell számoljon. A számítás végén jelzést küld a megjeleníto modulnak, ami része a felhasználói kapcsolattartó szálnak, és leáll. Fontos megjegyezni, hogy a számítás közben az adatokban ért változtatásokról a számoló rutin értesítést kapjon.

Ez a többszálú megoldás tényleges interaktivitást mutat a felhasználónak, ugyanis a program a felhasználói eseményekre azonnal reagál, ennek figyelését végzi aa felhasználói interfész folyamatszál. A számítási eredmények megjelenítése mindkét megoldási típusban azonos sebességgel történik. A párhuzamosra tervezett számítási modul és a modell állapotdiagramja látható a 4. és az 5. ábrán.



4. ábra: Analízis állapotdiagramja



5. ábra: Modell állapotdiagramja

3. ÖSSZEFOGLALÁS

A cikkben bemutatásra került egy objektum elven tervezett többszálú interaktív rúdszerkezet számító program modellje, melynek a számítást végzo analízis része párhuzamosan fut a felhasználói kapcsolattartó szállal biztosítandó az interaktivitást. A program Visual C++ nyelven lett implementálva. A modell nem épül a szerkezet bárminemu linearitására (geometriai, anyagi), így továbbfejleszthetо nemlineáris viselkedésu szerkezetek vizsgálatára is.

HIVATKOZÁSOK

[1] Sike S. – Varga L.: Objektum elvu modellalkotás UML-ben, *ELTE TTK Informatikai Tanszékcsoport*, Budapest, 2001.

[2] Archer G.C.: Object-oriented finite element analysis, *dissertation, University of California at Berkeley*, 1996.

[3] Duboispelerin Y., Zimmerman T.: Object-Oriented Finite Element Programming. 3. An efficient implementation in C++, *Computer Methods In Applied Mechanics And Engineering*, 1993 september.

[4] Lu J., et al: A matrix class library in C++ for structural engineering computing, *Computers & Structures, Vol. 55. No. 1, 95-111,* 1995.