MŰSZAKI ÉS TERMÉSZETTUDOMÁNYI EGYESÜLETEK SZÖVETSÉGE
NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG

I. ORSZÁGOS KONGRESSZUS ELŐADÁSOK

II.

Szeged
1979. december 3–7.

I. ORSZÁGOS KONGRESSZUS

ELŐADÁSOK

II.

SZEGED
1979. december 3–7.

A konferencia szervezőbizottsága:

Elnök: Muszka Dániel

Fülöp József
Gyémánt László
Gyimóthy Tibor
Lenkehegyi Ibolya
Madarász István
Nagy Julianna
Szabó Lenke
Tombácz József

A konferencia programbizottsága:

Elnök: Dömölki Bálint

Ada-Winter Péter
Borbáth György
Csendes Mihály
Filep György
Ivanyos Lajos
Legendi Tamás
Makay Árpád
Náray Miklós
Simák Pálné
Széphalmi Géza
Tamás Endre
Tóth Tamásné
Westsik György

Az előadások lektorai:

Balogh Kálmán
Báthor Miklós
Borbáth György
Csibi Sándor
Dávid Gábor
Dettrich Árpád
Fedina László
Fidrich Ilona
Filep György
Gergely Csaba
Hámori Miklós
Ivanyos Lajos
Jantsár Sándor
Jakabffy Imre
Kecskés József
Kerekes István
Kiefer János
Kovács Péter
Legendi Tamás
Lukács József
Makay Árpád
Mérő László
Molnár László
Náray Miklós
Németi Tibor
Nyiri Géza
Pongrácz Tibor
Pótzy Péter
Révész Ferenc
Simkó János
Stahl János
Szentiványi Tibor
Tóth Imre
Varga Lajos
Vasvári György
Volf László
Wagner Gyula

Felelős kiadó: Vámos Tibor
Kulturális Minisztérium kiadói engedély száma: 49531
Készült az OVK nyomdaüzemében 700 példányban
Felelős vezető: Fehérvári Béla
Munkaszám: 791017

TARTALOMJEGYZÉK

Kuba Attila–Csernay László (SZOTE):
Számítógépes tomográfia szimulációja 5
Dr.Kunszt György (Építéstudományi Intézet):
A tudományos kutatás irányításának számítógépes segítése. Egy logikai modellekre támaszkodó hazai törekvés tapasztalatai 16
Legendi Tamás (MTA Automataelméleti Kutató Csoport):
Sejtprocesszorok tervezése és programozása 23
Dr.Lengyel György (Pest megyei Állami Építőipari Vállalat):
Építőipari termelésprogramozás mini számítógép segítségével 29
Molnár László–Széphalmi Géza (ÉSZSZ)–Pauka Tibor (Orvostovábbképző Intézet):
Az egészségi állapot országos felmérésének számítástechnikai megoldása 35
Molnár István (KG ISZSZI):
Gépipari vállalat informatika rendszer tervezésének modellszintű megközelítése . . . 43
Müller Márta–Szmrecsényi Klára (SZÁMKI):
Az INES–2 általános célú információs rendszer 52
Németh Pál–Köves Péter–Mannhardt Endre (SZKI):
Az R–15 számítógép alapvető műszaki-rendszertechnikai jellemzői 56
Németi Tibor (SZKI):
Az SZKI TIME SHARING számítógépes rendszerének kialakítása, optimalizálása . . . 66
Dr.Obádovics J. Gyula (MüM SZÁMTI):
Programok hatékonyságának javításáról 75
Páldi Vince–Fóti György (KSH SZIG):
Rendszer-információk felhasználása egy hatékony kétgépes üzem kialakítására . . . 83
Rákóczi Ferenc (Videoton Fejlesztési Intézet):
A számítógépes nyomtatás technológiája és annak hazai vonatkozásai 90
Reményi Péter (FTV) –dr. Abaffy József–Lukács Márta (SZTAKI):
Az Építési Geotechnikai Adattár, mint a számítógéppel segített mérnöki tervezés gépi információs rendszere 97
Révész György (MTA SZTAKI).
A típusmentes λ-kalkulus alkalmazása programnyelvek szemantikájának leírására . . 102
Réti Tamás–Küllős József (Gépipari Technológiai Intézet).
Síkbeli mikroszkópos részecskék alakjának morfológiai jellemzése képelemzési módszerrel, különös tekintettel a kvantitatív metallográfiai alkalmazásra 111
Simonfai László (SZÁMKI):
Az adatbáziskezelő lehetőségei és problémái Magyarországon 120
Simor Gábor (SZKI):
Nyelvorientált számítógépek utasítás-készletének tervezése 128
Somogyi József (SZÁMKI):
Operációs rendszerek hordozhatóságáról 138
Szigetvári Miklós (KSH SZIG):
Automatizált adathordozó előkészítő rendszer és két nagygép kapcsolatának szimulálása software eszközökkel 143
Dr.Tóth József (Agrártudományi Egyetem, Debrecen):
A mezőgazdasági vállalatok automatizált tervezési rendszere és alkalmazásának tapasztalatai 151

Dr. Treer Róbert—Wlassics Péter (ÁSZSZ):
Absztrakt gépre alapozott rendszerprogramozási környezet létrehozása az ÁSZSZ számítógépein 162
Tóth Béla (ÉLGAV):
A felhasználók szükségleteivel összehangolt számítóközpont fejlesztése 170
Dr.Varga Gyula—Dr.Abaffy József (MTA SZTAKI) —Reményi Péter (FTV):
Interpolációs eljárás, megbízhatósági vizsgálatok és rajzoló program geotechnikai függvényekre 183
Vasas Sándorné dr.—Sarkadi Miklós (ÉGSZI):
Termeléstervezési és erőforrásszükséglet számító programrendszer a DBOMP és az LPS programcsomagok felhasználásával 187
Vámos Tibor—Báthor Miklós—Mérő László (MTA SZTAKI):
Ismeretanyag alapján dolgozó interaktív robotszem-rendszer 194
Vinkovits László—Trencsényi István (VIDEOTON Fejlesztési Intézet):
Az R—11 számítógép hardware és software jellemzői 201
Wildner Dénes (ÉGSZI):
Műszaki felületillesztési feladatok megjelenítési problémái és a megvalósításhoz szükséges két algoritmus bemutatása 205
Zámori Zoltán (KFKI):
Personal Computerek 212
Dr.Zeley István (ÉGSZI):
A rendszerelvű építést segítő számítógépes rendszer 220
Zsombok Zoltán (KSH ÁSZSZ):
Egy új programozási szemléletmód, a dinamikusan struktúrált vezérlés 232

SZÁMÍTÓGÉPES TOMOGRÁFIA SZIMULÁCIÓJA

Kuba Attila – Csernay László
SZOTE, KÖZPONTI IZOTÓPDIAGNOSZTIKAI LABORATÓRIUM, SZEGED

1. Bevezetés

Az utóbbi évtizedben forradalmi változást idézett elő az orvosi diagnosztikában a számítógépes tomográfok megjelenése. Ezek a készülékek lehetővé teszik, hogy a vizsgált beteg testének keresztmetszeti képeit állíthassuk elő Röntgen-sugárzás vagy rádióaktív izotóp alkalmazásával, a legkisebb sebészi beavatkozás nélkül. A számítógépes tomográfok működésének elméleti alapjait jelentő matematikai módszereket (az ún. rekonstrukciós eljárásokat) megszületésük pillanatától fogva nagy figyelem kíséri, mivel előnyös vagy előnytelen tulajdonságaik a gyakorlati alkalmazásokban döntő mértékben befolyásolhatják az eredményt.

Előadásunkban ennek a képalkotási módnak számítógépes szimuláció segítségével való modelljét mutatjuk be és beszámolunk a szimulátor bizonyos paramétereinek változtatásával kapott tapasztalatainkról is.

2. A számítógépes tomográfia mint matematikai probléma

A tomográfiás eljárások célja, hogy valamely térbeli tárgy síkmetszeteinek képeit állítsák elő. Ehhez általában a tárgy különböző irányokból megkapott vetületeit használják fel (ilyen vetületek például a Röntgen-felvételek, ill. a nukleáris medicinában az ún. szcintigráfiás felvételek).

Matematikailag ez a feladat a következőképpen fogalmazható meg:

Jelölje f (x,y) a vizsgált metszetet, az f (x,y) függvény γ^{ρ}-szögű vetületét pedig $f_{\gamma^{\rho}}(u)$.

$$f_{\gamma^{\rho}}(u) = \int_{-\infty}^{\infty} f(x,y)\, dv \qquad (1)$$

ahol u,v az x,y koordináta-rendszer γ^{ρ}-szögű elforgatásával kaphatók (1. ábra).

Rekonstrukciós probléma: Adott $f_{\gamma^{\rho}1}(u)$, $f_{\gamma^{\rho}2}(u)$, ..., $f_{\gamma^{\rho}N}(u)$; keresendő az az f (x,y), aminek éppen a megadott függvények a megfelelő szögű vetületei. (A 3-dimenziós rekonstrukciót is hasonló módon lehet megfogalmazni, azonban a gyakorlat számára elegendőnek látszik a térbeli tárgy kellően sok metszetét rekonstruálni.)

3. Rekonstrukció konvolúciós algoritmussal

A rekonstrukciós probléma megoldására számos eljárás született (általános áttekintést ad pl. [1]), amelyek lényegében három nagy csoportra bonthatók: algebrai iteratív rekonstrukciós technikák [2], Fourier rekonstrukciós módszerek [3] és a konvolúciós algoritmusok [4]. Ezek közül a konvolúciós algoritmus az, amely előnyös tulajdonságai folytán a legelterjedtebb a Röntgen-sugárzást használó ún. transzmissziós számítógépes tomográfiában. Mivel szimulációs programrendszerünkben is ezt a rekonstrukciós eljárást alkalmazzuk, ezért röviden vázoljuk megoldási elvét.

Legyen $f_P(r, \varphi)$ függvény a rekonstruálandó alakzat és $F_P(\varrho, \Phi)$ ennek 2-dimenziós Fourier-transzformáltja polárkoordinátás alakban; ekkor

$$F_P(\varrho, \Phi) = \int_{0}^{\pi} \int_{-\infty}^{\infty} f_P(r, \varphi)\, e^{-jr\varrho\cos(\varphi-\Phi)}\, (r)\, dr d\varphi$$

és

$$f_P(r, \varphi) = \frac{1}{4\pi^2} \int_{0}^{\pi} \int_{\infty}^{\infty} F_P(\varrho, \Phi)\, e^{jr\varrho\cos(\varphi-\Phi)}\, (\varrho)\, d\varrho\, d\Phi. \qquad (2)$$

A (2) egyenlet szerinti előállításhoz felhasználjuk a rekonstrukciós irodalomban jól ismert ún. vetület-szelet tételt [5], amely szerint

$$f_{\gamma^\vartheta}(u) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F_P(\varrho, \Phi = \gamma^\vartheta)\, e^{jr\varrho}\, d\varrho \qquad (3)$$

Ezek után legyen

$$\widetilde{F}_P(\varrho, \Phi) = F_P(\varrho, \Phi)\, W(\varrho, \Phi), \qquad (4)$$

ahol

$$W(\varrho, \Phi) = \begin{cases} 1, \text{ ha } (\varrho) < A \\ 0 \text{ különben} \end{cases}$$

és ennek az $\widetilde{F}_P(\varrho, \Phi)$-nak megfelelő $\widetilde{f}_P(r, \varphi)$ függvényt állítsuk elő $f_P(r, \varphi)$ helyett. Ekkor (4) inverz transzformáltja (2) szerint

$$\widetilde{f}_P(r, \varphi) = \frac{1}{4\pi^2} \int_{0}^{\pi} \int_{-\infty}^{\infty} F_P(\varrho, \Phi)\, W(\varrho, \Phi)\, (\varrho)\, e^{jr\varrho\cos(\varphi-\Phi)} d\varrho\, d\Phi \qquad (5)$$

A belső integrál a konvolúciós tétel értelmében (lásd pl. [6]) két függvény konvolúciójával helyettesíthető:

$$p(r, \Phi = \gamma^\vartheta) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f_{\gamma^\vartheta}(u)\; h(r-u)\, du \qquad (6)$$

ahol $f_{\gamma^\vartheta}(u)$ a (3) egyenletből adódik, $h(r)$ pedig:

$$h(r) = \int_{-\infty}^{\infty} (\varrho)\, W(\varrho, \Phi)\, e^{-jr\varrho} d\varrho = 2 \int_{0}^{A} \varrho\, \cos(r\varrho)\, d\varrho$$

végül $\widetilde{f}_P(r, \varphi)$ ($p(r, \varphi)$-ből, így kapható:

$$\widetilde{f}_P(r, \varphi) = \frac{1}{2\pi} \int_{0}^{\pi} p(r\cos(\varphi-\gamma^\vartheta), \gamma^\vartheta)\, d\gamma^\vartheta \qquad (7)$$

Vagyis az $\widetilde{f}_P(r, \varphi)$ függvény a vetületek (6) szerinti szűrése után a (7) által leírt ún. visszavetítéssel kapható meg.

4. Számítógépes szimuláció

A rekonstrukciós algoritmusok vizsgálatában általánosan elterjedt módszer a szimuláció alkalmazása, vagyis a szóban forgó folyamat egy olyan modelljének a megalkotása számítógépre írt programok segítségével, amely képes a különféle, bonyolult módon összefüggő hatások

egymástól jól elkülöníthetően visszatükrözni. Az így elkészült szimulátorok kitűnően alkalmazhatók rekonstrukciós eljárások kipróbálására, összehasonlítására, bihonyos zaj-hatásokkal szembeni ellenállási képességeik kimutatására és speciális fizikai, technikai adottságok közötti működésük vizsgálatára is.

Az általunk elkészített szimulátor jelenlegi formájában elsősorban a vetületi paraméterek változtatásának hatásait tükrözi vissza. A szimulátor három fő részből áll.

a) Vetületképzés

Ennek a résznek a feladata a rekonstrukció számára szükséges vetületi adatok előállítása. A rekonstruálandó f (x,y) függvényt ún. fantom képek fogják jelenteni, amelyeket bizonyos egyszerű geometriai síkidomokból (kör, ellipszis, sokszög) lehet felépíteni, tetszőleges bonyolultságú ábrákat képezve. Minden egyes síkidomhoz egy-egy anyagi minőségre (pl. sűrűségre) jellemző érték rendelhető. Az így összeállított alakzatokból készíthetők el a vetületek.

A vetületeket az (1)-ben megadott vonal menti integrálok helyett most csak közelíteni tudjuk. Mégpedig úgy, hogy meghatározott irányú párhuzamos egyenesek által közrefogott sávokon belüli integrálokat számítunk ki (2. ábra) és osztjuk el a sáv szélességével. Az egyes fantomokból kapott vetületek ezek alapján 3 paraméterrel jellemezhetők:

N: a vetületek száma (egyenletesen felosztva a $(0, \pi)$ szögtartományt),

M: az egyes vetületeket alkotó mérési pontok száma, vagyis hogy egy-egy irányban hány sáv húzódik egymás mellett (egyenletes beosztást feltételezve),

s: a vetítéshez kijelölt sávok szélessége.

Szimulátorunkban N és M értéke 1 és 360 között változhat.

b) Rekonstrukció

A korábban ismertetett konvolúciós algoritmust használjuk a vetületekből való rekonstrukcióhoz. A művelet két lépésben hajtódik végre: először a (6) egyenlet szerinti konvolúciót kell elvégezni, majd az így módosított vetületekre kell a (7) szerinti vissza-vetítést (back-projection) alkalmazni.

c) Megjelenítés és a képek feldolgozása

A rekonstrukció eredményeként kapott mátrixot off-line módon tudjuk képi formában megjeleníteni egy TPA-i kisszámítógép által vezérelt színes TV-n. A képernyő 128x120-as képet ad 8 színben. A képek minél informatívabb megjelenítése céljából a viszonylag kevés szín minél gazdaságosabb kihasználására több kijelzési módszert is kidolgoztunk [7].

Azonban a vizuális élményt jelentő képi megjelenítéseken kívül szükség van még olyan mérőszámokra is, amelyek a rekonstruált képeknek az eredetitől való eltérését képesek mérni. Többek között ilyen pl. a relatív általános négyzetes eltérés is.

$$\delta = \sqrt{\frac{\Sigma (f_{ij} - f'_{ij})^2}{\Sigma (f_{ij} - \bar{f})^2}}$$

ahol f_{ij} jelöli az eredeti kép elemeit, f'_{ij} a rekonstruált képét és $\bar{f}$ az eredeti kép átlagos értékét.

5. A szimulációs kísérletek eredményei

Szimulációs kísérleteink elsődleges célja az volt, hogy a rekonstrukció pontosságát be-

befolyásoló vetítési tényezők hatását mérjük le. Ehhez azt a módszert követtük, hogy a vizgált metszet képét reprezentáló fantomról a vetítési paraméterek valamelyikének folyamatos változtatásával kapott vetületekből rekonstrukciók sorozatát hajtottuk végre, miközben a többi paramétert fix értéken hagytuk. A rekonstruált képeknek az eredetivel való összehasonlításából következtettünk az egyes tényezők befolyásának erősségére.

A vizsgálatokban több fantom képet is használtunk. Ezek közül az elsőn, amelyet korábban Shepp és Logan közöltek [8], mutatjuk be a vetületek számának (N) és a vetületeket alkotó mérések számának (M) a változtatásából származó hatásokat.

a) A vetületek számának a növelése

A vetületek számát N=9 és N=300 között változtattuk, miközben a többi paraméter fi maradt. A rekonstruált képek 80x80-as mátrixaiból számítottuk ki δ értékét és kaptuk a következő grafikont (4. ábra). Ez alapján megállapítható az az előre is sejthető eredmény, hog a vetületek számának a növelése javítja az eljárás pontosságát. Egyúttal igazolódni látszik Wagner eredménye [9], amely szerint a vetületek száma és a két kép közötti eltérés között fordított arányosság van:

$$\delta \sim \frac{1}{N}$$

A rekonstruált képek (5., 6. ábra) megtekintése is alátámasztja, hogy lényeges javulás csak kb. N=180-ig van.

b) A vetületek alkotó mérések számának a növelése

Az egyes vetületek mérési pontjainak a számát M=24 és M=342 között változtattuk úgy hogy közben a vetítéshez használt nyalábok vastagsága mindig a szomszédos mérési pontok távolságával volt egyenlő. Más és más vetületszámra megismételve a kísérleteket, jutottunk a 7. ábrán látható grafikonhoz. Tehát a mérések számának a növelésével javul a rekonstrukció minősége (8., 6. ábra). Ebben a sorozatban M=240 után már elhanyagolható volt a javulás mértéke.

c) A vetítési sugár-nyaláb szűkítése

Végül külön foglalkoztunk azzal a kérdéssel, hogy milyen változás tapasztalható a rekonstruált képben, ha a vetítéshez használt nyaláb vastagságát változtatjuk. Különböző s értékre megismételve az eljárást, kaptuk a 9. ábrán látható grafikont, amely szerint egyértelm javulás figyelhető meg a nyaláb szűkítésével. Ez az eredmény látszólag ellentmondásban van azzal a ténnyel, hogy rögzített N és M mellett vékonyabb nyalábbal a rekonstruálandó metszet kisebb területéről gyűjtünk be egyszerre információt. Ezt a hatást azonban kiegyensúlyozhatja az, hogy a szűkebb nyaláb pontosabb vetületi adatokat szolgáltat a konvolúciós algoritmus számára.

Ezen utóbbi állítás igazolására tekintsük a következőt: A pontos vetületi értékek hely a vetületképzésnél elmondottak alapján mi az

$$\overline{f}_{\gamma}(u; s) / s$$

értékeket számoljuk ki, ahol

$$\overline{f}_{\gamma}(u; s) = \int_{-\infty}^{\infty} f_{\gamma}(\zeta)\, q_s(u-\zeta)\, d\zeta$$

és

$$q_s(u) = \begin{cases} 1, \text{ ha } (u) < \frac{s}{2} \\ 0, \text{ különben.} \end{cases}$$

Ennek a függvénynek a Fourier-transzformáltja a konvolúciós tétel értelmében az $f_\vartheta(u)$ és a $q_s(u)$ függvények Fourier-transzformáltjainak a szorzata lesz:

$$\bar{F}_\vartheta(\varrho : s) = F_P(\varrho, \Phi = \vartheta) \cdot Q_s(\varrho), \tag{8}$$

mivel $f_\vartheta(u)$ transzformáltja a vetület-szelet tétel szerint éppen $F_P(\varrho, \Phi = \vartheta) \cdot Q_s(\varrho)$ pedig:

$$Q_s(\varrho) = \int_{-\infty}^{\infty} q_s(u)\, e^{-j\varrho u} du = \frac{2 \sin(\frac{s\varrho}{2})}{\varrho} . \tag{9}$$

A vetületképzésben leírt eljárás tehát azt feltételezi, hogy

$$\bar{f}_\vartheta(u; s)/s \approx f_\vartheta(u),$$

vagyis, (8) szerint

$$\bar{F}_\vartheta(\varrho; s) / S \approx F_P(\varrho, \Phi = \vartheta)$$

aminek (9) alapján csak úgy van létjogosultsága, ha minden ϱ -ra

$$\frac{2 \sin(\frac{S}{2})}{\varrho} \sim S$$

ami kb. $(\frac{S}{2}) < \pi/20$ feltétel teljesülése esetén helytálló. Tegyük fel, hogy $\bar{F}_\vartheta(\varrho; s) = 0$, ha ha $(\varrho) > A$; akkor az

$$S < \frac{\pi}{10A} \tag{10}$$

relációnak kell s-re teljesülnie.

A nyaláb szűkítésének hatása figyelhető meg a 10. ábrán látható fantom segítségével [10]. Erre a képre N=180 vetület és M=120 mérés esetén a 11. ábrán látható képet kapjuk, (ekkor s=20) és $\pi/10A = 2$ volt, míg a 12. ábra esetében (s=0,2), vagyis ezen nyaláb-vastagság mellett (1) teljesül. A javulás elsősorban a belső körök kontrasztosabb rajzolatában szembetűnő a 11. ábrához képest.

Irodalom:

[1] R.A. Brooks, G.DiChiro: Principles of computer assisted tomography (CAT) in radiographis and radioisotopic imaging; Phys. Med. Biol., 21, 689–732 (1976).

[2] G.T. Herman, A. Lent: Iterative reconstruction algorithms; Comput. Biol. Med., 6, 273–294 (1976).

[3] R.N.Bracewell: Strip integration in radio astronomy; Aust. J. Phys., 9, 198–217 (1956).

[4] G.N.Ramachandran, A.V.Lakshminarayanan: Three-dimensional reconstruction from radiographs and electron micrographs; Proc. Natl. Acad. Sci. USA, 68, 2236–2240 (1971

[5] R.M.Mersereua: Direct Fourier transform techniques in 3–D image reconstruction; Comp. Biol. Med., 6, 247–258 (1976).

[6] A.Papoulis: The Fourier Integral and Its Applications Mc Graw-Hill, New York, 1962.

[7] Kuba A., Csernay L., Kovács A.; Computeres tomográfiai szimulációjával szerzett tapasztalatok; Számítástechnikai és kibernetikai módszerek alkalmazása az orvostudományban és a biológiában, Kollokvium, Szeged, 1977.

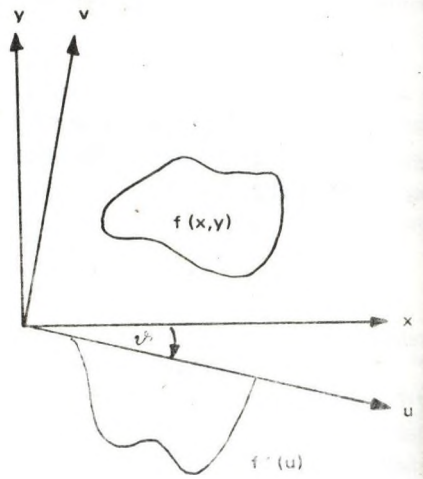
[8] L.A.Shepp, B.F.Logan: The Fourier reconstruction of a head section; IEEE Trans. on Nucl. Med., NS–21, 21–43 (1974).

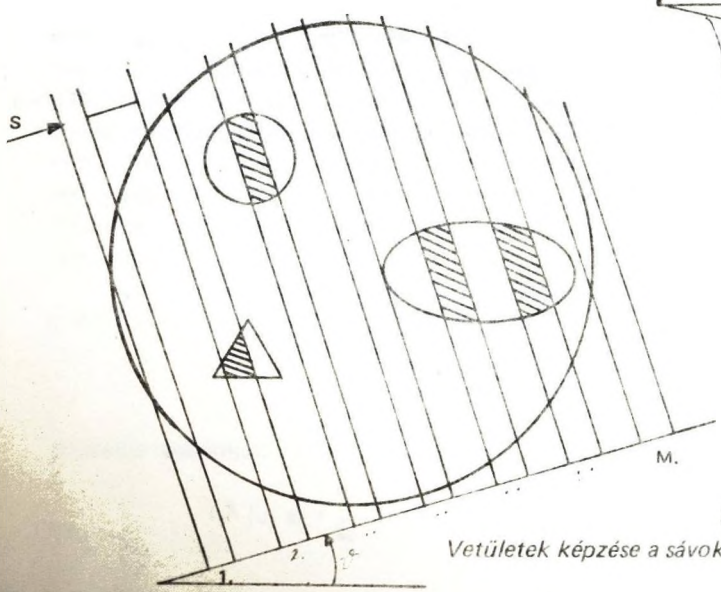
[9] W. Wagner: Reconstruction of objects layers from their X-rays projections: a simulation study; Comp. Graph. and Image Proc., 5, 470–483 (1976).

[10] B.K.P. Horn: Density reconstruction using arbitrary ray-sampling schemes; Proc. of IEEE 66, 551–562 (1978).

1. ábra

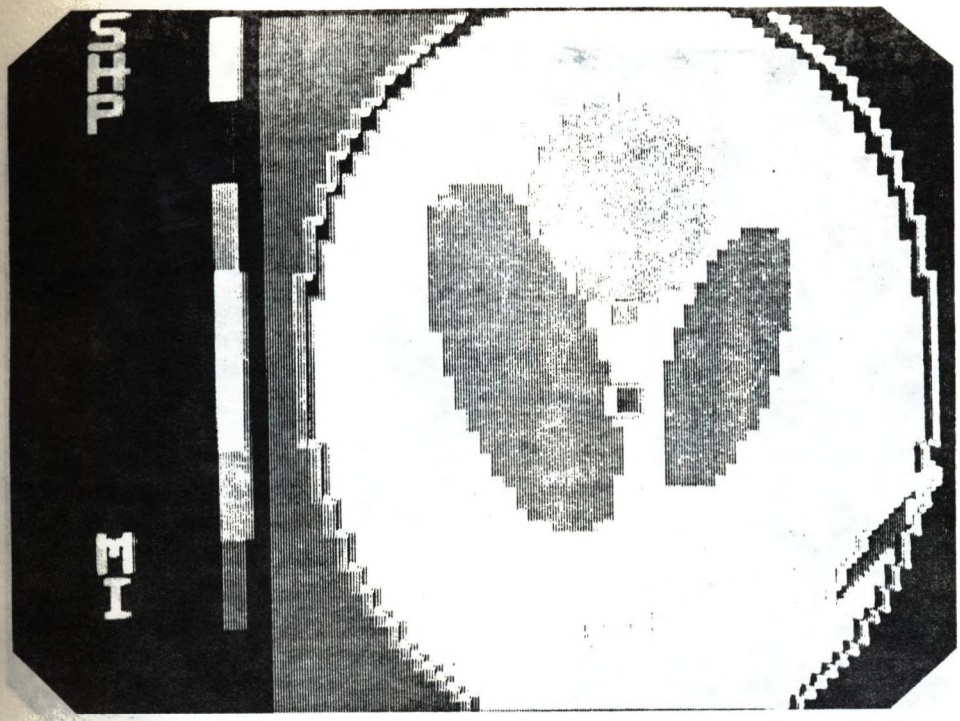
f(x,y) függvény -szögű vetülete



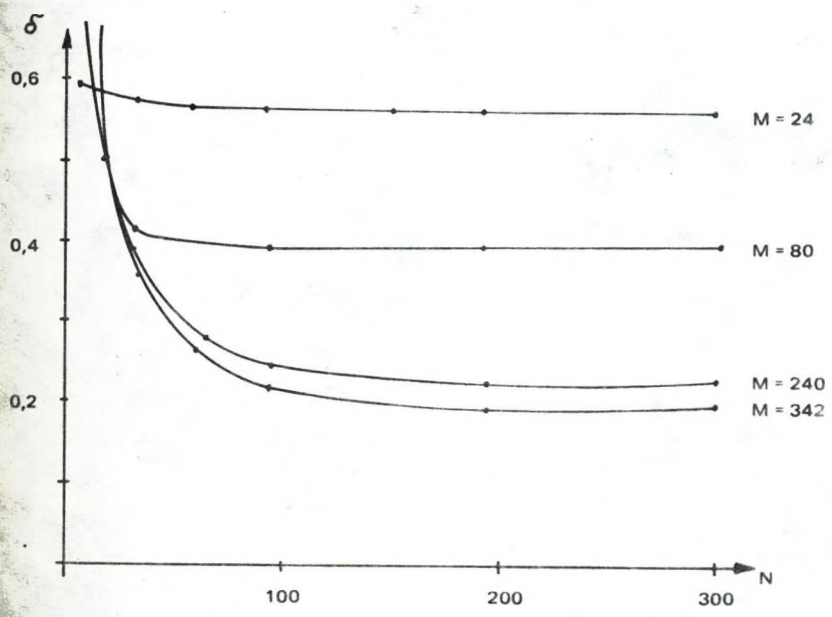


2. ábra

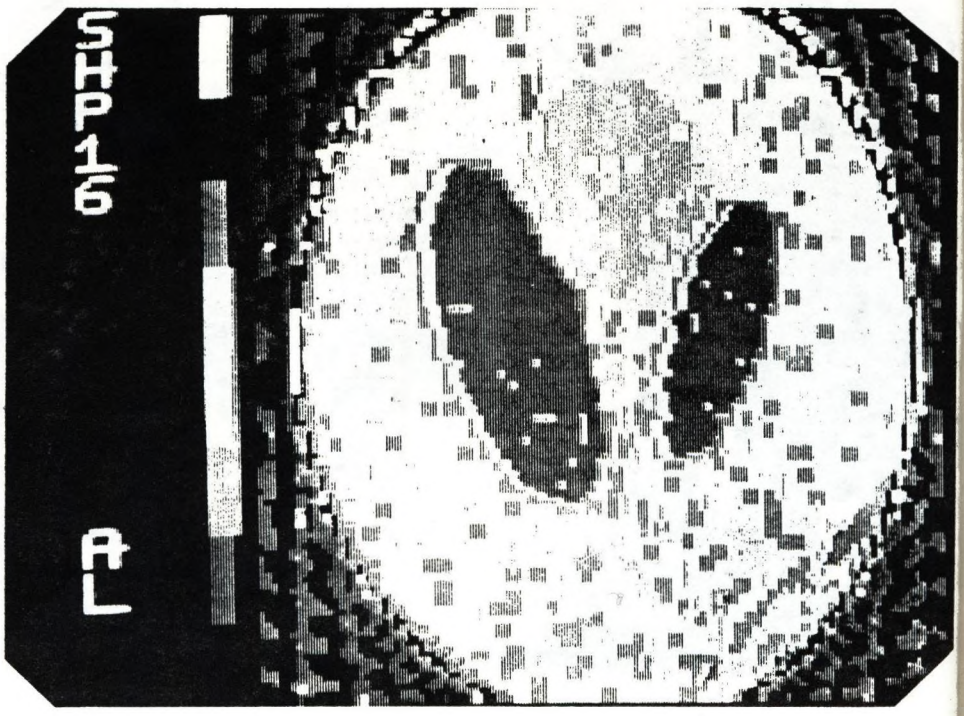
Vetületek képzése a sávok által meghatározott területeken



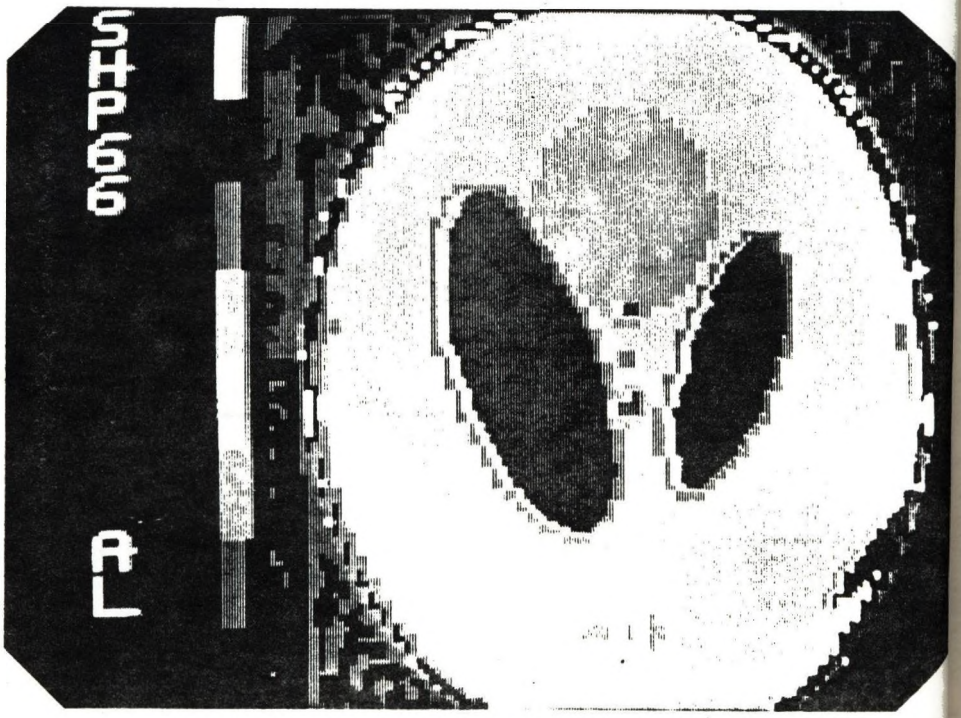
3. ábra A vizsgálatokhoz használt egyik fantom képe



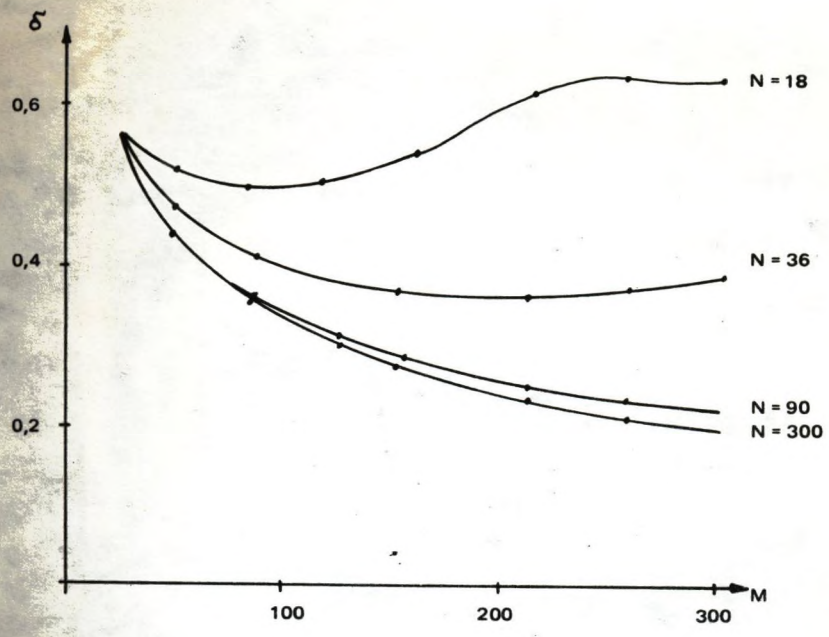
4. ábra A rekonstrukció pontossága a vetületszám függvényében



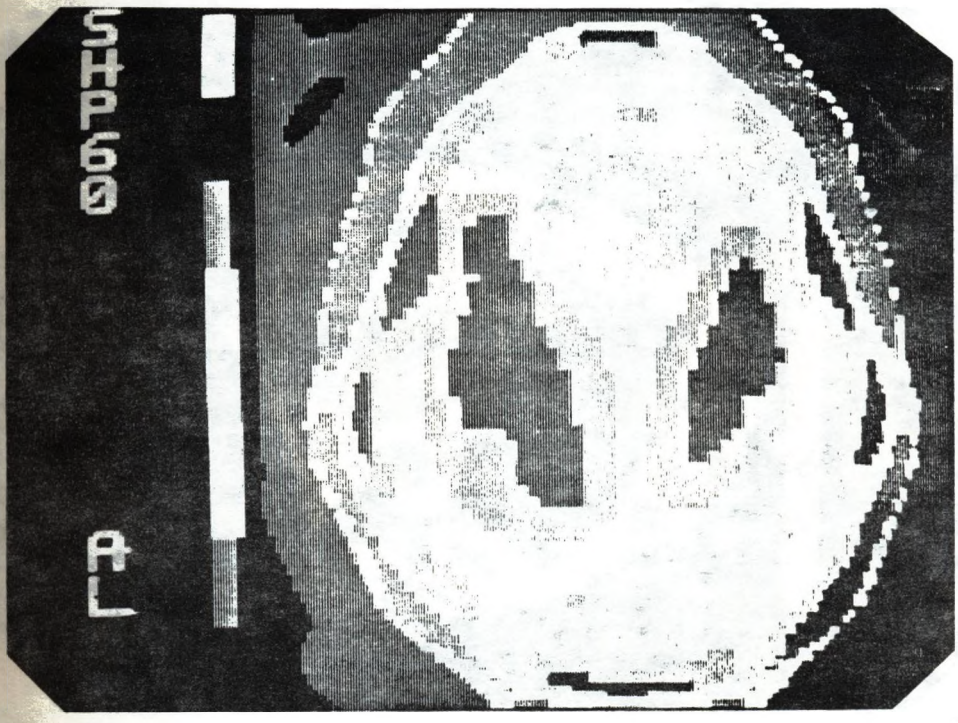
5. ábra A rekonstruált kép, N=18, M=240



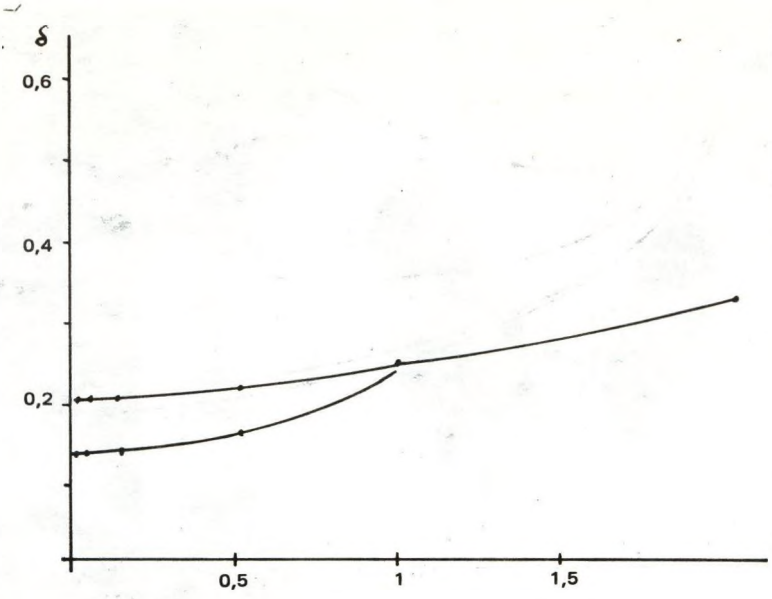
6. ábra Rekonstruált kép, N=180, M=240



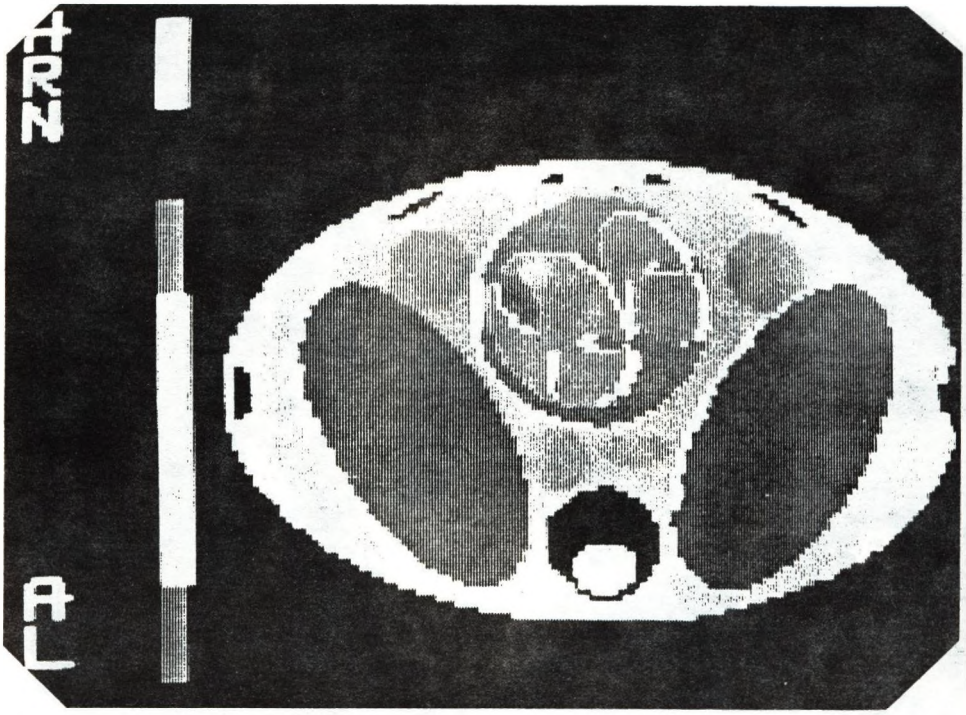
7. ábra. A rekonstrukció pontossága a mérések számának függvényében



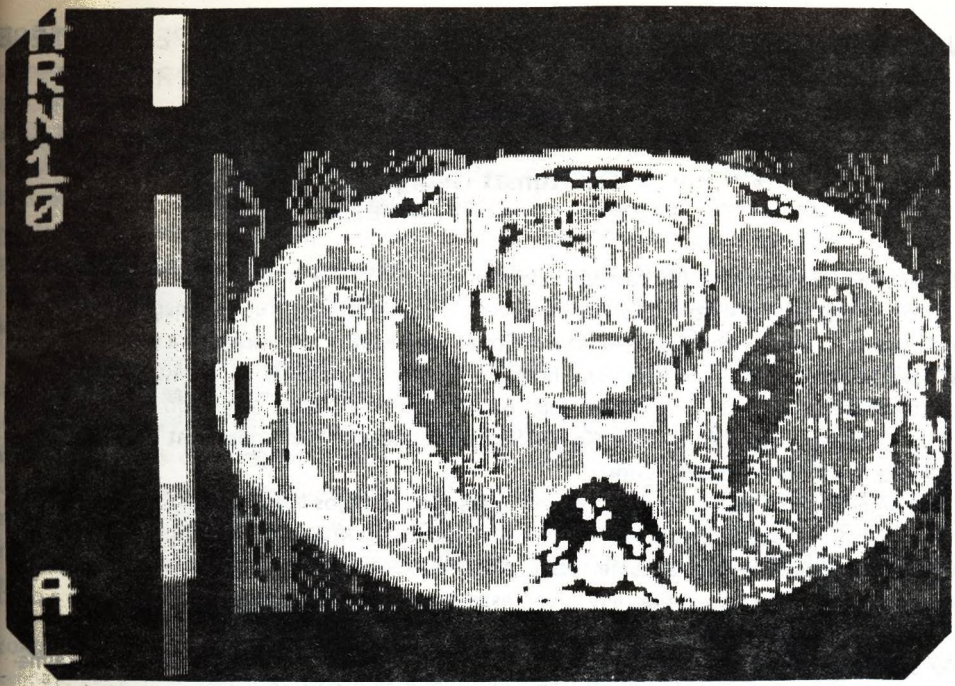
8. ábra Rekonstruált kép, N=180, M=24



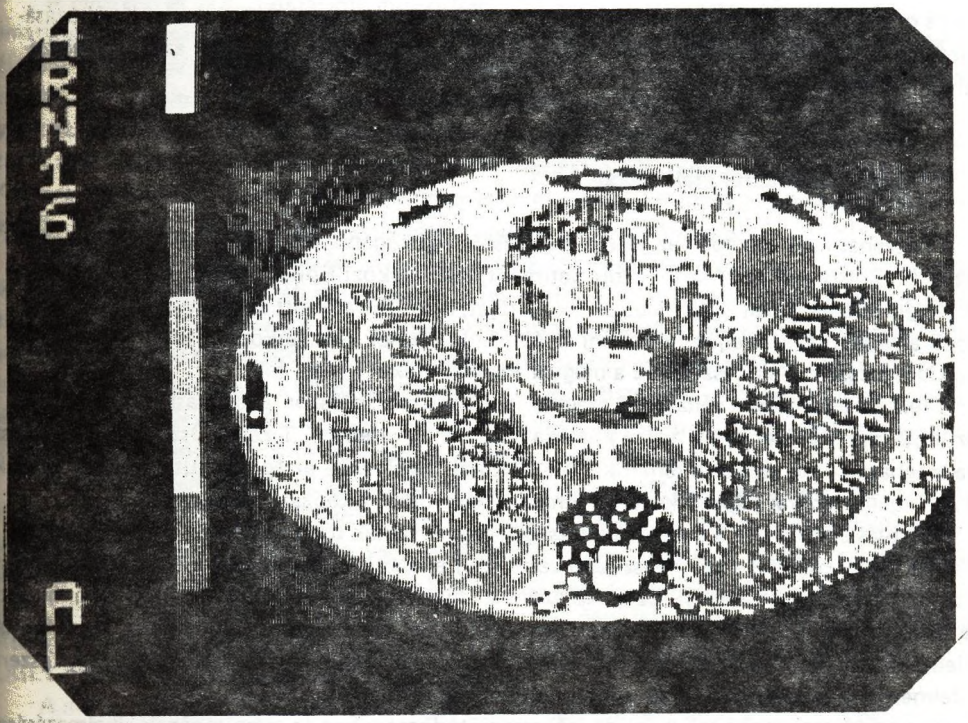
9. ábra A rekonstrukció pontossága a sávszélesség függvényében



10. ábra A kísérletekben felhasznált másik fantom



11. ábra A 10. ábrán látható fantom rekonstruált képe, N=180, M=120, s=20



12. ábra A 10. ábrán látható fantom rekonstruált képe, N=180, M=120, s=0,2

A TUDOMÁNYOS KUTATÁS IRÁNYÍTÁSÁNAK SZÁMÍTÓGÉPES SEGÍTÉSE EGY LOGIKAI MODELLEKRE TÁMASZKODÓ HAZAI TÖREKVÉS TAPASZTALATAI

Dr. Kunszt György
ÉPÍTÉSTUDOMÁNYI INTÉZET

A tudományos kutatás irányításának számítógépes segítésére hazánkban eddig csekély szá mú erőfeszítést tettek. Ezek áttekintésére mindmáig nem került sor és ennek megfelelően egyelőre nem alakult ki átfogó koncepció a hazai fejlődés szervezett meggyorsítására sem.

A jelen előadás nem vállalkozik arra, hogy a tudományos kutatás irányításának számítógépes segítését célzó hazai erőfeszítéseket áttekintse és értékelje. Ehhez az előadó nem rendelkezik kellő tájékozottsággal és ez az eddigi erőfeszítések szétszórt és rendszerint lokális jellege miatt egyébként is nehéz feladat lenne.

A Neumann János Számítógéptudományi Társaság Országos Kongresszusa kiváló alkalom viszont arra, hogy — hozzászólások révén — összkép alakuljon ki az eddigi hazai próbálkozásokról. Az előadó abban a reményben ismerteti a látókörébe eső területen végzett munkát és annak tapasztalatait, hogy ezzel bevezethet egy olyan eszmecserét és vitát, amely megteremtheti a probléma országos feldolgozásának és megoldásának elvi alapjait.;

Az előadó az úgynevezett LOGEL (LOGikai modELI)-módszer kialakulásáról és alkalmazásáról kívánja tájékoztatni a Kongresszust. Ez a módszer az építési ágazatban — elsősorban az Építéstudományi Intézetben — jött létre és eddig elsősorban az építés területén alkalmazták. Előzetesen is meg kell jegyezni azonban, hogy a módszer kialakítása szoros kapcsolatban állt bizonyos országos intézkedésekkel (elsősorban az Országos Kutatási Nyilvántartás létrehozásával), széles körű alkalmazásának lehetőségei adottak, hasznosításuk pedig már az építési ágazaton kívül is megkezdődött.

1. A LOGEL-módszer kialakulása

A hatvanas évek közepén az építési kutatások legátfogóbb nemzetközi tudományos szervezetében (CIB, Conseil International du Batiment, Nemzetközi Építéskutatási Tanács) megszervették a tagintézetek kutatási témalapjainak cseréjét, majd felvetődött az a gondolat, hogy e témalapok alapján megkíséreljék felvázolni az építési kutatások nemzetközi helyzetképét. A helyzetképkidolgozásának előkészítésére a magyar tagintézeteket kérték fel.

A feladat elvégzésére vállalkozó magyar munkacsoport először a kutatási statisztikák megszokott módszerével nyúlt hozzá a feladathoz. Az Egyetemes Tizedes Osztályozásra támaszkodva meghatározták azokat a tematikai osztályokat, amelyekbe az építési kutatások sorolhatók majd a vizsgálat tárgyát képező kutatási témákat besorolták a képzett osztályokba.

E sorolás eredménye a szokott nehézségek (egyes témák több osztályt is érintő tartalma, más témák egyik osztályba sem illeszkedő volta, stb.) ellenére is adott képet a nemzetközi építési kutatás tematikai struktúrájáról. Ez a kép azonban meglehetősen elmosódott volt, s nem mutatott sem a legfontosabb konkrét problémákat, sem pedig ezek kapcsolatait. Ugy látszott, hogy formális és objektív karakterű módszerekkel nem lehet használható eredményt nyerni, amikor egy újszerű ötlet felmerülése megváltoztatta az adott módszertani szituációt.

Az a gondolat merült fel, hogy az információtudományban (a számítógépek alkalmazásán is kiterjedő korszerű könyvtári gyakorlat elméletében) tárgyalt koordinált indexelés alapelveire támaszkodva tárgyszavak halmazaira képezzük le a vizsgált témákat, s a közvetlen statisztikai

elemzés tárgyává a témák rendszere helyett ennek a rendszernek azt a modelljét tegyük, amelyet a szóbanforgó leképzés szolgáltat. E leképzés matematikai reprezentációja célszerűen olyan páros gráf, amelyben a csúcsok egyik halmazának a vizsgált kutatási témák felelnek meg, a csúcsok másik halmazának pedig azok a tárgyszavak, amelyeket a témák tartalmi jellemzése során felhasználtunk; a gráf élei összekötik a témákat a jellemzésükre használt tárgyszavakkal. Az élek által reprezentált logikai reláció affiliációs relációnak nevezhető. A témarendszert modellező ezen gráfban a témát reprezentáló csúcsba futó élek száma (a téma-csúcs fokszáma) azt mutatja, hogy az adott témát hány tárgyszóval jellemeztük, a tárgyszavat reprezentáló csúcsba futó élek száma (a tárgyszó-csúcs fokszáma) pedig azt, hogy az adott tárgyszót hány téma jellemzésére használtuk.

A témákat és tárgyszavakat egyaránt tartalmazó — imént jellemzett — affiliációs modell alapján más affiliációs gráfok is konstruálhatók, külön a témákra és külön a tárgyszavakra. A csupán témákra vonatkozó affiliációs gráf csúcsa témát reprezentál, éle pedig — amelyhez számértéket is rendelünk — azt mutatja, hogy a hozzá illeszkedő témapárnak hány közös tárgyszava van. A csupán tárgyszavakra vonatkozó affiliációs gráf csúcsa tárgyszót reprezentál, éle pedig — amely szintén multiplicitásos jellegű — azt mutatja, hogy a hozzá illeszkedő két tárgyszót hány téma jellemzése során használtuk fel együtt. Az élekhez tartozó multiplicitás értéke természetesen 0 is lehet.

A CIB vizsgálat során felmerült tudománystatisztikai ötlet mármost abból állt, hogy a szokott statisztikai vizsgálatoktól eltérően figyelmünket nem tematikai osztályok népességének meghatározására fordítottuk, hanem az imént jellemzett három affiliációs modell számszerű strukturális elemzésére, amely elsősorban a csúcsok fokszámának és az élek multuplicitásának meghatározását, majd a nyert értékek értelmezését jelentette.

A tárgyszavak és kombinációik gyakorisági vizsgálata az elemzett rendszer szokatlanul éles tematikai jellemzését tette lehetővé, s így a felmerült ötlet segítségével mód nyílott a kitűzött cél elérésére. Már az 1966 és 67-ben végzett vizsgálat során is szükségesnek mutatkozott azonban, hogy bizonyos szemantikai torzítások megakadályozásának érdekében további kölcsönöket vegyünk az információtudomány logikai eszközeinek tárából. Nevezetesen, vizsgálatainkat a tárgyszavak affiliációs kapcsolatainak elemzésén kívül kiterjesztettük generikus relációk elemzésére is. (Két tárgyszó között — az információtudományban elterjedt definíciók értelmében — akkor áll fenn genetikus reláció, ha az egyik szemantikailag magában foglalja a másikat.)

A generikus relációk elemzése generikus gráfok szerkesztését, s egyben a generikus modellek újszerű tudományszervezési alkalmazását is jelentette. Ebben a tekintetben a legmesszebbre azzal mentünk el, hogy a vizsgálatok szemantikai korrektségének érdekében lényegesnek — de tudományszervezési szempontból is relevánsnak — minősítettük a vizsgálatok során használt diszkriptorok összességének a vizsgálatot követő —, vagy célszerűen — megelőző tezaurológiai feldolgozását, vagyis a generikus reláció segítségével hierarchikusan rendezett és szabályozott (szinonímáktól és homonímáktól mentes) szóállomány alkalmazását.

Az említett nemzetközi vizsgálat alkalmával mintegy másfélezer téma rendszerét elemeztük. A munka elvégzése során kiderült, hogy ilyen méretű rendszer affiliációs modellezése (pl. a tárgyszavak kombinált gyakoriságát, vagy a témák közös tárgyszavainak számát megadó gráfok megkonstruálása) gyakorlati körülmények között már nem képzelhető el pusztán manuális eszközök felhasználásával, hanem feltétlenül szükség van gépi feldolgozásra. Az ehhez szükséges software kidolgozását hamarosan megkezdtük és ez adta meg a későbbiekben kifejlődő LOGEL-módszer számítástechnikai kiindulópontját.

A módszer a teljes kifejlődés során nagy mértékben túlment azon a körön, amelyet az eddigiekkel jeleztünk: jelentős mértékben bővült azoknak a logikai relációknak a köre, amelyeket bekapcsoltunk az elemzésekbe és megnőtt azoknak a tudományszerzési feladatoknak a száma is, amelyek megoldására a jelzett módon képződő logikai modelleket elkezdtük felhasználni.

2. A LOGEL-módszer teljes koncepciója

A módszer kifejlesztése során figyelembe vett további logikai relációk közül a következő négy ítélhető a legfontosabbnak:

- szubsztitúciós reláció
- funkcionális reláció
- explanációs reláció
- effektuális reláció.

A szubsztitúciós reláció meghatározására az a felismerés indított, hogy a kutatási témák címében szereplő tárgyszavak – bizonyos határok között – úgy helyettesíthetők más tárgyszavakal, hogy a helyettesítéssel kutatásra érdemes további témák címét nyerjük. Az ilyen értelemben vett helyettesítési viszonyban álló tárgyszavakról azt állítottuk, hogy közöttük szubsztitúciós reláció áll fenn. Kézenfekvő gondolat volt, hogy a szubsztitúciós reláció felhasználásával valamely adott kutatási témából kiindulva más (értelmes, kutatásra érdemes) témákat generálhatunk.

Funkcionális relációban levő tárgyszavaknak minősítettük azokat, amelyek között matematikai értelemben vett függvény-kapcsolat áll fenn, illetőleg állapítható meg.

Az explanációs relációt tudományos megállapítások, tézisek között értelmeztük. Explanációs reláció áll fenn megállapítások, feltevések, vagy más – logikai értelemben ítéletnek minősíthető – mondatszerű képződmények között, ha létezik olyan érvényes következtetés, mely a szóbanforgó két állítást a premissza és konklúzió viszonyába hozza egymással.

Az effektuálisi relációt tudományos problémák, tudományos célkitűzések, hasznosítási lehetőségek és egyéb olyan tényezők között definiáltuk, amelyek valamilyen tudományos program megvalósítása szempontjából egymással kapcsolatban vannak. Effektuálási reláció akkor áll fenn az ebben az értelemben vett programelemek között, ha az egyik realizálása feltétele a másik realizálásának.

A szubsztuációs, a funkcionális, az explanációs és az effektuális relációk segítségével további gráfok szerkeszthetők, amelyek csúcsai a szubsztitúciós és a funkcionális relációk esetében tárgyszavak vagy változók (összefoglaló névvel terminusok), az explanációs és az effektuális reláció esetében pedig tézisek vagy problémák (összefoglaló néven kognitívumok). A terminusok és a kognitívumok egymással kapcsolatban levő különböző együtteseit átfogóan kognitív rendszereknek nevezzük.

Az eddigiekben jelzett logikai entitások (terminusok, kognitívumok, kognitív rendszerek) és a hozzájuk értelmezett különböző logikai (affiliációs, generikus, szubsztitúciós, funkcionális, explanációs és effektuális) relációk gráfban reprezentálható logikai modellek széles körének képzésére alkalmasak és a vizsgálódások során világossá vált, hogy ezeknek a modelleknek a felhasználásával a tudományszervezési és irányítási feladatok széles körének megoldása segíthető.

Kiváltképpen a következő nyolc feladat megoldása tűnt az ismertetett modellállomány felhasználásával segíthetőnek:

- tudományos kutatási információs és információ-visszakeresési rendszerek kiépítése:
- a tudományos kutatás tematikai statisztikája;
- a tudományos kutatási problémák modelljeinek konstruálása, integrálása és tipizálása;
- tudományos ismeretek rendezése, értékelése, tudományos elméletek alkotása;
- tudományágak, kutatási területek helyzetelemzése;
- tudományágak fejlődésének prognosztizálása;
- a tudományos kutatások koordinálása;
- a tudományos kutatások koncentrálása és az alternatív kutatási-fejlesztési programjavaslatok közötti döntés.

Az információs és statisztikai feladat megoldására elsősorban affiliációs és generikus model- használhatók. A modellszerkesztési és az elméletfejlesztési feladat megoldásába célszerű bele- nni az összes említett relációt. A helyzetelemzési és a prognosztizálási feladatban dominál effiliációs, az explanációs és az effektuálási reláció alkalmazása; a prognosztizálási feladatban zubtitúciós relációnak is fontos szerepe van. A koordinálási feladat megoldásában az affiliá- s reláción kívül elsősorban a generikus és a funkcionális a lényeges, a koncentrálási feladatban affiliációson kívül elsősorban az explanációs és az effektuálási.

A modellek kiépítése és alkalmazási lehetőségeinek vizsgálata során nyilvánvalóvá vált, hogy zülük többnek matematikai optimálása további hatékony eszközt adhat a tudományos kutatás nyításának és szervezésének kezébe. Igy pl. a minimális vágás elvének alkalmazása módot ad a, hogy bizonyos modellek segítségével meghatározzuk a kutatási problémák olyan együtte- amelyeket célszerű valamilyen külön szervezett program keretében irányítani. A gráfok csú- i fokszámának és élei multiplicitásának vizsgálata alkalmas lehet arra, hogy a modelleken belül an részgráfokat definiáljunk, amelyek a modellekkel reprezentált kognitív rendszerek tematikai ypontjainak tekinthetők. Az explanációs gráfok bizonyos strukturális jellemzőinek optimálásá- javaslatokat lehet generálni arra vonatkozóan, hogy a tudományos elmélete és empirikus eretszövevények továbbfejlesztése, axiomatikus rendezése milyen lépések, kutatási akciók ferálását teszik szükségessé. Effektuálási modellek optimálásával információkat szerezhetünk a vonatkozóan, hogy szerteágazó kutatási problémák területén mely problémacsoportok elő- be helyezése szolgálhatja a leghatékonyabban a területen fennálló gyakorlati célok megvaló- ását.

A modellek strukturális optimalizálásával elérhető eredmények jelentőségét felismerve a dszer kidolgozásával foglalkozó kutatócsoport megkezdje a matematikai optimálási proble- k megfelelő formulázását. a problémák megoldására alkalmas algoritmusok kidolgozását, jd a tényleges vizsgálatok elvégzéséhez szükséges gépi programok elkészítését. A különböző ikai relációk felhasználásával konstruálható modellek legtöbbjére vonatkozóan készültek gya- rlatilag is alkalmazhatónak tűnő algoritmusok és programok, s kisebb feladatokon ezek mind- yikét sikerült ki is próbálni.

A modellek és feladatok integrálásával felvázoltuk a tudomány generatív modelljének am- iózus koncepcióját s tudományszervezési alkalmazásának lehetőségeit. Mindezt az előadó rzletes monográfiában publikálta [1], s könnyen áttekinthető összefoglalást is közölt [2].

3. A LOGEL-módszer eddigi alkalmazásai

A módszer kialakulásával foglalkozó részben röviden ismertettük azt a nemzetközi vizsgá- t, amely a módszer első alkalmazásának tekinthető.

Ezt követően először olyan hazai alkalmazásra került sor, amelynek jellege és vizsgálata olytatásának körülményei is nagymértékben megegyeztek a CIB keretei között végzett nemzet- zi vizsgálattal. A hazai felhasználás során az Országos Kutatási Nyilvántartás adatai alapján vizsgáltuk, hogy 1968–69 folyamán a népgazdaság területén az építési ágazat problémáit ntően milyen kutatások folytak. Ez mintegy ötszáz kutatás téma vizsgálatát jelentette, vagyis ebb mennyiséget, mint a CIB számára végzett vizsgálatban. Ezzel szemben a témákra vonatko- an részletesebb információk álltak rendelkezésre, s ezért a témákat több és élesebb tárgyszóval emezhettük, mint az első esetben. A második vizsgálatban is ad hoc – a vizsgálat közben pített – tezaurusszal dolgoztunk és a témarendszert ebben az esetben is affiliációs és generi- s modellekkel reprezentáltuk.

A vizsgálat eredményeként jellemeztük a vizsgálat kutatási terület tematikai struktúráját tematikai súlypontokat és ezek kapcsolatait) s rámutattunk a koordinálás és koncetrálás mpontjából kiemelkedő fontosságú feladatokra.

A harmadik nagyobbarányú alkalmazásra az 1971–85. évi Országos Távlati Kutatási Te építési feladatainak meghatározása érdekében került sor, 1970-ben.

A LOGEL-módszerrel végzett vizsgálatok dokumentatív alapját itt egy kérdőívre befut javaslattömeg szolgáltatta, amely kereken két- és félszáz programjavaslattá volt rendezhető. Ezek segítségével a két korábbi alkalmazáshoz hasonló módon affiliációs és generikus modellen ket konstruáltunk és ezek vizsgálatára támaszkodva tettünk javaslatot az ágazat mintegy harmb kutatási-fejlesztési célprogramjára, illetőleg kutatási főirányára. E vizsgálat keretében használtr először a módszerhez tartozó gépi programokat, ez esetben CDC 3300 típusú gépen. A progra mok meghatározása során a tisztán tárgyszavakat tartalmazó affiliációs gráf minimális vágásán módszerét is használtuk.

Ezeket az alkalmazásokat az [1]-jelű monográfiában részletesen ismertettük. Az eddig e említett feladatokhoz hasonló és kapcsolódó volt az építési ágazat kutatási tervének kidolgoz az V. ötéves terv periódusára. Ennek a munkának a keretében mintegy ötszáz kutatási javaslare affiliációs és generikus feldolgozásából indultunk ki. A távlati tervezési munkától eltérően a gépi vizsgálatokat most az építési ágazat SIEMENS 4004 típusú gépével végeztük el. Matemait kai szempontból igényesebb vizsgálatainkat (a tisztán tárgyszavakat és tisztán témajavaslatok tartalmazó gráfok megkonstruálása, a témajavaslatok csoportosítása) a megfelelő tematikai kta cetrálódás koncepciójának kialakítása érdekében végeztük el [3].

Az eddig említett vizsgálatok mindegyikében eseti elemzésekről volt szó. Kezdettől fogta nyilvánvaló volt azonban, hogy a LOGEL-módszer leghatékonyabb alkalmazása olyan körülmtt nyek között várható, amelyek között egy kutatási terület folyamatosan üzemelő információs rendszerére támaszkodva lehet foglalkozni az irányítás problémáival és azok megoldásával. Ennek érdekében a hetvenes évek elejétől arra törekedtünk, hogy létrehozzunk ilyen informás ciós rendszert, azt bevezessük egy kutatási intézmény hétköznapi gyakorlatába és erre építve strukturális vizsgálatokat végezzünk. Ilyenfajta információs rendszer létrehozásához természe sen állandóan használt és továbbfejlesztett tezauruszra van szükség.

Ilyen értelemben folyamatosan üzemelő rendszert az Építéstudományi Intézetben hozt létre, először a kutatási szerződések nyilvántartási rendszerének kialakításával. A rendszert 1977-ben vezettük be és azóta üzemben van [4].

Előkészítettük, hogy hasonló elvek alapján létrehozzuk az építési ágazaton belüli főhat sági megbízásból végzett kutatási szerződések nyilvántartási rendszerét is. Az V. ötéves terv rán végzett munkák a módszer segítségével utólagosan már áttekinthetők és értékelhetők le Közbenső vizsgálatokat már eddig is végeztünk ennek a rendszernek a keretében.

Az előzőkön kívül az építési kutatások területén több más eseti vizsgálatra is került sor Ezekre vonatkozóan több publikáció jelent már meg a hazai és a nemzetközi szakirodalomban [5].

Néhány év óta a módszer iránt érdeklődés nyilvánult meg az építési ágazaton kívüli terü tekről, s külföldi érdeklődők is jelentkeztek. A lefolytatott tárgyalások hatására eddig két terü leten került sor a módszer alkalmazásának megkezdésére.

Ezek közül az első az Országos Számítástechnikai Kutatási Célprogram, amely az OMF irányítása alatt folyik. Itt a LOGEL-módszer felhasználásával alakították ki a célprogram szá mítógéppel segített irányítási információs rendszerét, s ezt az OMKDK-ba telepítették ESZ 10 típusú gép felhasználásával [6].

A közelmúltban a Könnyűipari Minisztérium döntött úgy, hogy a kutatások irányításán segítésére használatba veszik a LOGEL-módszert és programrendszert is.

4. Az alkalmazások értékelése és további kutatásai

Az eddigi alkalmazásokat áttekintve elsősorban azt kell megállapítani, hogy a rendelkezésünkre álló modellállomány egészét még egyetlen esetben sem alkalmazták. Az alkalmazások többsége az affiliációs és a generikus modellek felhasználására korlátozódott s csak elvétve került sor a többi reláció felhasználására.

A módszer alkalmazásának előnyei eddig elsősorban a kutatási információ, statisztikai és a helyzetelemzés feladataira vonatkozóan mutatkoztak meg, a tudományos problémák megoldására irányuló modellalkotás és a tudományos elméletek továbbfejlesztésére kidolgozott módszerek pedig egyelőre alig kaptak alkalmazást.

A módszer kidolgozására irányuló elméleti szakaszban megalkotott generatív modell egyelőre elméleti konstrukció maradt, amelynek gyakorlati értékét eddig még semmi sem bizonyítja, jóllehet az elméleti modellképzés szakaszában ennek a konstrukciónak nagy jelentőséget és jövőt tulajdonítottunk.

Az alkalmazás során lépten-nyomon azzal a problémával kellett megküzdeni, hogy a kutatási munkák nyilvántartásának a gyakorlata az Országos Kutatási Nyilvántartás létrehozásának ellenére is igen kezdetleges állapotban van. Az adatszolgáltatási fegyelem gyenge, a megkezdett kutatások jelentős hányadáról semmiféle információ nem áll rendelkezésre, a ténylegesen befutott adatszolgáltatások pedig minőségi szempontból okoznak igen nagy problémát. A kutatók túlnyomó többsége húzódozik a munkájára vonatkozó információs kötelezettségek teljesítése elől és sok esetben sajnos az irányító szervek sem látják még világosan a korrekt és átfogó országos kutatási adatszolgáltatás és statisztika jelentőségét; az irányító szervek gyakran kizárólagosan helyezik előtérbe az esetenként megkérdezett szakértők sokszor nagyon is szubjektív véleményét.

Tudományos életünket általában jellemzi a tudományos kutatási adminisztrációtól és bürokratizálódástól való félelem, s ezért a felelős irányító hatóságok általában húzódoznak attól, hogy a kutatások irányítására és koordinálására ütőképes irodákat, vezérkari jellegű irányítási egységeket hozzanak létre. Mindennek velejárója a megbízható kutatási információk nagyfokú hiánya, ennek pedig következménye az ad hoc informálási kötelezettségek burjánzó ismétlődése, az informálási minőség színvonalának állandó csökkenésével.

Mindezek a követelmények pillanatnyilag nem teszik túlságosan kedvezővé a kutatási irányítás számítógépes segítésének hazai kilátásait.

Döntően megváltozhat természetesen a szituáció, ha a kiterjedt tudományos kutatással járó népgazdasági szintű vállalkozások a jelenleginél nagyobb mértékben kerülnének előtérbe, s megnövelné az irányítás eszközeivel szembeni igényeket is.

Az ilyen természetű kedvező fejlemények kibontakozására váró irányítási kutatócsoportok a kedvezőbb körülmények kialakulásáig az eddigi eredmények megszilárdításával, kiegészítésével és az eredmények hasznosítására alkalmas személyi állomány kiképzésével foglalkozhatnak.;

Ezeknek a feladatoknak a LOGEL-módszer kidolgozásával és elterjesztésével foglalkozó kutatócsoport is igyekszik eleget tenni. A közelmúltban végzett matematikai és számítástechnikai továbbfejlesztő munka területén ki kell emelni a fentiekben már említett és tematikai súlypontkeresési módszerek továbbfejlesztésére irányuló munkákat. Ezen a területen kutatócsoportunk a klaszterelemzés nemzetközi mezőnyéhez kapcsolódva jelentős lépéseket tett a klaszteranalemzés bizonyos módszereinek a LOGEL-programrendszerbe való beépítésére, illetőleg a klaszterelemzési módszerek továbbfejlesztésére is. Ebben a vonatkozásban jelentős matematikai és számítástechnikai munka folyt, részben az ÉTI-ben, részben pedig a KSH-ban [7, 8].

A munka eredményeit ez év tavaszán a Neumann János Számítógéptudományi Társaság operációkutatási szakosztályának, valamint a Magyar Tudományos Akadémia rendszertechnikai és statisztikai bizottságának részvételével rendezett klaszterelemzési ankéton ismertettük. Ezen az ankéton több rokonszándékú hazai törekvés is találkozott és hasonló találkozás nagyon indokolt lenne a tudományos kutatás számítógéppel segített irányításainak átfogó témaköreiben is.

Előadásunk bevezetését idézve, azzal a reménnyel zárjuk beszámolónkat, hogy a Neumann János Társaság Kongresszusának vitái hozzá fognak járulni a LOGEL-módszer alkalmazásával rokon hazai törekvések helyzetének áttekintéséhez és ahhoz, hogy kialakuljon egy olyan összehangolt fejlesztési koncepció, amely lehetővé tenné a számítógéppel segített tudomány-irányítás fejlődésének meggyorsítását.

Irodalmi hivatkozások

[1] Kunszt György: A tudományos kutatás logikai modellezése és tematikai irányítása. Akadémiai Kiadó, Budapest, 1975.

[2] Kunszt György: Tudományszervezési és tudománypolitikai problémák logokibernetikai megközelítése. Magyar Tudomány, 1975. 8–9. sz.

[3] Kunszt György és szerzőkollektíva: Az építési ágazat V. ötéves tudományos kutatói terve (Tervezet). Építéstudományi Intézet, Budapest, 1975.

[4] Grád Gusztávné: A LOGEL-rendszer gyakorlati alkalmazása az Építéstudományi Intézetben. Tudományos Műszaki Tájékoztatás, 1979.

[5] Csorba, E.–Grád, J.: Building research in developing countries-analysis of topics and priorities. CIB Seminar, New Delhi, 1977.

[6] Futó Péter–Gazsi Endre: LOGEL ESZ 1020 programrendszer. Építéstudományi Intézet, Budapest, 1976.

[7] Futó Péter: A clusteranalízis egy új modellje és algoritmusa. SZIGMA, 1977. 4. sz.

[8] Futó Péter: Hipergráf modellen alapuló klaszterelemzés. Kandidátusi értekezés. Budapest, 1

SEJTPROCESSZOROK TERVEZÉSE ÉS PROGRAMOZÁSA

Legendi Tamás
MTA AUTOMATAELMÉLETI KUTATÓ CSOPORT

A dolgozat összefoglalja a hazai technológiával létrehozható, számítási rendszer részeként emeltethető sejtprocesszorok rendszertechnikai elveit (és előnyeit), ismerteti a nagy hatékonyságú bitpárhuzamos sejtprogramok készítésében elért eredményeket.

Sejtprocesszorok létrehozásának aktualitása

Részletes indokolás található (Legendi, 1976.) és (Legendi, 1978.)-ban, itt tömören felrolást adunk:

- a hagyományos számítógépek sebessége korlátozott, az elemi hardware elemek szintjén a kihasználtság igen alacsony
- az LSI technológia homogén alapelemet igényel; nagy sebességű elektronikus háttértárak létrehozását teszi lehetővé, ami a hagyományos architektúra egyensúlyát megbontja (nagy sebességű párhuzamos működésű processzort igényelve); javítja a nagy elemszámú sejtprocesszorok megvalósításának lehetőségét
- rekonfigurálható architektúrák iránti igény, ezen belül különösen az egyes algoritmusokhoz dinamikusan igazítható architektúrák trendje.

A területtel foglalkozó irodalom erőteljes gyarapodása, az épülő (megépült rendszerek iolay, 1969.), (Duffi 1975.) sorozatgyártási előkészületek (EVM-leírás, 1977.) is egyre nyilnvalóbbá teszik a terület növekvő fontosságát.

Megfontolandó, hogy a hazai kutatási-fejlesztési politika keretén belül – a követő fejlesztseknél ugyan valamivel nagyobb kockázat vállalásával – nem lenne-e célszerű növekvő támogast biztosítani az újelvű, ugrást előirányzó, megfelelően megalapozott terveknek, mivel a gyakorazt mutatja, hogy a követő fejlesztésekkel a gap nem csökken megfelelően.

A továbbiakban bemutatjuk egy összefüggő koncepció néhány elemét, amely az eddigi méleti és gyakorlati eredményekhez képest nagyobb méretű, sokkal gazdaságosabb sejtprocesszorok rendszertechnikáján és részletesen kidolgozott nagy számú sejtalgoritmuson alapul.

A javasolt rendszertechnika

Részletes kifejtés található (Legendi, 1976 a)-, (Legendi, 1978.)-ban, itt a főbb elemeket ljuk meg:

niverzális felhasználási célú, mikroprogramozható sejtprocesszorok főbb elemei:

CPU – a sejtmező vezérlésére, az I/O forgalom kezelésére
CPU RAM 1: a CCPU működtető programját tartalmazza
CPU RAM 2: a sejtmezőt működtető mikroprogramot tartalmazza
ufferek: a sejtmező adat–I/O kiszolgálására
ejtmező: gyártástechnológiailag homogén felépítésű, azonos (kvázi) sejtekből áll, sejtenként és lépésenként változtatható mikroprogram hozzárendelést biztosít (térben és időben inhomogén sejttér).

A rendszernek a megvalósíthatóságát (a rendszer méretének a megépíthető rendszernagy alá szorítását) a soros-párhuzamos működésű kvázi-sejt rendszer, a programozhatóságot ill. a gazdaságos méretű és nagy párhuzamosságú programok létrehozását pedig a térben/időben inhomogén működés biztosítja.

Speciális célú alkalmazásokra speciális sejtprocesszor architektúrák is kialakíthatóak: pl. az általános célú rendszerben általában egy összefüggő sejtmezőt alkalmazunk – folyamatszabályozási rendszerekben célszerű több, kisebb, de egymással változtatható módon is összekapcsolható sejtmező alkalmazása.

3. Célsejt-hardware

Egyes műveletek behuzalozott megvalósítása jelentős alkatrészmegtakarítást és komoly sebességnyereséget jelenthet, és igen gyakran előforduló műveletek esetén létjogosult lehet. Tobek között (Dakovski, 1976.) is javaslatot tesz konkrét célhardwarekre, valamint általános tervezési metodológiára is.

Több tervünk (szortoló, mátrixszorzó, adattömörítő) közüli a szortoló célhardware a legérettebb, a (Legendi, 1977a)-ban közölt n lépéses sejtalgoritmus átültetése. (Hazai n–MOS technológiát feltételezve 64 tok szükséges 1 Kbyte rendezéséhez, ez 1 Mbyte/sec adatáramlási-feldolgozási sebességgel történhet. A rendezés modulárisan bővíthető mind a szószám, mind a szó hossz irányában, a sebesség a szóhossz növelésével lineárisan nő.)

4. Szimulációs nyelvek

A néhány ismert sejtautomata-szimulációs nyelv a kifejlesztett rendszer szimulációjára nehézkesen vagy egyáltalán nem alkalmazható, ezért célorientált szimulációs nyelvcsaládot fejlesztettünk ki: (Legendi, 1977b), (Legendi, 1975.), (Molnár, 1977a), (Hegedüs, 1979.):

CELLAS 1.0	CDC–3300	80 utasítás	batch	FORTRAN–IV	18 Kso
INTERCELLAS 1.0	TPA–i	50 utasítás	párbeszédes	ASSEMBLY	8 Kso

Ezeket a rendszereket mindennapos használatba vettük, továbbfejlesztésük folyik. Viszonylag nagy méreteik mellett/ellenére egyszerű szerkezetűek, egyszerű adatstruktúrákat tartalmaznak, bemenő nyelvük igen egyszerű. Az egyes rendszerek körülbelül 1/4-ét kitevő függvénydefiniáló alrendszerek felépítése és funkciója viszonylag bonyolultabb, a rendszer legfőbb értékét jelenti az elég magas szintű, adekvát függvényleíró résznyelv (Legendi, 1979c).

5. Sejtprogramok

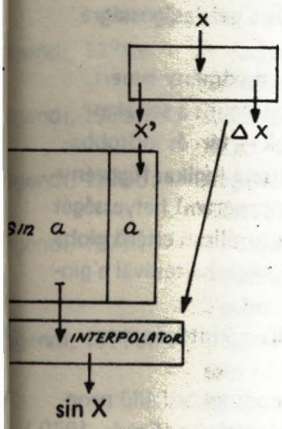
Viszonylag kevés sejtalgoritmus ismert, és ezek egy része is csak elméleti jelentőségű (Vollmar, 1979.), (Fáy, 1976.). Az általunk kidolgozott kb. 100 sejtalgoritmus (Legendi, 1978), (Legendi, 1977a), (Legendi, 1979a) és néhány összetett sejtprogram (assembler) (Molnár, 1979), adatfeldolgozó rendszer, folyamatszabályozási rendszer (Legendi, 1978b), jól mutatja az alkalmazhatóságot, esetleges későbbi sejt-alapsoftware magja lehet.

Az algoritmusok közös jellemzője a (kisméretű, 50–80 kapuból álló) sejtek 70–100%-ának olyamatos hasznos működtetése. A kidolgozott sejtalgoritmusok alapelveit és két konkrét algo-tmust (rendezés, összeadás) ismertet (Legendi, 1977a), a létrehozott sejtalgoritmusok jó áttekin-ését adja (Legendi, 1978.), részletes leírásokat tartalmat (Legendi, 1979a).

Itt tömören összefoglaljuk a főbb sejtprogramozási alapelveket, jellemezzük az eddigi sejt-lgoritmusokat, egyszerű példával illusztrálva.

Sejttérben két fő párhuzamosíthatósági lehetőség van: független műveletek egyidejű vég-ése; egymást követő műveletek pipe-line-ba szervezése és (különböző adatokkal történő) átla-olt elvégeztetése (v. ö. data-flow programozás). Ezeket a lehetőségeket egyidejűleg több szinten kihasználjuk: az egyes algoritmusokat a sejttérben egymással összekapcsolt, (az adott algorit-musnak legjobban megfelelő funkciójú, számú és méretű) egyidejűleg működő műveletvégző egységekre képezzük le; a műveletvégző egységeken belül is bitpárhuzamos algoritmusok kerülnek végrehajtásra.

Például könyvtári függvényszámító (sinus, ...stb.) eljárás:



$$X = X' + \Delta X$$

asszociatív tár

t az egész szerkezet pipe-line-ban működik, folyamatosan fogad input adatokat, ad output edményeket; nagy mértékben kihasználva az asszociatív tár 100%-osan párhuzamos belső ködését (Legendi, 1979a).

Az asszociatív tár mellett stack és (gyűrűbe is kapcsolható!) shift regiszter („software enger") típusú memóriaelemeket alkalmazunk (címezhető tár kialakítása igen gazdaságtalan) nikor „az átáramolva feldolgoztatás" mellett feltétlenül szükség van stabil tárolási funk-óra is.

A konkrét szerkezetek körét az n számot n lépésben rendező algoritmus nyitotta meg Legendi, 1977a), ez párhuzamosan végrehajtható párcserén alapszik (az egyes párcsere-lépések dig pipe-line-ba vannak szervezve). Sok bináris aritmetikai művelet algoritmusa áll rendelkezés-: n szám összeadása/kivonása (tetszőlege szóhosszú) operandusonként egy-egy billenési lépés att;* (n elemű vektorok ill. nxn mátrixok esetén), iteratív osztás stb.

Decimális aritmetikai műveletek is működnek különböző reprezentációkban (BCD; egy ámjegy/egy sejt), egyes műveletek meglepően gyorsan végezhetők el, pl. tetszőleges számú cimális jegyből álló decimális számoknak kettővel történő osztása egyetlen billenési lépést ényel (decimális-bináris konverterben is jól alkalmazható).

vektorok skaláris szorzása, mátrixszorzás $O(n)$ lépés alatt.

Tetszőleges N-alapú számrendszerben végzett alapműveletek is működnek, sőt maradékos számrendszerben is működik (átvitel_ nélküli!) összeadás, kivonás, szorzás.

A további alapszerkezetek közül kiemeljük még a logikai kifejezések kiértékelésére szolgáló modult, az nxn-es logikai mátrixok szorzását n lépésben végző modult (gráfbejárás, relációk tranzitív lezárának számítása stb.), a bináris és tetszőleges modulusú számlálót, a számsorozat generátorokat (lépésenként adnak-tetszőleges szóhosszúságú — egy-egy újabb elemet: különösen a véletlenszám generátorokat.

Az ennek a pontnak az elején felsorolt összetett szerkezetekben hatékonyan sikerült alkalmazni (összeépíteni) az alapszerkezeteket.

6. Sejtalgoritmusok verifikálása

A sejtprogramok készítése jelenlegi tapasztalataink birtokában már egyre könnyebb, de egyelőre még nehezebb feladat a hagyományos programozásnál; a tesztelés pedig nehézkes. A kézzel történő helyességbizonyítás igen munkaigényes, az egzaktságra és a gazdaságosságra való törekvés egyaránt a feladat automatikus megoldását igényli.

A szekvenciális algoritmusok verifikálásában már sok, bár nem teljes eredmény ismert, a párhuzamos algoritmusok verifikálásában kezdeti eredmények ismertek. Végülis a sejtalgoritmusok verifikálása azért nem okoz elvi problémát, (gyakorlati akadályok — tár- és időrobbanás — természetesen felmerülhetnek), mert a lokális átmenetfüggvények tiszta logikai függvényként kezelhetők. A jelenlegi megoldás a lokális átmenetfüggvények (mikroprogram) helyességét szimbolikus szimulációval ellenőrzi;- a lokális leírástól minőségileg és strukturálisan eltérő globális leírásban megadott kezdőállapotból kiindulva a szimbolikus szimuláció végrehajtásával a globális leírásban megadott végállapotba kell jutni,

A rendszer PROLOG nyelven implementáljuk a HONEYWELL 6060 számítógépen (Fehér, 1979.).

Kezdeti matematikai eredmények és egy PROLOG és FORTRAN modulokból álló programrendszer alkotja az első kísérletet átmenetfüggvények (korlátozott) szintézisére (Fehér, 197

7. Sejtprocesszorok digitális architektúrába történő integrálása

A hardware szintű illesztés igen fontos, de lényegében egyszerű: a sejtprocesszor I/O egységként csatolható, BMA-n keresztül kommunikálhat, memóriamodulként épülhet be a rendszerbe.

A software illesztés sokkal bonyolultabb probléma és többféle megoldási javaslat vethető fel.

A jelenlegi sejtprogramozás szimulátorok segítségével történik: ez mikroprogram ill. gépi kódú szintű programozást jelent (bár a szimulátorok sok szolgáltatást, kényelmet nyújtanak).

A legtermészetesebb lépés sejt-assembly nyelv, majd magasabb szintű sejt-nyelvek létrehozása lehetne. Assembly szinten paraméterezhető műveletvégző egységek hívása és összekapcsolása elegendő (Legendi, 1977a). A főbb problémát ilyen nyelveknek a gyakorlatba való bevezetése jelentené.

A több kínálkozó lehetőség közül a gyakorlatban legkönnyebben megvalósítható, már viszonylag kis sejtmező méret (50 ezer sejt — kb. 1000 tokos rendszer hazai technológiát feltételezve) mellett is hasznosan alkalmazható megoldást hagyományos makro assembly nyelv új rendszermakrokkal való bővítésével érhetünk el. Az új rendszermakrok automatikusan a rend-

szerhez tartozó sejtprocesszorban kerülnek végrehajtásra (párhuzamosan az egyéb processzor(ok) működésével). A rendszermakrokat (előkészítés, sejtprocesszorba töltés stb.) és az azok funkcióit megvalósító sejtprogramokat rendszersoftwares ill. sejtprogramozó írja meg, az átlagos felhasználó előtt ezek rejtve vannak.

Igy az assembly szintű felhasználónak nem kell sejtprogramban gondolkodnia, lényegében coroutine-ok állnak a rendelkezésére. Ha ezen a bővített makro-assembly nyelven operációs rendszer modulokat, vagy magas szintű nyelv futási idő alatti (run-time system) könyvtárának egyes részeit megírják, akkor a továbbiakban a munkavezérlő nyelven ill. a magasszintű nyelven író felhasználó számára áll közvetlen rendelkezésre a sejtprocesszor.

Ennek a rendszernek a kipróbálására jelenleg készül egy — egyelőre természetesen szimulált sejtprocesszort tartalmazó — konfiguráció, szimbólumtáblából keresés, rendezés stb.) (Dévay, 1979). 1979.)

Irodalom:

Legendi, 1976a: T. Legendi: Cellprocessors in Computer Architecture
Computational Linguistics and Computer Languages XI. 1976.

Legendi, 1978.: Legendi T.: Homogén számítási rendszerek éa alkalmazásuk I.
SZKI számára készült tanulmány 1978.

Legendi, 1978b: Legendi T.: Homogén számítási rendszerek és alkalmazásuk II.
SZKI számára készült tanulmány 1978.

Legendi, 1977a: T, Legendi: Programming of Cellular Processorts Braunschweig Cellular Meeting June 2—3, 1977 Informatik Berichte Nr. 7703
Technische Universität Braunschweig, 1977.

Legendi, 1977c: Legendi T.: A 2D Transition Function Definition Language for a Subsystem of the CELLAS cellular Processor Simulation Language
to appear in: Computational Linguistics and Computer Langauages XIII. 1979.

Legendi, 1977b: T. Legendi: Intercellas — an Interactive Cellular Space Simulation Language
Acta Cybernetica (1977) Tom. 3.

Legendi, 1975.: Legendi T.: Sejtautomaták szimulációja.
A CELLAS szimulációs nyelv.
Számítógépes szimulációs módszerek és alkalmazásuk a műszaki, orvosi és közgazdasági tudományokban, Pécs, 1975, szept. 2—3.

Legendi, 1979a: Legenfi T.: Sejtalgoritmusok
Párhuzamos Számítási Rendszerek 1979/4.
(az NJSZT Párhuzamos Számítási Rendszerek Szakcsoport és a SZTAKI közös kiadványa)

Legendi, 1979b: Sejtprocesszor bibliografia, 1979
(300 tétel, ebben 15 bibliográfia, amelyek 2600 tételt tartalmaznak)
Számítógépes lista az MTA Aut. elm. Kut. Csop. házikönyvtár anyagánál.

Molnár, 1979a: Legendi T.—Molnár G.—Székely L.: CELLAS felhasználói kézikönyv 1979. április

Molnár, 1979b: Molnár G.: A CELLAS sejtautomata szimulációs nyelv implementálása és alkalmazása sejtalgoritmusok létrehozására
ELTE TTK Diplomamunka, 1979.

Hegedűs, 1979.: Legendi T.—Hegedűs Gy.—Pávölgyi L.: INTERCELLAS felhasználói kézikönyv 1979, március

Dévay, 1979.: Dévay Á.: Sejtprocesszorok közvetett programozása
ELTE TTK Diplomamunka, 1979.

Fehér, 1979.: Fehér Á.: Sejtprogramok verifikálása és szintézise
ELTE TTK Diplomamunka 1979.

Vollmar, 1979.: R. Vollmar: Algorithmen in Zellular Automaten Teubner Studienbücher, Stuttgart 1979.

Fáy, 1976.: Fáy Gy.: Cellular Design Principles: a Case Study of Maximum Selector in CODD—ICRA Cellular Space I—II.
Computational Linguistics and Computer Languages XI., XII. 1976, 1978.

Golay, 1969.: M. J. E. Golay: Hexagonal Parallel Pattern Transformations
IEEE Transactions on Computers, Vol. C—18. No. 8. August 1969.
K. Preston jr.: Use of the Golay Logic Processor in Pattern—Recognition Studies Using Hexagonal Neighbourhood Logic
Symposium on Computers and Automata Polytechnic Institute of Brooklyn 1971., Ápril 13—15.

Duff, 1975.: M. J. B. Duff—D. M. Watson: A Cellular Logic Image Processor
In: Cainello E. R. (Ed.) New Concepts and Technologies in Parallel Information Processing, Noordhoff—Leydén 87—103, 1975.
C. D. Stamopoulos—M. J. B. Duff—D. M. Watson: Some Aspects of the Logic Functions of CLIP 3
In: Caianello E. R. (Ed.) New Concepts and Technologies in Parallel Information Processing, Noordhoff—Leyden 75—86, 1975.

EVM-leírás, 1977.: EVM PSZ—300 homogén bázisú számítógép leírása „ELVA" Tbiliszi 1977.

Dakovski, 1976m: L. G. Dakovski—E. L. Anavi: Cellular Array for Conveyor Processing of Polinomial Codes.
IV. National Conference an Computer Science, Varna, 1976. Sept.

ÉPÍTŐIPARI TERMELÉSPROGRAMOZÁS MINI SZÁMÍTÓGÉP SEGÍTSÉGÉVEL

Dr. Lengyel György
Pest megyei Állami Építőipari Vállalat

Általában megfigyelhető, hogy iskolapéldákban, de megvalósult számítógépes megoldások ismertetései között is lényegesen többször hallunk, olvasunk telepített ipari, mint építőipari esetekről.

Ennek elsősorban az az oka, hogy a telepített iparban, különösen a gépgyártásban sokkal nagyobb múltra tekinthet vissza a szervezés, mint tevékenység is. Az építőiparban a szervezés „intézményesítése" a vállalatok gyakorlatában csak a hetvenes években kezdődött el és e tevékenység számítógépes segítése ma is ritkább, mint a népgazdaság egyéb területein. Nem beszélek itt természetesen az ügyviteli munkában már itt is általános megoldásokról, az anyagkönyvelésről, állóeszköznyilvántartásról. Ezek az építőipar területén is általánosan alkalmazottak, többnyire az ágazat központinak nevezhető számítógépparkján, az ÉGSZI-nél és a FÜTI-nél.

A gazdálkodási elemek a nyilvántartási alrendszereken belül már csak szórványosak, termelésirányítási célokra pedig a számítógépet még ennél is ritkábban használnak.

Annak, hogy az építőipari termelésirányítás korszerű, számítógéppel segített módszerei oly nehezen alakulnak ki, számos oka ismert. A legfőbb okok az építőipari termelés sajátosságaira vezethetők vissza.

Köztudott, hogy az építőipari termelés során

– a termék áll elmozdíthatatlanul és a termelő apparátus mozog;
– úgyszólván minden épület egyedi termék, sorozatgyártásról alig beszélhetünk;
– a munka főként szabadban folyik, erősen kitéve az időjárás viszontagságainak;
– az általában körül sem kerített munkaterület munkaszervezési, munkafegyelemmel összefüggő problémái igen nagyok;
– a technológia nem olyan egyértelműen rögzíthető, mint más ipari tevékenység esetében, még a technológiai sorrend vonatkozásában is felállítható több egyaránt célszerű alternatíva adott esetekben;
– területileg szétszórt, viszonylag kis szervezeti egységekben folyik a munka;
– kis területen számos – esetleg nem is egy vállalathoz tartozó – szakma, illetve szervezet együttműködéséről van szó;
– igen nehéz elfogadható szociális ellátottságot biztosítani.

Az építőipari termelésirányításnak ennek megfelelően rugalmasabban kell alkalmazkodnia az adott körülményekhez.

Ha megvizsgáljuk egy vállalat termelésirányítási rendszerének alkotó elemeit,

– a rendelésnyilvántartási,
– a termeléselőkészítési,
– a termelésprogramozási és
– az effektív építésvezetési

folyamatokat, azt lehet megállapítani, hogy ezekből a termelésprogramozási alrendszer az, amelyet viszonylag a legkönnyebben lehet számítógép-orientáltan megszervezni.

Itt is felmerül azonban egy komoly probléma. Termelést programozni úgy, hogy a program reálisan teljesíthető is legyen, csak jó minőségű, folyamatosan karbantartott normatíva

rendszerre lehet építeni. Erre pedig az ÉKN nem alkalmas. Az ÉKN ugyanis országos átlagokat tartalmaz, s így magán viseli az átlagoknak azt a tulajdonságát, hogy az konkrét esetre hasem igaz.

Ez azt jelenti, hogy vállalati termelésprogramozás csak úgy valósítható meg, ha előzőleg elkészül a vállalati sajátosságokat tükröző normatívarendszer és biztosítható ennek folyamatos — célszerű időközönként rendszeresen ismétlődő — karbantartása. Számítógépes feldolgozás pedig akkor valósítható meg, ha ez a normatívarendszer törzsadatként alkalmazható.

A vállalati sajátosságokat tükröző normatívarendszer kidolgozása kevés építőipari vállalatnál valósult meg eddig. Ennek is van egy olyan oka, amelynek feloldása elég nehéz.

A saját vállalati normatívarendszer olyan adatokat tartalmaz, amely az adott vállalat legbensőbb titkait felfedi. Gondoljuk meg, egy kapitalista vállalat esetében az ilyen adatok hétcsétes titkok! Néhány ember ismeri ezeket! És aki ismeri, az tudja, hogy pl. meddig mehet a vállalat egy versenytárgyaláson, hogy milyen munkának mennyi a nyereségtartalma, stb. Nos, legyünk őszinték. Ma már egy szocialista vállalatnak sem áll érdekében ilyen jellegű adatait közkinccsé tenni. Ilyen körülmények között igazán célszerűen saját vállalati normatívarendszert csak saját számítógépen helyes alkalmazni.

Ebben az esetben ugyanis a termelési programhoz szükséges adathalmaz segítségével valóságos, az adott vállalatra egyértelműen érvényes erőforrás szükségletek állapíthatók meg és reális anyag- és költségelszámoltatás végezhető, ugyancsak számítógép segítségével.

Saját számítógépe azonban kevés építőipari vállalatnak van és anyagi eszközök sem állnak mindenütt rendelkezésre ilyenek beszerzésére. Különösen érvényes ez utóbbi megállapítás olyan esetekre, amikor valamilyen nagy, vagy akár csak közepes számítógépre gondolunk.

A felsorolt problémák megoldásához segítségként, elgondolkoztató példaként ismertetném saját vállalatom, a Pest megyei Állami Építőipari Vállalat megoldását.

1975. áprilisában — kb. 2 éves előkészítő szervezés után — vásároltunk egy magyar gyártmányú — VILATI készítette — Practicomp 4000 típusú mini számítógépet. Ennek jelenlegi kiépítettsége:

- 8 K szó kapacitású (1 szó = 32 bit) központi egység
- 1 db 800 K byte kapacitású MOM DISC (fix lemezegység)
- 4 db floppy-disc meghajtóegység,
- 2 db egyidejűleg on line kapcsolható Prepaline egység, amelyek lyukszalag lyukasztó, lyukszalag olvasó és consul írógép egységekből állnak és
- 1 db DZM—180-as mozaikprinter.

Azzal az elképzeléssel állítottuk munkába a berendezést, hogy annak optimális kihasználtságát biztosítandó, 5—6 év alatt minden külső számítógépes feldolgozásunkat megszüntetjük és az addigiakhoz képest néhány új rendszert is ennek segítségével oldunk meg. E célkitűzés megvalósítása érdekében a gépet a szervezési osztály üzemelteti, jelenleg két műszakban.

A saját számítógép felhasználási lehetőségei között kiemelt jelentőségű a termelésprogramozás. 1976-ban kezdtünk hozzá a szervezéshez és 1978. II. félévére próba-programot, 1979-re pedig már valóságos durva- és finom programot készítettünk a gépen.

A korábban elmondottaknak megfelelően mindenekelőtt kidolgoztuk a vállalati normatívarendszert. A vállalat tevékenysége során a tapasztalatok alapján előforduló valamennyi munkafolyamatnak megfelelő, mintegy 2500 db összevont folyamatnormát készítettünk. A normagyűjteményt VKN-nek (vállalati költségnormák) neveztük el.

A VKN egyes tételei — a tevékenység kódszámán, megnevezésén és mértékegységén kívül — tartalmazzák az egységre eső munkaerő szükségletet szakmánként órában, gépszükségletet gépfajtánként órában és anyagszükségletet anyagfajtánként természetes mértékegységben.

A normatívarendszerben figyelembe vett szakmák száma 27. Ez magában foglalja a vállalatnál előforduló valamennyi munkafolyamat ellátásához szükséges szakmát, bizonyos össz

onásokkal. Pl. az adattárban „csőszerelő" szerepel és nincs külön központi fűtés, vízvezeték, ázvezeték szerelő. A munkaerő törzsadattárban szerepel a szakmánként a vállalatnál kialakult tlagórabér.

Nyilvánvaló, hogy az ismertetett adatokból teljesen pontos munkaerő- és létszámszükség- t nem számítható, hiszen nincs figyelembe véve a munkakategória sem. Ha számításba veszzük viszont, hogy vállalati átlagórabérekkel dolgozunk, igen erősen közelítő értékeket kapunk. Az a megállapításunk, hogy ennél nagyobb pontosságra nem érdemes törekedni, mert – a kobban már elmondott problémák ismeretében – az iparban lehetetlen a munkaerő tervezése orán jobb eredményt elérni. Gondoljunk arra, hogy az építőiparban a technológiai fegyelem etartása sokkal nehezebb, mint másutt. A technológia sem olyan kidolgozott, így a telepített parban sikerrel alkalmazott munkaszervezési módszerek, pl. a 3M módszer bevezetése elképelhetetlen. E mellett gyakran előfordul, hogy – a munkaerőhelyzet adott munkahelyi feltéelei között – a dolgozók nem éppen saját szakmájuknak megfelelő munkát végeznek. Ehhez árul még az a probléma is, hogy az építőipari dolgozók általában brigádban dolgoznak. Egy-egy rigád többnyire egy faluból kerül ki. A brigádok létszáma adott. Nem biztos, hogy a munka lvégzéséhez legcélszerűbben szükséges létszám áll rendelkezésre, hanem egy adott brigád. Ez nmagában is bizonyos kompromisszumok elfogadását teszi szükségessé. Mindez azt jelenti, ogy az említettnél nagyobb pontosságra törekvés már nem gazdaságos.

A gépszükségleti normák készítésekor a vállalatnál alkalmazott „legfontosabb" 32 gépfaja figyelembe vételére került sor. A törzsadattárban ezeknek a vállalatnál kialakult 1 órára eső nköltsége vagy bérleti díja szerepel.

Ebben a kérdésben is viták alakultak ki. Először csak az úgynevezett vezérgépeket akaruk szerepeltetni, ezek száma 4–5 féleség. Később a „legfontosabb" meghatározást alkalmazuk. Természetesen nincs akadálya a féleségek száma szaporításának sem.

A normatíva készítésnél figyelembe vett anyagféleségek száma mintegy 800. Természetesen az ésszerűség korlátai közt ez a szám is változtatható, gyakorlatilag igen nagy mértékben. ény azonban, hogy ez a 800 féle anyag adja a felhasználási érték volumen 75–80%-át.

Az anyag-törzsadattárban szerepel az anyagok kialakult beszerzési átlagára. Ezen kívül nyagféleségekként szerepel egy kódszám, amely azt mutatja meg,hogy az illető anyagfajta álalában közvetlenül az építéshelyre kerül a szállítótól, vagy a központi raktáron keresztül jut da. Ha most – szintén kódszámok segítségével – megadjuk az illető anyagfajta szállítására és kodására vonatkozó jellemzőit (pl. ömlesztett, darabos 1 kg-on aluli, 1–5 kg-os, stb.), létetményenként pedig megadjuk az építéshely km-ben mért távolságát a központi raktártól, a ásodlagos fuvarok szervezésére és költségeire vonatkozó értékes adatok nyerhetők.

Ahhoz, hogy a felsorolt normatíva adatok, illetve az egyes erőforrásokból figyelembe ett tételszámok jobban érzékelhetők legyenek, elmondom, hogy a Pest megyei ÁÉV 2600 ős, évente kb. 1 milliárd termelési értéket előállító szervezet és egyidejűleg mintegy 150–200 tesítményen dolgozik. Munkái Pest megye területén folynak, s bár budapesti székhelyű, a övárosban csupán Csepelen van néhány munkahelye. A megyei vállalatok közül a leginkább ekoncentrált. Kapacitásának mintegy felét lakásépítésre fordítja, egyre több házgyári elem lhasználásával. Saját házgyára nem lévén, az elemeket vásárolja. A kapacitás másik felét yárak, iskolák, szociális intézmények, stb. építésére fordítja.

A VKN 1976-ban készült azóta minden elemét rendszeresen karbantartjuk.

A termelésprogramozás hálós technikával készül. Ennek számítógépes működtetéséhez 0 „hálós tevékenység"-et határoztunk meg, ezeket úgynevezett „készültségi fok"-ba csoporosítottuk.

A programozás menete az ismertetett alapfeltételek kialakítása és a számítógépbe tárois után röviden a következő:

A munkák elvégzéséhez a vállalat rendelkezésére bocsátott és a vállalat által észrevétele jóváhagyott költségvetések jelentik a program alapját. Az első lépés az, hogy az ÉKN-nek felelő szerkezetű költségvetés adatait, vagyis annak minden sorát átkódoljuk a VKN tételé megfelelően. Ezt a munkát a termelést és annak feltételeit ismerő, műszaki szakemberekbő álló termelésprogramozási csoport végzi. A kódolás eredményeként olyan lyukszalag készül amely költségvetési soronként tartalmazza

- a VKN kódot
- a hálós tevékenység kódját
- a mennyiséget és
- a költségvetés szerinti értéket.

Ezeknek a szalagoknak és a gépben tárolt törzsadatoknak segítségével teljesítményen erőforrás szükségleti táblák készíthetők. Ezek hálós tevékenységenként, illetve teljesítmény összesen vonatkozásában tartalmazzák a munkaerő- a gép és az anyagszükségleteket termés tes mértékegységben és értékben.

A munkaerő szükségleti táblákon minden hálós tevékenységet követően megjelenik e variációs sor, amely megmutatja,hogy az adott tevékenységet hány hét alatt hány fő végzi a VKN alapján. A variáció maximális száma 16, de a gép abba hagyja kiírásukat, ha a létsz szükségletet 0,8 „fő" alá csökken. Ismét visszautalok a korábban ismertetett munkaerővel csolatos helyzetre. Gondolom egyértelmű, hogy a létszám tekinthető a fő korlátnak, s az hogy a hétnél pontosabb időegység nem használható gazdaságosan.

Az erőforrás szükségletek táblái a teljesítményre vonatkoznak, függetlenül attól, hog nak megépítése milyen intervallumban folyik. Ezek tehát még nem tartalmaznak semmine ütemezést.

A munkaerőszükségleti táblák birtokában a termelésprogramozási csoport helyismeret (pl. létszámhelyzet ismeretével) is rendelkező műszaki szakemberei meghatározzák a hálós vékenységek sorrendjét, kapcsolati időit hétben, az egyes hálós tevékenységek átfutási ide szükséges létszámát. Utóbbi vonatkozásában csak a számítógép által kiszámított és a munk szükségleti táblára nyomtatott variációk egyikét lehet feltüntetni.

Ezeknek az adatoknak kódlapokra rögzítése után megtörténik ezeknek lyukszalagra is. Ennek a szalagnak beolvasása után a számítógép hálós eredménylapot nyomtat ki létesí nyeként, amely hálós tevékenységenként tartalmazza:

- a termelési értéket,
- a szükséges létszámot,
- az átfutási időt,
- a legkorábbi kezdést,
- a legkorábbi befejezést,
- a legkésőbbi kezdést,
- a legkésőbbi befejezést,
- a tartalékidőt

és kiszámít egy nem pontosan a számviteli fogalmat takaró, de azt megközelítő befejezett fejezetlen termelési értéket.

Ez a hálós eredménylap abszolut idős, tehát minden munka a 0-dik héten kezdődik valameddig tart.

Ezt követi a termelésirányítási apparátus részéről annak közlése, hogy a létesítmény kor kezdődhet el legkorábban, mikorra kell feltétlenül befejeződnie (szerződéses határidő), területenként milyen létszám áll rendelkezésre (szakmánként) és milyen a létesítmények p tása.

Prioritás alatt itt nem abszolut sorrend értendő, hanem egy pozícióval jelölik, hogy illető létesítmény

– lakás
– kormánykiemelt
– maximum 2 hetet csúszhat
– maximum 4 hetet csúszhat
– maximum negyedévet csúszhat
– éves programban meg kell kezdeni
– el is maradhat.

A számítógép ezeknek az adatoknak birtokában a létszám limit és a prioritás alapján optimalizálást végez és optimális programot készít.

Heti bontásban tartalmazza a termelési értéket (alapösszegű, bruttó saját és generál termelésről külön), létszámot szakmánként, gépeket gépfajtánként, negyedévenként az anyagszükségletet anyagféleségenként. Amennyiben valamely létesítmény elkészültét a program a vállalt szerződési határidőnél későbbre hozza, vállalatvezetői döntést kell hozni arra vonatkozóan, hogy szerződés módosítás kezdeményezésére vagy valamilyen erőszakos átcsoportosításra kerüljön sor.

Több év tapasztalati adatai alapján képes lesz a rendszer egy olyan adatbank-szerű halmazt előállítani, amelynek alapján már a vállalkozás idejében lehetőség nyílik a pontosabb befejezési határidőre való szerződés megkötésére.

A létesítményenkénti táblák összesítése is elkészül természetesen főmérnökségi és vállalati szinten is, így ezeken a szinteken is rendelkezésre állnak a termelési értékre, létszámra, gépekre, anyagokra vonatkozó heti, negyedévi, évi adatok.

Az elmondott adatok birtokában számítógéppel képesek vagyunk a létesítményre vonatkozó, tervezett, normalizált költségszinteket, eredményeket is számítani. Közvetlenül kiszámítható ugyanis a közvetlen anyagköltség. Ebből – több évi tapasztalat alapján kiszámított kulcs segítségével – elsődleges, az ismertetett módon másodlagos fuvarköltség számítható.

A programozás során kijött közvetlen munkabéreket – ugyancsak több évi tapasztalati szám alapján – felpótlékoljuk béralap szintre. Itt számításba vesszük a közvetett béreket, a különféle kieső időkre eső bérhányadot, még a törtnapi időveszteség okozta többletet is.

A béralap szintű összegből számíttatható a bérek közterhe, a szociális költség és az építéshelyi általános költség is.

A felsoroltak összege az összes költség, s ezt a termelési értékkel összevetve eredmények is számíthatók.

A normatívák állandó finomításával és az említett pótlékkulcsok pontosításával eljutottunk odáig, hogy az építéshelyi operatív tervek ezeknek a tábláknak alapján kerülnek kiadásra, így összhangot sikerült teremtenünk a tervezési rendszer és a termelésprogramozás között.

Az anyag- és költségelszámoltatás számára is fontos információkat nyújt az ismertetett termelésprogramozási rendszer.

Minthogy az elszámolás alá vont valamennyi anyagfajta szerepel a VKN anyagfelhasználási törzsadattárában, az anyagelszámoltatási modul működtetése nem különösen problematikus.

Igen sok vita alakult azonban ki a költségelszámoltatást illetően. A vélemények többsége szerint költségelszámoltatást csak ÉKN alapon lehet végezni. Nehezen, de sikerült meggyőzni egy maroknyi kisebbségnek – igaz ennek az igazgató is tagja volt – az ellenzőket arról, hogy amennyiben a VKN valóban a vállalati sajátosságokat tükrözi, akkor sokkal reálisabb, úgy is mondhatjuk erkölcsösebb ennek alapján elszámoltatni. Ha ugyanis elismerjük, hogy adottságaink indokolnak túllépést az ÉKN-nel szemben, felesleges zaklatás ezeknek a tételeknek az indokoltatása. Az is természetes viszont, hogy ha feltételeink kisebb költségszintet tesznek lehetővé, ennek kiaknázását is elvárjuk munkahelyi vezetőinktől.

Természetesen a költségelszámoltatás alapját képező normatív táblák nem fillér-pontosságúak. A normatívák készítése során figyelembe vett erőforrás szükségletek nem törekednek

valamilyen kigrammozására ezeknek. Ismertettem az egyes erőforrások esetében eltűrt pontatlanságokat.

Tapasztalatunk alapján a számítógéppel kiszámított értékek vállalati, de még főmérnökségi szinten is csaknem teljesen pontosak, a létesítmények egyrészénél azonban vannak benne eltérések. Az eltérések a nagy számok törvénye alapján a magasabb összegfokozatoknál kiegyenlítődnek.

Az így nyert adatok tehát úgy alkalmazhatók az elszámoltatáshoz, hogy ahol egy tűrési határon túl van valamely költségnem eltérése a tényszámoktól, ott a termelőszervezettel szembeni felépés előtt kézi módszerekkel meg kell vizsgálni, hogy a differencia nem adódik-e a VKN adta pontatlanságokból. Ugyancsak tapasztalatból mondhatom, hogy ahol az eltérés a VKN-ből adódik, ott kiugróan nagy differencia jön ki, így könnyen ki is deríthető a hiba. Ilyen ugyanis akkor fordul elő, ha az illető létesítményen nagyon sok a VKN-ben összevont szereplő, átlagolt tétel, vagyis nagy a speciális, egyedi munkák aránya.

Összefoglalóan megállapítható, hogy mintegy 3 évi munkával megteremthető egy építőipari vállalat termelésprogramozási rendszere úgy, hogy az ugyan elsődlegesen programozás-, tehát termelésirányítás centrikus, de igen komoly segítséget képes nyújtani a vállalat,

- munkaerő- és bérgazdálkodási,
- anyaggazdálkodási,
- gépgazdálkodási,
- tervezési és
- elszámoltatási

alrendszerei számára.

Ennek a rendszernek alapvető feltétele egy jó vállalati normatívarendszer kidolgozása. Megítélésünk szerint ideálisan csak saját számítógéppel működtethető. Ez a számítógép azonban még akkor sem kell, hogy túllépje a mini kategóriát, ha azzal ezen kívül még igen sok vállalati alrendszert segíteni tervezzük.

AZ EGÉSZSÉGI ÁLLAPOT ORSZÁGOS FELMÉRÉSÉNEK SZÁMÍTÁSTECHNIKAI MEGOLDÁSA

Molnár László – Széphalmi Géza – Pauka Tibor
KSH ÁSZSZ ÉS ORVOSTOVÁBBKÉPZŐ INTÉZET

Tanulmányunkban a Komplex Országos Morbiditási Vizsgálattal (KOMOV), illetőleg annak számítástechnikai megoldásával foglalkozunk. A KOMOV alapjaiban különbözik a hasonló elnevezésű, jól ismert Kórházi Morbiditási Vizsgálattól. Ez utóbbi egy országos felmérés, melyben az egyes évek során a fekvőbeteg intézményekben megjelentekről egy-egy adatlapot töltenek ki és az összegyűlt adatok évente kerülnek feldolgozásra. Ezzel szemben a KOMOV nem tekinthető nyilvántartási rendszernek, mert elsődleges célja adatok szolgáltatása az egészségügy következő ötéves tervének megalapozásához, országos egészségügyi „pillanatkép" készítése. A vizsgálat az Orvostovábbképző Intézet, valamint a szegedi, a pécsi és debreceni orvostudományi egyetem Szervezési Intézete által gondozott négy különböző mintában összesen mintegy 50 ezer embert fog össze. A felmérés 1978-tól 1979 első negyedévéig tartott, pontosan egy éven keresztül.

A feladat megoldásához a hagyományos feldolgozással szemben adatbáziskezelő rendszer alkalmazása mellett döntöttünk. Erre a célra az ÁSZSZ Honeywell–Bull számítógépének IDS–I rendszere kínálkozott, mint a rendelkezésünkre álló – a tervezés időpontjában – egyedüli lehetőség. Az adatbázis struktúrája, illetve az adatbáziskezelő rendszer által nyújtott lehetőségeken túl meg kívántuk teremteni a gyors szeriális, ill. szekvenciális feldolgozások lehetőségét is, hogy ezáltal megnöveljük a rendszer hatékonyságát.

Tervezéskor a feladat jellegzetességeit számítástechnikai szempontból a következőkben állapítottuk meg:

1. Az adatfelvétel nagyon kiterjedt, mintegy 10–12 kérdőívtípust használ. Szerepelnek közöttük demográfiai adatok (foglalkozási, iskolai végzettség, ...), járó- és fekvőbeteg ellátási (igénybevételi) adatok, követéses vizsgálati adatok (rejtett morbiditás feltárására), retrospektív egészségügyi adatok (három évre visszamenőleg), szűrővizsgálati (kontroll) adatok, stb. A felsorolásból is kitűnik, hogy az egyes bizonylattípusok között tartalmai kapcsolatok vannak.

2. A kérdőíveken a felmérés során várhatóan mintegy 150 millió karakter információ gyűlik össze.

3. A felmérési időszak után rendkívül széles skálájú, több szintű, az első fázisban többnyire statisztikai jellegű igénylistákat kell kielégítenünk, ugyanakkor fel kell készülnünk ad-hoc kérdések megválaszolására is. Hosszabb távon igény van az adatok interaktív – és akár egyidőben több felhasználó által történő – lekérdezésére is.

4. Kapacitáshiány miatt az adatok rögzítését csak alvállalkozóval végeztethetjük. Ott történhet meg részben az adatellenőrzés és javítás is, de csak „kérdőíven belüli" ellenőrzésekre kerülhet sor.

A felsorolt szempontok alapján mérlegeltük, hogy a feladatot hogyan oldjuk meg a HwB számítógépen. A kiválasztott IDS–I a CODASYL típusú hálózatos adatbáziskezelő rendszerek előfutára, amelyre ha „ráültetjük" a MDQS interaktív lekérdező rendszert is, akkor lényegében egy „modern" adatbáziskezelő rendszert nyerünk. Választásunkat az alábbi érvek indokolják:

– az adatbázis struktúrája jó lehetőségeket biztosít a kérdőívek közötti kapcsolatok ellenőrzésére, tehát a teljes adatállomány konzisztenciájának biztosítására;

– lehetővé válik az adatok interaktív „konkurrens" lekérdezése

Kialakítandó adatbázisunk külön jellegzetessége a statikusság, abban az értelemben, a javítások befejezése után a meglevő adatok már nem módosulnak és új adat sem érkezik.

Választásunk ellen legsúlyosabb érvként azt lehetne felhozni, hogy általános, statiszti jellegű feldolgozások futtatása közvetlenül adatbázisból vett adatok alapján nagyon időigén lehet. Az adatbázis logikai struktúrájának viszonylagos „merevsége" miatt különösen igaz akkor, ha sok különféle igényt kell kielégíteni. A feldolgozások menete sematikusan a kö kező: bizonyos adatmezők tartalma alapján „csatolt" adatokat keresünk vissza, ezekkel va milyen műveletet végzünk, majd az eredményt „láthatóvá" tesszük. Az egyik feladat teh az, hogy lehetővé tegyük az adatbázisban szereplő rekordok minél sokrétűbb osztályozás Elképzelésünk szerint olyan rendszert alakítunk ki, amely ehhez az adatbázis struktúráján kihasználása mellett – kiaknázva az adathalmaz statikus jellegét – szükség esetén a nem, vagy esetleg részben rendezett mintán történő szeriális ill. szekvenciális feldolgozási módo is biztosítja. Ettől a kompromisszumtól a hatékonyság jelentős növekedését várjuk.

A fenti elv megvalósítására a következő lehetőségek állnak rendelkezésünkre:

1. Az adatbázis struktúrája által nyújtott lehetőség: bizonyos mezők tartalma alapjá rekordokat egymáshoz láncolva tároljuk. Mivel ezt az adatbázis töltésekor kell végrehajtan ez a csoportosítás „végleges". Ez a megoldás azért sem rugalmas, mert az IDS-ben a lánc szerinti olvasás nem paraméterezhető. Mindebből következik, hogy ezt a lehetőséget csak előre ismert legfontosabb felosztásokra célszerű alkalmazni.

2. Rugalmasabb (részben paraméterezhető) megoldást kínál, ha lehetővé tesszük az a bázis bizonyos részeinek gyors, szeriális végigolvasását. A kiolvasott rekordokat osztályozz és a folyamat lényegében egy táblázat kitöltésével folytatódik. Akkor gyors a feldolgozás, ez a táblázat a memóriában elfér. Ennek érdekében a feladat volumenét többszöri végigol sással csökkenteni lehet, szükség esetén. (Kb. ötszöri olvasással a futási idő még várhatóan nem lesz hosszabb, mint az adatok kigyűjtése, átrendezése és szekvenciális feldolgozása es tén).

Az *adatbázis létrehozásának lépései* a következők:

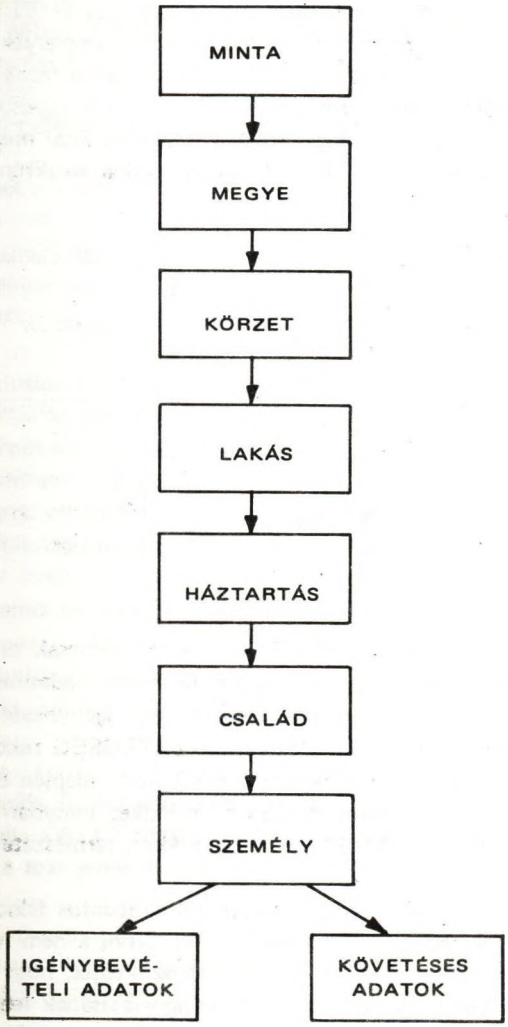
0., Szaktudományos modell kialakítása
1., Szerkezeti modell kialakítása
2., IDS „logikai" szintű javaslat-modell kialakítása
3., IDS „fizikai" szintű javaslat-modell kialakítása
4., Fizikai megvalósítás (Az első tapasztalatok legkorábban 1979 végére várhatók.)

0. lépés

Orvos-egészségügyi szaktudományos modell kialakítása, azaz a feladat kitűzése a vé használó által. Itt csak annyit jegyzünk meg, hogy az egyes minták jellemzőit sikerült úg egységesíteni, hogy a megvalósításra kerülő struktúra is egységes lehet.

1. lépés

Az adatok szerkezeti modelljének kialakítása. A szaktudományos modell elemzéséb feladatot a számítástechnikai feldolgozás számára újra megfogalmaztuk. Eközben Codd re ciós adatmodellezésének eszközeire támaszkodtunk. Az ún. adatbáziskezelési anomáliák el lése végett modellünket normalizáltuk olymódon, hogy a kérdőívek összességén szereplő mennyi adatot (adattípust) összegyűjtöttük és a normalizáló eljárás segítségével alakítottu belőlük a relációkat. Igy egy logikai adatszerkezetet kaptunk, amelynek alapján az adatb kezelő rendszer (IDS–I) láncolási szabályait is figyelembe véve javaslatot tehettünk a kif tendő adatbázis („logikai" felépítésére. Ez – némileg leegyszerűsítve – a következő volt



Jelmagyarázat:

——►: IDS lánc-kapcsolat rekordok között

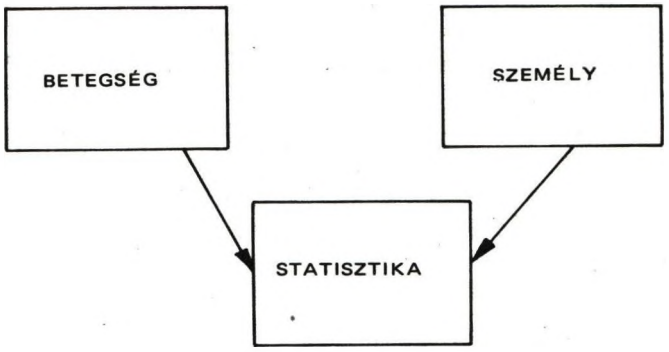
1. ábra A KOMOV adatbázis kiinduló logikai struktúrája

A felmérés során egy igénybevételhez és egy követéses vizsgálathoz is több betegségkód (BNO) tartozhat. A normalizálás során itt az ún. ismétlődő csoport kiküszöböléséhez azt a megoldást választottuk, amit a kérdőívek szerkezete is sugall, azaz minden rekordban fix (maximális) számú BNO kód kapott helyet.

2. lépés

A következőkben tovább módosítottuk az adatbázis logikai felépítését:
Mivel a feldolgozások középpontjában a személyek helyett néhol a BNO-k állnak, ezért szük-

ség van az igénybevételek és a követéses adatok betegségkód szerinti osztályozására. A hely szükséglet csökkentése és egyéb megfontolások alapján ezt a SZEMÉLY rekordok szintjén lósítottuk meg, azaz azt kívánjuk regisztrálni, hogy az egyes személyek a vizsgálat során me BNO-kkal „kerültek kapcsolatba". Mivel ez a reláció N:M típusú (azaz egy igénybevételhez ill. követéses vizsgálathoz több BNO tartozhat és viszont) ezt a kapcsolatot csak ún. „konn tor" rekordtípus beiktatásával lehetett megvalósítani. (Az IDS által megvalósítható struktúra nem komplex, hanem egyszerű hálózat.) A módosított logikai struktúrát az említett részre leszűkítve az alábbi ábra szemlélteti:



A konnektor funkcióban levő STATISZTIKA rekord nemcsak azt mutatja, hogy egy adott személy egy adott betegséggel kapcsolatba került, hanem adatmezői alapján egyéb in formációkat is szolgáltat. (Az adott személy követéses vagy igénybevételi rekordjaiban szer pel ez a BNO ill. hol jelentkezett időben először.) A BETEGSÉG rekordot olyannak kell finiálni, hogy az gyorsan visszakereshető legyen a BNO kód alapján és biztosítani kell az hogy a BETEGSÉG–SZEMÉLY „közvetett láncon" mindkét irányban gyorsan tudjon mo ni. (A KÖVETÉSES és az IGÉNYBEVÉTELI rekordokban természetesen megtartottuk a BNO-kat.)

A tanulmány elején említettük, hogy a rendkívül változatos feldolgozási igények mia a láncolással történő csoportosításon kívül lehetővé kell tenni a nem- vagy részben rendez mintán alapuló feldolgozásokat is. Mindkét esetben az szükséges, hogy megteremtsük az i adatok *rendkívül gyors,* szeriális (ill. szekvenciális) végigolvasásának lehetőségét.

A nem rendezett szintén történő gyors, szeriális feldolgozás szükségessége az alábbi vetkezményekkel jár:

1. az igénybevételi és követéses rekordok adattartalmát ki kell egészíteni olyan adat amelyekre a rájuk vonatkozó feldolgozásokban gyakran van szükség. Gondosan kell mérle hogy melyek legyenek ezek, mert a túlságosan nagymérvű redundancia már nem gazdasá

2. az említett rekordok számára egy külön területet kell kijelölni (PAGE–RANGE), amit az IDS gyorsan végig tud szekvenciálisan olvasni (RETRIEVE SERIAL ige segítségé

A részben rendezett mintán történő feldolgozást a következő elv alapján tehetjük le vé (az alábbiakban kihasználjuk az adatbázis *statikus* jellegét!):

Minden – igénybevételre vonatkozó – kérdés megválaszolásához egyszerűen eldönth hogy a szükséges input adatokra milyen adatmezők szerinti rendezettségben lenne szüks az is, hogy a rendezettség mely hierarchiái lennének megfelelőek – a kitöltendő táblázat retétől függően.

Jelöljük A-val azon mezők halmazát, amelyek szerinti rendezésre szükség van, vagy várha szükség lehet. Ezek közül kiválaszthatjuk azokat, amelyekre többször is szükség lehet; je ezt B-vel. Nyilván B⊂A.

zokat a rendezési mezőket is kiválaszthatjuk, amelyek az igénybevételi adatokat sok osztályba sorolják; ezek halmazát jelöljük C-vel. C⊂A teljesül, de C és B viszonyát nem ismerjük előre. Megoldásunkkal lehetővé tesszük az igénybevételeknek C elemei szerinti *tetszőleges* rendezettségét, ami szükség esetén a legtöbbször olyan előrendezettségként használható, amely megoldja tárigény gondjainkat.

Az eljárás alapja a következő:

1. lépés:

Az input igénybevételi rekordokat ki kell egészíteni C elemeivel és ki lehet egészíteni B minden elemével, így a rekordokat B∪C szerint olyan *fizikai* rendezettségben tároljuk a korábban említett PAGE–RANGE-ben, amit szintén lehet majd a feldolgozásoknál hasznosítani. Természetesen az adatbázisban nem kell B∪C minden elemét tárolni, a rendezettség a fontos. Lényeges az, hogy a *rendezettség „legmagasabb szempontjai" C elemei legyenek.*

2. lépés:

Ezt a rendezett adathalmazt C alapján részhalmazokra lehet osztani, és betöltés után isszerni fogjuk minden részhalmaz első és utolsó rekordjának címét. (A két cím közötti rekordok C szerint azonos részhalmazba tartoznak.) Az adatbázisban létrehozunk olyan (OSZTÁLYOZÓ) rekordokat, amelyek C elemeinek egy adott érték n-eséhez megadja a hozzátartozó részhalmaz tartományát, amelyet címtől címig szekvenciálisan végig lehet olvasni a RETRIEVE SERIAL utasítás segítségével. (Ennek az utasításnak megvan az a nagy előnye, hogy végrehajtása bármikor megszakítható, pl. hiányzó – és más rekordban megtalálható – információk megkeresése miatt és utána tovább folytatható az eljárás.)

legyen $C = \{C_i\}$ $i = 1, \ldots, n$ és

$C_i = \{C_{ij}\}_i$ ahol C_{ij} pozitív egész szám

(tehát C_i-k a rendezési mezők, C_{ij}-k pedig a lehetséges értékek.)

A létrehozandó rekordok „CALC" osztályúak, amelyeknek randomizáló kulcsa C minden eleme, adattartalma pedig a már említett tartomány kezdő- és végcíme, tehát ha egy ilyen rekord például

$C_{1j} = k_1, \ldots, C_{nj} = k_n$	x, y

akkor minden olyan igénybevételi rekord címe x és y között van, amelyre $C_{1j}=k_1, \ldots, C_{nj}=k_n$ más „kulcsú" rekord nincs ebben a tartományban.

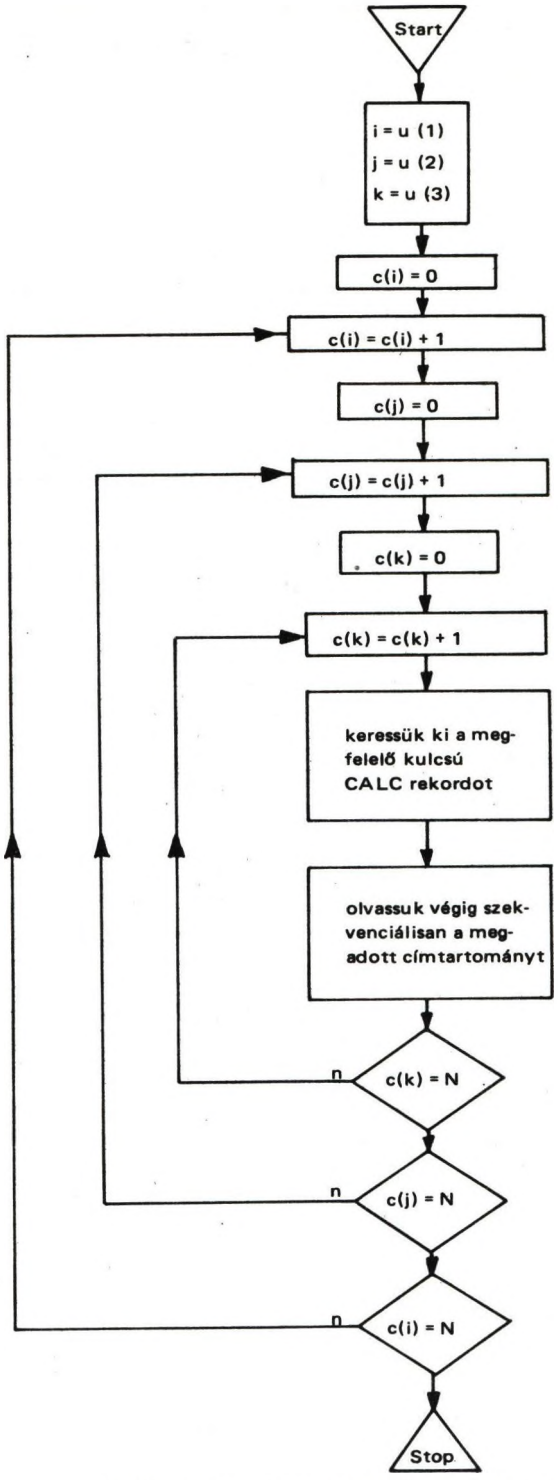
3. lépés:

Ezzel a módszerrel tetszőleges olyan rendezettséget lehet „szimulálni", amelynél a rendezési mezők C-beliek. Ehhez még annak kell teljesülni, hogy minden C_i lehetséges értékei folyamatosan kitöltött intervallumot alkossanak. Ez megfelelő átkódolással elérhető. Az általánosság csorbítása nélkül feltételezhetjük, hogy

$$1 \leqslant C_{ij} \leqslant Ni=1, \ldots n.$$

Legyen n=3 és kérjük az igénybevételi adatokat $C_2/C_3/C1$ mezők szerinti rendezettségben:

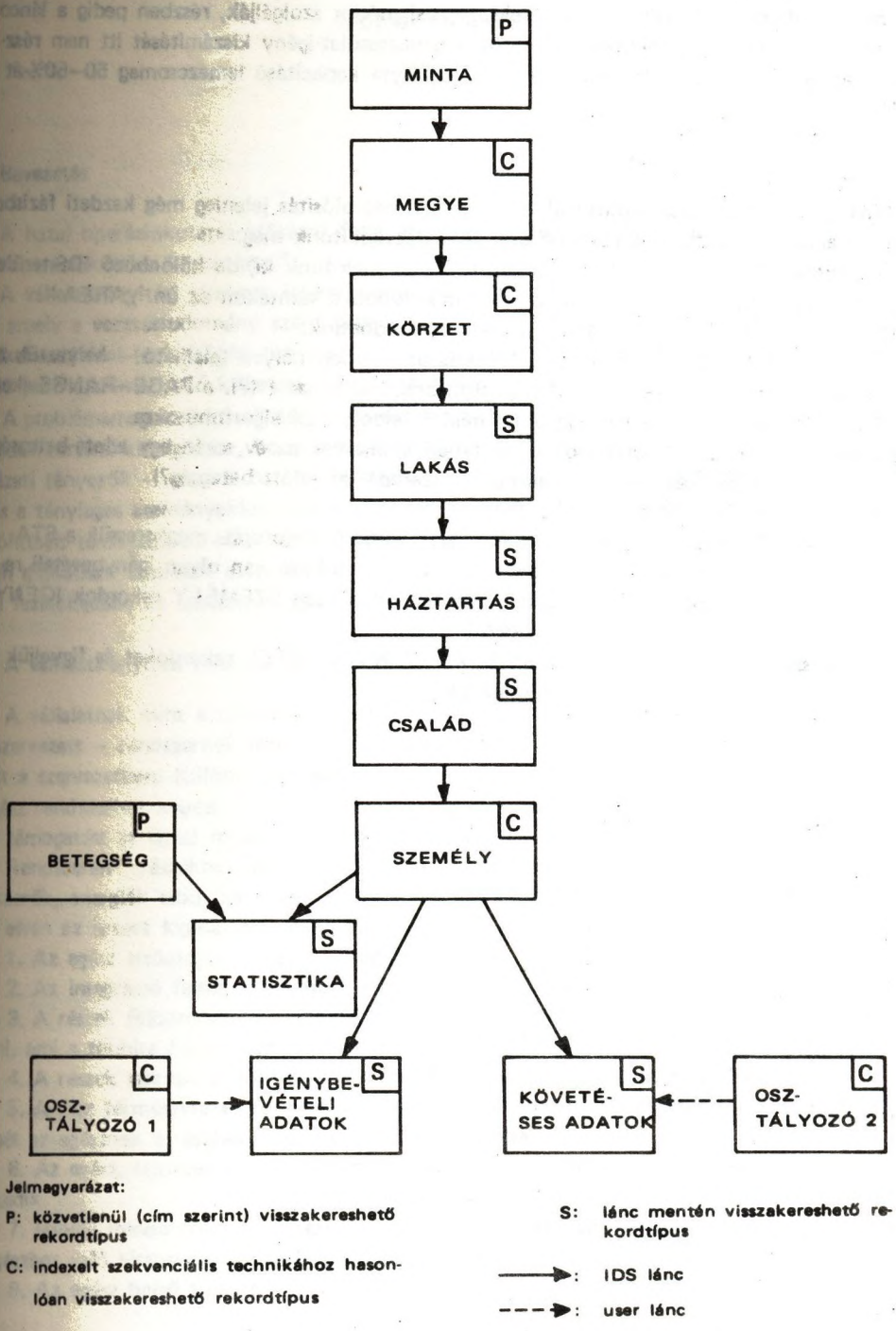
Legyenek a feldolgozás paraméterei: esetünkben u/1/=2, u/2/2=3, u/3/=1 – amelyek a rendezés hierarchiáját definiálják, ekkor az eljárást a 2. ábra mutatja.



2. ábra Paraméterezett rendezettségre épített eljárás sémája

Ebben az eljárásban kihasználhatjuk azt is, hogy minden ilyen tartományon belül C szerint zettek a rekordok.

A 2. lépésben elmondottak alapján adatbázisunk logikai struktúrája – az 1. ábrán bemutatotthoz képest – a következőképpen módosul: lásd a 3. ábrát.



3. ábra A KOMOV adatbázis módosított logikai struktúrája

3. lépés

A fizikai szintű tervezésből csak két fontos elemet emelünk ki. Itt kellett megtervez-nünk az előző pontban említett PAGE–RANGE-eket (CLUSTER-eket).

Ezek részben a már leírt szeriális feldolgozási módot szolgálják, részben pedig a lánco-kon való „navigálás" meggyorsítását. A pontos lemezterület-igény kiszámítását itt nem rész-letezzük, az egyes minták átlagosan egy 100 megabyte kapacitású lemezcsomag 50–50%-át foglalják le.

1 *4. lépés*

Mivel – mint azt már említettük – a fizikai megvalósítás jelenleg még kezdeti fázisb-tart, ezért ebben a pontban is csak néhány részletet említünk meg:

Az egyes minták adataiból azonos struktúrákat alakítunk ki, de különböző IDS-terüle-ken (AREA-kon) így az azonos feldolgozásokat különböző mintákon az ún. „AREA–SWITCHING" technika segítségével egyszerűen megoldhatjuk.

A MINTA rekordban – mely az adatbázisban kiemelt helyen található – helyeztük el az adott mintához tartozó rész-adatbázis legfontosabb jellemzőit (Pl. a PAGE–RANGE-eket).

Végezetül nézzünk meg egy egyszerű példát feldolgozási algoritmusokra.

Kérdés: Hányszor fordultak orvoshoz a mintabeli személyek az év során egy adott betegség miatt? (Hány igénybevételi rekordban szerepel az adott betegség?)

Lehetséges válaszadó algoritmusok:

1. lánc szerint: kikeressük az adott betegség BETEGSÉG rekordját, megkeressük a STA-TISZTIKA rekordok segítségével, hogy kiknek van olyan igénybevételi re-kordjuk, ahol ez a BNO szerepel, majd ezen SZEMÉLY rekordok IGÉNY-BEVÉTEL rekordjait nézzük végig.
2. nem lánc szerint: végigolvassuk szeriálisan az IGÉNYBEVÉTEL rekordokat és figyeljük (szeriálisan) adott betegség előfordulásának gyakoriságát.

GÉPIPARI VÁLLALAT INFORMATIKA RENDSZER TERVEZÉS MODELL SZINTŰ MEGKÖZELÍTÉSE

Molnár István
KG ISZSZI

Bevezetés

A hazai operációkutatás időszerű feladatai között kiemelkedő helyet foglal el a szerve- s és operációkutatás kapcsolata.*

A vállalatirányítási rendszer fejlesztésének egyik megközelítése az operációkutatás és szer- zés, amely a vezetéstudomány szemlélete és módszerei alapján a korszerű informatikai eszkö- k alkalmazásával (számítógép tudomány) az irányítási rendszernek a környezeti változásokhoz rténő jobb alkalmazkodó képesség kifejlesztését segíti elő.

A problémamegoldás közelítési módja az irányított rendszer absztrakt analitikus modelljének lymódon történő kidolgozása, hogy a működést befolyásoló véletlenszerű események — mint ockázati tényezők — ésszerű határok között tervezhetők, a várható események előre jelezhe- k és a tényleges eseményekkel összemérhetők legyenek. Az összemérhetőség alapján a veze- s döntéseit támogatja az irányítási rendszer modellje. A valóságos fizikai rendszert modell zinten digitálisan szimulált software rendszerben képezzük le, amely a termeléstervezés és irá- yítás funkciójában az analitikus modellt támogatja.

A vállalatirányítási rendszer struktúrája

A vállalatnak mint a társadalmi környezetben működő — intézményesített, tehát cél sze- nt szervezett — rendszernek legjobb közelítési módja, ha leírjuk az események áramlási folya- atait a szervezetben. Külön-külön elemezzük a részeseményeket és kiderítjük ezek viszonyát z egész rendszerhez képest. Ilymódon megállapítható, hogy mely részek nem nyújtanak maxi- nális támogatást az egész működése szempontjából.

Rendszeren részeknek vagy elemeknek (az adott esetben termelőeszközök, anyagok, unkaerők, energiák stb.) bármilyen egésszé történő rendezését értjük. A rendezés az integ- áció elvén az ismert logikai axiomákon alapul:

1. Az egész elsődleges, a részek másodlagosak.
2. Az integráció feltétele a részek kölcsönös összefüggései egymással az egészen belül.
3. A részek felbonthatatlan egészet alkotnak, amelyben egyetlen résszel sem történhet emmi, ami a többire hatást nem gyakorol.
4. A részek szerepe annak a célnak van alárendelve, melyet az egész szolgál.
5. A rész természete és funkciója az egészben elfoglalt helyzetéből következik és visel- edését az egésznek a részhez való viszonya szabályozza.
6. Az egész, tekintet nélkül céljára és bonyolultsági fokára egyetlen egységes egészként iselkedik.
7. Mindig mindennek az egésszel mint feltételezéssel kell kezdődni, s a részeknek mint z egészhez való viszonyokkal ebből kell kifejlődnie.
8. Az egész belső tulajdonságait a környezethez való viszonya határozza meg.

---

* Vezetéstudomány X. évf. 1979/1. Trethon Ferenc: A hazai operációkutatás időszerű feladatai

Az integráció elve szerint a vállalati struktúrát az alábbiak szerint határozhatjuk m

A. Információs alrendszer
mely a rendeltetés szerinti objektív anyagi folyamatok eseményeire vonatkozó formációk rendszere

B. Szervezeti alrendszer
mely az információk célszerű áramlási kapcsolatait biztosítja a kommunikáció resztül

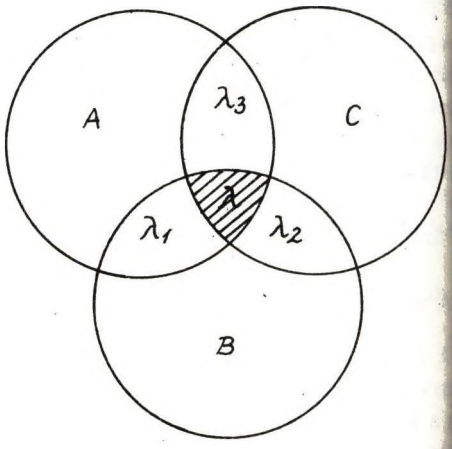
C. Az információ feldolgozó (műszaki) alrendszer
mely a vezetés irányító tevékenységének döntését támogató rendszere, és azt képességet nyújtja, hogy a környezeti változásokhoz történő alkalmazkodást a nimális időn belül optimálisan tegye meg a rendszer objektív A. eleme és szu tív B. eleme összehangolására.

E struktúrákat az 1. ábra szerint vizsgáljuk.

Az irányítási rendszer strukturális felépítésének fogalmi és kapcsolati rendszerét tapasztalati úton határoztuk meg. Kérdés, hogy e tapasztalatot helyesen értelmezzük-e és hogy az egyes alrendszerek logikailag milyen viszonyban vannak egymással az 1. sz. ábra alapján.

Az elemzést a fogalmak azonossági jellemzői szerint vizsgáljuk:

Egy A jelű fogalom B jelű fogalommal azonos abban az esetben, ha a két fogalom terjedelme és tartalma azonos. Ezt az állapotot a „tartalmazza" fogalommal fejezzük ki és a következő szimbólummal jelöljük „$\subset$".



A = Információs alrendszer
B = Szervezeti alrendszer
C = Információfeldolgozó (műszaki alr

1. ábra
A vállalat strukturális felépítése

Az A és B fogalmak kapcsolatát a kijelentés logika szabályai szerint alkalmazott ko gációs jellel „." fejezzük ki.

A „nem tartalmazza" kijelentést a negált szimbólum jel fejezi ki „$\not\subset$".

Az előzőekből következik, hogy összeegyeztethetetlen (diszjunkt) fogalmak között miféle terjedelmi és tartalmi megegyezés nem állhat fenn.

Az 1. sz. ábra szerint megadott fogalmak terjedelmileg és tartalmilag kizárják egym de van mégis közös elemük (osztályuk).

Abban az esetben, ha a megadott fogalmak terjedelemben és tartalomban egymást i tesítenék, akkor egymás alá rendelt (egymást fedő) fogalmak lennének és ebben az esetb zös osztályként viselkednének.

Ebből következik, hogy az A, B és C csak egymás mellé rendelt fogalmak lehetnek (koordinált fogalmak). A létrejött „közös metszetosztály" (jelöljük: λ) szükségképpen A, B, C fölérendelt, koordinációt tartalmazó fogalom.

Jelölése: $[\overline{A \cap B \cap C} = \lambda]$

ahol

„metszetet alkot" kijelentés szimbólikus jele „ $\cap$ " és a *negáció értelmezése:* A, B, C halmazok metszetet alkotnak és λ *nem üres osztály.* Az 1. sz. ábra alapján kijelenthetjük, hogy A nem tartalmazza B-t és B nem tartalmazza C-t és C nem tartalmazza A-t és A, B, C metszetet alkot és nem üres osztály.

Tehát írjuk

$$1.\ [\overline{(A \subset B)} \,.\, \overline{(B \subset C)}.\ \overline{(C \subset A)}.\ \overline{(A \cap B \cap C\ = \lambda)}] = R$$

ahol R a teljes rendszer.

A konjugációs kapcsolatok új metszet osztályt λ-át hoztak létre

1.1 $(\lambda = \overline{A \cap B \cap C}\ )$ tehát

1.2 $R = \lambda \,.\, [\,(\overline{A \subset B}\,) \,.\, (\overline{B \subset C}\,) \,.\, (\overline{C \subset A}\,)\,]$

λ mint irányító funkciót értelmezzük és kell, hogy a döntési és végrehajtási szinteken kielégítse a rendszerben a tervezés és ellenőrzés funkcióját.

A tervezés problémájának, mint a vezetési célokat meghatározó logikának a megközelítése a következő lépésekben történik:

1. Fel kell mérni a környezet jövőbeli alakulását, politikai, gazdasági és konkurrenciális szempontból.

2. Meg kell határozni milyen szerepet szánnak az adott rendszernek ebben a környezetben.

3. Fel kell mérni a szükségleteket és a rendelők kívánságait.

4. Meg kell állapítani milyen értelemben változnak meg az érintett érdekcsoportok — kooperálók, szállítók, munkások stb. — szükségletei és kívánságai.

5. Meg kell teremteni a kommunikáció és információ áramlási rendszert, melynek segítségével a szervezet tagjai résztvehetnek a célok szerinti tervezésben és ellenőrzésben.

6. Átfogó tervet kell készíteni a vállalati célok megvalósítására, mely a szervezet erőfeszítéseit megszabja.

7. A tervet részekre — kutatás, fejlesztés, tervezés, gyártás, üzemeltetés, elszámolás, ellenőrzés stb. — kell bontani a tevékenységek szerint.

8. A résztevékenységeken belül — összhangban az adott rendszer terveinek egészével — ki kell dolgozni az operatív részterveket az erőforrások szabályozott felhasználásával.

Összegezve

a tervezési és koordinációs modellnek ki kell elégíteni:

— a stratégia tervezést (középtávú)
— a taktikai tervezést (rövidtávú)
— az operatív végrehajtás tervezést (munka ütemezés)
— az elszámolást és ellenőrzést.

E feltételek mellett λ-át mint tervezési és irányítási modellt kell meghatározni.

Az 1. ábra alapján a struktúra további vizsgálatával a következő tulajdonságokat állapítjuk meg.

A rendszer „A" alrendszere az információs alrendszer, mely a teljes működő anyagi rendszerben létrejött (mozgásállapotok) eseményekre vonatkozó *objektív* struktúra.

A rendszer „B" alrendszere a kommunikációs (szervezeti) alrendszer, mely az eseményei mozgásának emberi elhatározások alapján létrehozott szabályozott rendszere. E szabályozottság folytán az alrendszer *szubjektív* struktúra.

A rendszer „C" alrendszere az „A" és „B" alrendszereket *integrálja* a rendszertechnika eszközeivel (elméletek, törvények, axiomák, módszerek és műszaki eszközök) azzal a céllal, hogy az „A" alrendszer *objektív folyamatait* és a „B" alrendszer *szubjektív lehetőségeit optimálisan összehangolja.*

E hármas sajátosságot kifejezik a struktúrák azon metszetosztályai, amelyek csak $\overline{A \cap B}$, illetve $\overline{B \cap C}$ illetve $\overline{C \cap A}$ által létrehozottak, de nem elemei λ-nak.

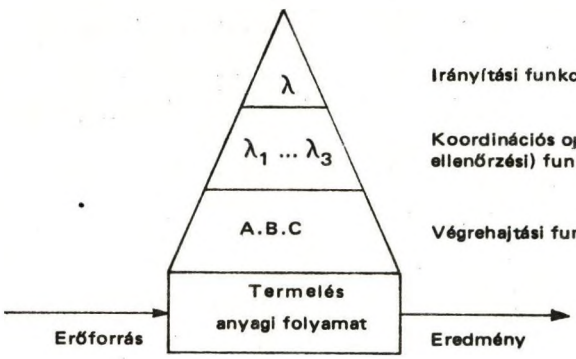
Tehát írhatjuk, hogy

2. $\lambda_1 = (\overline{A \cap B}) \setminus \lambda$
3. $\lambda_2 = (B \cap C) \setminus \lambda$
4. $\lambda_3 = (C \cap A) \setminus \lambda$

ahol

$\lambda_1 \ldots \lambda_3$ koordinációs funkciót tölt be értelemszerűen
λ_1 az információs és szervezeti alrendszer között (folyamatszervezés)
λ_2 a szervezeti és információfeldolgozó alrendszer között (rendszertervezés)
λ_3 az információs és információfeldolgozó rendszer között (rendszerprogramozás)

A megadott λ_1, λ_2, λ_3 tartalmi, fogalmi és terjedelmi tulajdonságai teljesen kizárják egymást. Ezen azt értjük, hogy λ_1, λ_2, λ_3 osztályok egymást nem helyettesítik. Ebből következik hogy egymásnak nem alárendelt, hanem mellérendelt fogalmak. Ebben az esetben értelmezhető λ alárendelt λ_1, λ_2, λ_3 koordinált rendszer hierarchia a 2. ábra szerint.



2. ábra A vállalatirányítás hierarchikus struktúrája

A vizsgálat alapján azt a következtetést is levonhatjuk, hogy a rendszertervezés alapvető követelménye a rendeltetés és cél szerinti munkafolyamatokban és munkahelyeken a munkaszervezés optimális szintre történő emelése. (λ_1)

A rendszer viselkedés jellemzőit elvileg nem érinti, ha csak $\overline{A \cap B}$ halmazok léteznek (klasszikus modell).

Ebben az esetben írhatjuk, hogy

5. $\lambda = (\overline{A \cap B})$ de figyelembevéve
6. $\lambda_1 = (\overline{A \cap B}) \setminus \lambda$ és

5. kifejezést 6-ba helyettesítve kapjuk

7. $\lambda_1 = (\overline{A \cap B}) \setminus (\overline{A \cap B}) = \emptyset$

Ami logikailag azt fejezi ki, hogy (A ∩ B) nem eleme (A ∩ B)-nek, vagyis $\lambda_1 (=0)$ nem értelmezhető, mint koordináció. Abban az esetben tehát, ha csak A és B halmaz létezik λ az irányítás és koordináció elemeit együtt tartalmazza. Ebben az esetben a vezetők személyisége meghatározó. A rendszer viselkedésének szubjektív oldala dominál, a kockázat csökkentés a vezetési magatartáson túl elsősorban szakismeretet követel.

A számítógép alkalmazás bevezetése illetve a számítógép-tudomány eredményei alapján különböző λ_2 struktúrák felépítése válik lehetővé az alkalmazott műszaki rendszer függvényében.

Ezek meghatározók λ_3 tartalmára. Az elemzés azt mutatja, hogy a számítástechnika bevezetésével az irányító és vezető koordinációs funkciók és felelősségek jobban elhatárolódnak mint a klasszikus modell esetén.

Az irányítási modell meghatározása

A vállalatot jellemezzük a következőkkel:

- C_o rendelkezésre álló termelői kapacitás az adott időszakban
- C a rendelkezésre álló termelői kapacitásból ténylegesen kihasznált kapacitás az adott időszakban
- b_o a rendelkezésre álló normatív munkaórák az adott időszakban
- b az adott időszakban ténylegesen felhasznált normatív munkaórák
- A_o a tervezett átlagos teljesítmény (pl. termelési érték (óra)
- A a tényleges átlagos teljesítmény az adott időszakban.

A kapacitáskihasználási tényező:

9. $$\eta_k = \frac{C}{C_o} = \eta_A \cdot \eta_b$$ ahol

10. $$\eta_b = \frac{b}{b_o} =$$ a foglalkoztatás szervezettsége

11. $$\eta_A = \frac{A}{A_o}$$ a hatékonysági teljesítménytényező

A tényleges rendelkezésre álló termékkibocsátó kapacitás tehát:

12, $$C = \eta_b \cdot \eta_A : C_o$$ melyet

a tervezett termékből előállított darabszámmal mérjük. Ebben az esetben legyen P_c a szükségletet leíró célfüggvény egyenlő a termelési célfüggvénnyel illetve a tényleges rendelkezésre álló kapacitással, tehát

13. $$P_c = \eta_b \cdot \eta_A \cdot C_o$$

A vezetés egyik célja, hogy $\eta_b \cdot \eta_A \Rightarrow 1$ mely a szervezettséget komplexen fejezi ki. η_b és η_A tartalmának pontosabb meghatározása vállalat-, szakágazat- és politikai függő.

A termékkibocsátás illesztését a szükséglethez a következő paraméter változtatások mellett biztosíthatjuk:

– a foglalkoztatottság fokának változtatása állandó teljesítmény tényező mellett. (Extenzív fejlesztés.)

– A teljesítmény tényező változtatása állandó foglalkoztatási tényező mellett. (Intenzív fejlesztés.)

– C_o kapacitás változtatása (gyártókapacitás bővítés vagy csökkentés).

Tételezzük fel, hogy P_c meghatározható mint a vállalat által gyártott termékek értékesítési terve egy adott értékesítési időszakban. Legyen $t_o - t_1$ az értékesítés időszaka, ekkor írhatjuk, hogy

$$14.\quad P_C = \int_{t_o}^{t_1} \sum_{n=1}^{i} A_i \cdot N_i \, dt$$

Ahol n=1 ... i a termékválaszték melyből „N" db értékesítését tervezik „A" kereskedelmi á
A szükségleti célfüggvénynek P_C egyenlőnek kell lenni a termelési feltételek célfüggvényére.
Ezért írhatjuk t_o–t_1 értékesítési és t_2–t_3 termelési időszakokra hogy:

$$15.\quad \int_{t_o}^{t_1} \sum_{n=1}^{i} A_i \, N_i \, dt = \eta_b \, \eta_A \int_{t_o}^{t_1} \sum_{n=1}^{i} C_{oi} \, K_i \, dt + \int_{t_o}^{t_1} \sum_{n=1}^{i} H_i \, dt$$

Ebben a kifejezésben C_{oi} az i-dik termék gyártásához rendelkezésre álló termelési kap
tás az i-dik termékből tervezett gyártás darabszámokban kifejezve, K_i az i-dik termék előál
sának teljes költsége és H_i az i-dik termék előállításával létrejött többletérték, mely a válla
nyereség képzést adja a mindenkor érvényes szabályozó feltételek mellett.

K_i két részből áll.

A közvetlen termelési célok érvényesítését és az ehhez szükséges, a termelési program létre
zását illetve módosítását biztosító K_v termelésváltozás költségei. (Termelésváltozási költség
azok a munkaráfordítást kifejező költségek, melyek a termelési program módosítását, megv
toztatását teszik lehetővé.)

Tehát:

$$16.\quad K_i = K_k + K_v$$

A közvetlen termelési költségek K_k tartalmazzák:

– a normatív órákban kifejezett munkaórát, melyet az i-dik termék előállítására ford
tottunk,

– a normatív értékekben kifejezett közvetlen anyagokat, alkatrészeket, melyeket az i
termék előállítására felhasználtunk.

– a normatív értékekben kifejezett amortizációs költségeket, melyeket az i-dik termé
előállítása során a szerszámokból és gépekből felhasználtunk és az előállított termékbe men
át,

– a normatív értékekben kifejezett energiákat, vagy energiahordozókat, melyeket az i
termék előállítására felhasználtunk,

– ezek közterheit.

Ezen értékek meghatározása a műszaki fejlesztés időszakában az ártervezés feladatai k
zött történik. Az ártervezés mint mérnök-közgazdasági funkció a sorozatnagyság, megbízha
és technológia (automatizálás) összefüggésében új vonása (cost engeneering) a korszerű terv
rendszereknek.

A termékváltozás költségei alatt K_v a következőket célszerű elsődlegesen figyelembe
– kereskedelmi és szervíz költségek
– kutatás-fejlesztési költségek
– gazdálkodási szervek költségei
– gyártás általános költségei
– üzemfenntartás költségei
– szociális költségek
– készletgazdálkodás költségei.

Egyes esetekben a kutatás-fejlesztési költségeket célszerűbb a K_k költségeknek az ártervezés során figyelembevenni. A készletgazdálkodás K_g költségei K_{g1} állandó és K_{g2} változó költségekre oszlanak.

A K_{g1} állandó költségek

- raktári épületek értékcsökkenése
- raktári épületek és raktár kiszolgáló eszközök, járművek üzemeltetési költségei
- raktározással és szállítással kapcsolatos bérköltségek és azok közterhei.

A K_{g2} változó költségek

- a készletezett anyagok, félkésztermékek és késztermékek értéke után felmerülő eszközlekötési adóterhek, kamatok, járulékok
- biztosítási díjak
- minőségromlás miatt bekövetkezett értékcsökkenés, leértékelés
- különleges kezelési és tárolási költségek
- elfekvő készletté válásból előálló veszteség.

Az irányítási rendszer absztrakt analitikai modellje

Az előző összefüggések alapján az analitikai modell mint a rendszer viselkedését jellemző összefüggés meghatározható.

Ezen makroszintű összefüggés keretein belül a vezetői célok szerint meghatározott adatok illetve célfüggvények fejezik ki a vezetés célját.

Az analitikai modell a rendszer viselkedésére jellemző. A viselkedés kezdeti feltételeit a modell egyes elemeiként kidolgozott célfüggvények tényleges vagy tervezett adatai adják meg.

Tehát írhatjuk

$$17.\ \int_{t_o}^{t_1} \sum_{n=1}^{i} A_i N_i\, dt = \eta_b \cdot \eta_A \int_{t_2}^{t_3} \sum_{n=1}^{i} C_{oi}\,(K_i + K_{vi} + K_{g1}{}^{i} + K_{g2i})dt + \int_{t_o}^{t_1} \sum_{n=1}^{i} H_i dt$$

Ebben a formájában a modell gyakorlatilag nem kezelhető, mert a modell minden tagja nem írható fel mint az idő függvénye.

A gyakorlati követelményeknek megfelelően írjuk:

$$18.\ \sum_{t=t_o}^{t_1} \sum_{n=1}^{i} A_i N_i = \eta_b \cdot \eta_A \sum_{t=t_2}^{t_3} \sum_{n=1}^{i} C_{oi}\,(K_i + K_{vi} + K_{g1i} + K_{g2i}) + \sum_{t=t_o}^{t_1} \sum_{n=1}^{i} H_i$$

ahol $t_1 - t_o = t_e$ értékesítési időperiódus és

$t_3 = t_2 = t_t$ termelési periódus, vagyis

hanem

$\Delta t \not\Rightarrow 0$

$\Delta t \Rightarrow P$

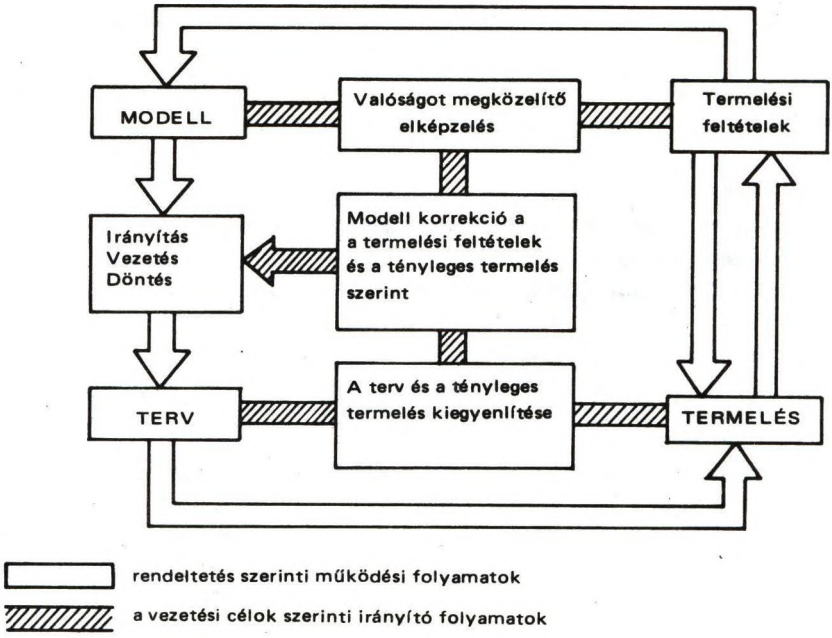
vállalati operativitási időperiódushoz mely a tervezési rendszer függvénye.

A közgazdasági rendszerek adaptív (öntanuló, környezethez igazodó) sajátossága hasonló a természetes rendszerek tulajdonságához.

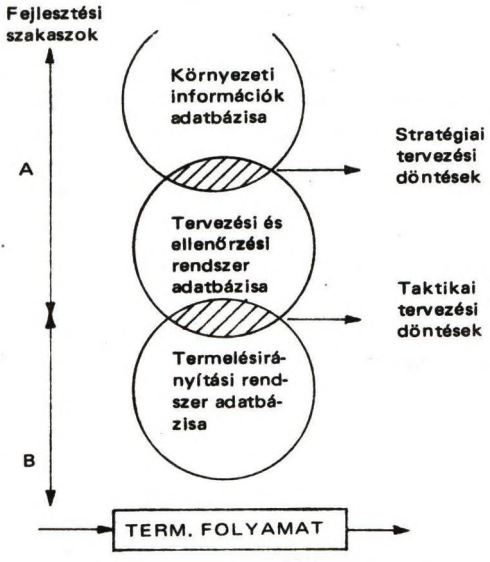
A rendszer viselkedési jellemzőinek ismeretén alapuló „egésze" történő rendezés tehát elsősorban vezetéselméleti és módszertani kérdés.

Az ismert vezetési módszerek általában tisztán, mint egyik vagy másik vezetési elv nem valósulhatnak meg. Azt azonban mondhatjuk, hogy egyes vezetési elvek dominálók lehetnek az adott vállalat helyzete és kitűzött céljai függvényében.

E vezetési rendszereket támogató számítógépes irányítási rendszerek tulajdonságai gyakorlatilag bármilyen vezetői célok támogatását tudják biztosítani a 3. ábra szerinti összefüggésben.



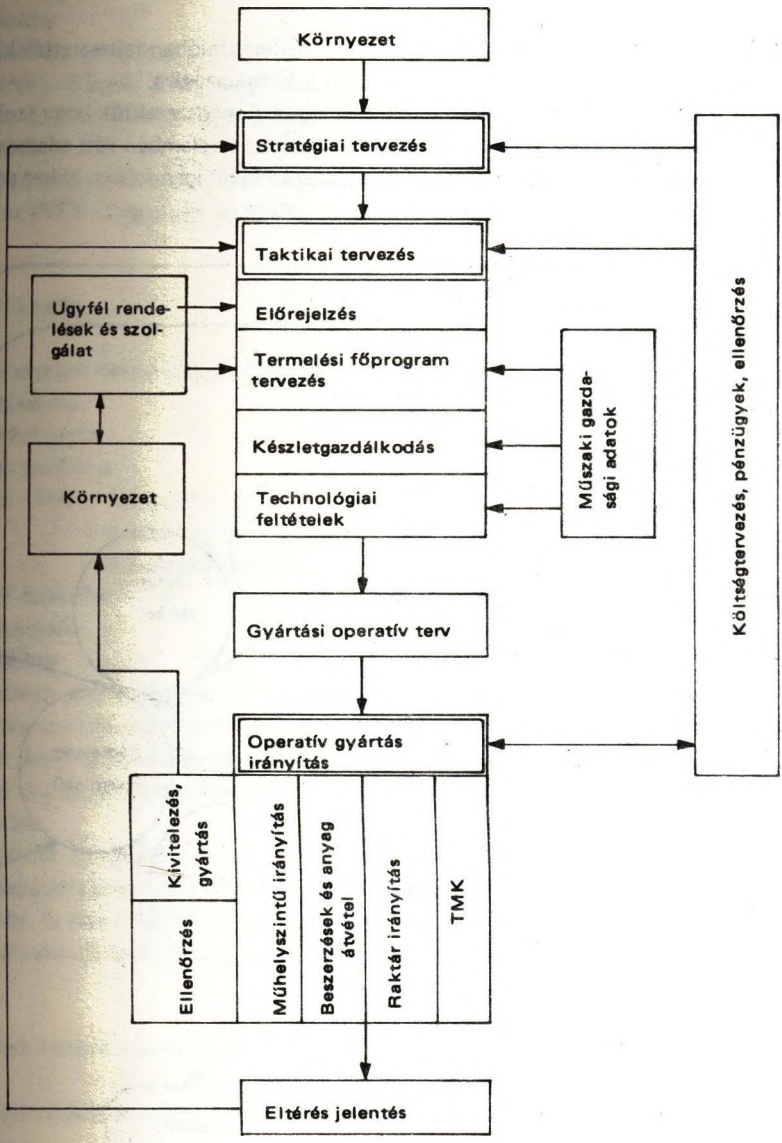
3. ábra



Számítógépes működési környezet	
Kötegelt feldolgozás	Távadat feldolgozás
Számítógépek R10 ..... R40	Számítógépek R15 ..... R55
Op. rendszer „A" DOS 26.2 POWER	Op. rendszer „A" DOS/VS CICS/VS
Adatbázis kezelés ALAPI. (BOMP szint)	Adatbázis kezelés ALAPII. (DL—1 szint) Op.rendszer „B" OS/VS ALAPIII (IMS, IDMS szint)
Ágazati szabvány ajánlás	
Fejlesztés A szakasz	Fejlesztés B szakasz

5. ábra TTIR hierarchikus adatbázis struktúra és működési környezet

Egy korszerű *T*ermelés *T*ervezési és *I*rányítási *R*endszer (TTIR) alkalmazási területeinek uktúrális információs modelljét a 4. sz. ábra és ennek információ feldolgozó adatbázis mo-lljét mint lehetséges megoldási irányokat az 5. sz. ábra szemlélteti.



4. ábra TTIR alkalmazási területek összefüggése

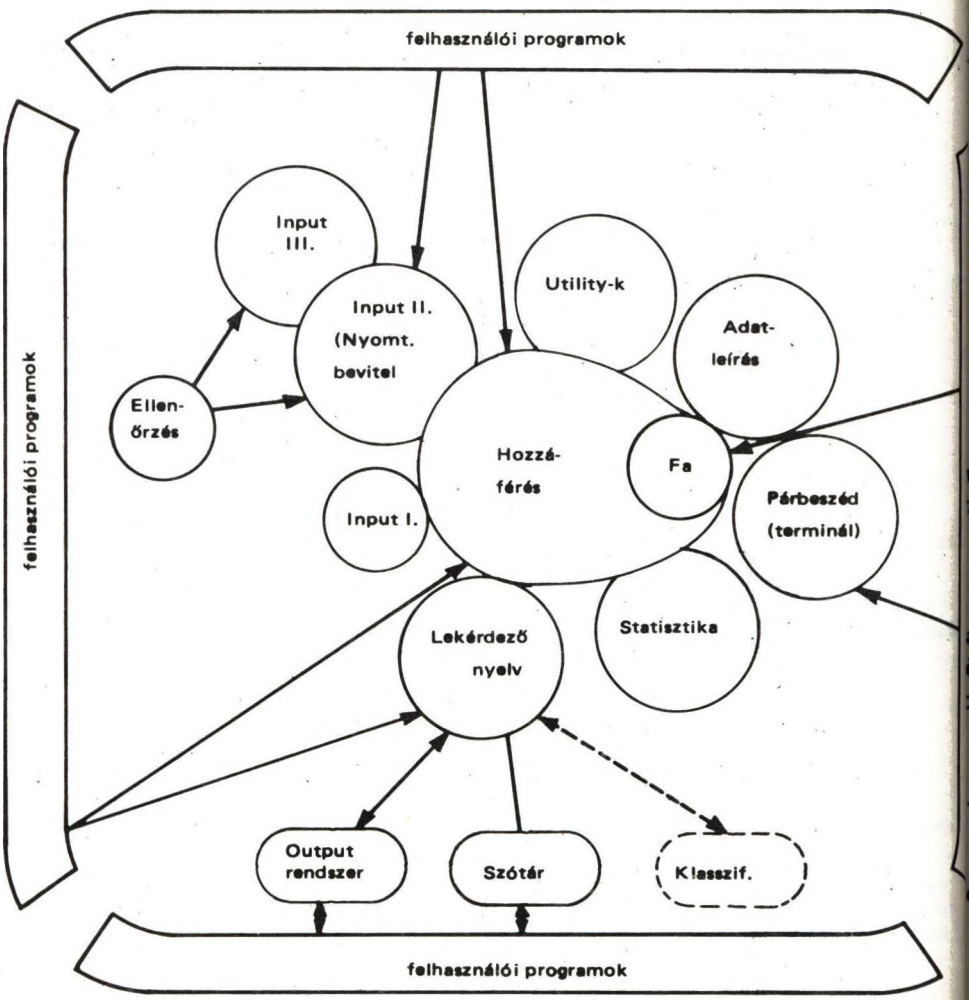
AZ INES–2 ÁLTALÁNOS CÉLÚ INFORMÁCIÓS RENDSZER

Müller Márta–Szmrecsányi Klára
SZÁMKI

1. Bevezetés

Az INES–2 általános célú információs rendszert a Szovjetunióban fejlesztették ki és az ismertetjük, mert alkalmasnak tartjuk hazai alkalmazások kidolgozására.

Az INES–2 abban különbözik az ismert adatbáziskezelő rendszerektől, hogy szolgáltatásaiban bővebb, többet, illetve mást nyújt, mint a hagyományos értelemben vett adatbáziskezelő rendszerek: az adatok integrált tárolását biztosító rendszeren kívül igen fejlett, ehhez csatlakozó (input, output stb.) rendszerekkel rendelkezik.



1. ábra
Az INES–2 alrendszerei

Az INES–2 jelenleg működő alrendszerei (ld. 1. sz. ábra):

- Adatleírás
- Input rendszerek
- Output rendszerek
- Lekérdező nyelv
- Terminálos kezelő alrendszer
- Szótár
- Statisztika alrendszer

Az INES–2 egyik előnyös tulajdonsága, hogy ezek az alrendszerek egymástól függetlenek; ugyanakkor összességükben egy nyílt rendszert alkotnak, amely tetszés szerint bővíthető további alrendszerekkel.

A továbbiakban röviden összefoglalva példa szerűen ismertetjük, hogy milyen lehetőségeket nyújt az INES–2 rendszer az adathalmaz belső ábrázolására, ill. tárolására.

2. Adatstruktúrák

A következő típusú adatok írhatók le:

- egyszerű
- strukturált
  - a) struktúra
  - b) tömb – 1. egyszerű
    - 2. számozott
    - 3. kulccsal ellátott
  - c) feltételes
- pointeres
- sablon szerinti

Egyszerű adat alatt értendő körülbelül mindaz, ami a CODASYL terminológia szerint egy mezőben szerepelhet: decimális szám, bináris szám, lebegőpontos szám, szöveg, literál stb. A *struktúrált* adatoknak három csoportját különbözteti meg a rendszer:

struktúra az, amely különböző fajtájú elemeket egyesít, a *tömb* elemei pdig feltétlenül egyneműek.

Feltételes adattal akkor találkozunk, ha valamilyen feltétel határozza meg a további adatokat (pl.: feltételes adat lehet az iskolai végzettség, ahol más jellegű adatokra vagyunk kíváncsiak középiskolai, és más jellegűekre felsőfokú végzettség esetén.) *Pointeres,* ill. *sablon szerinti* adatokat a redundancia csökkentése érdekében használhatunk.

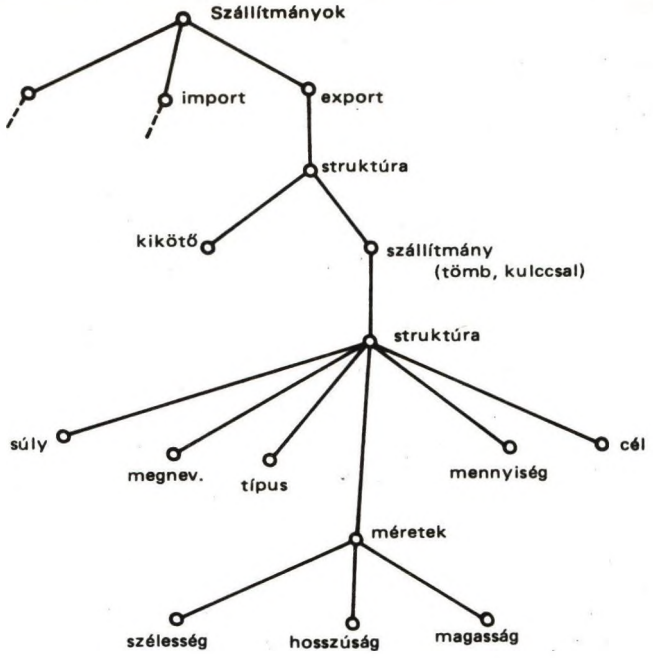
3. Az adatok közötti kapcsolatok ábrázolása

Az információ ábrázolás FÁ-k segítségével történik. Az INES–1 rendszer kétféle fát definiál:

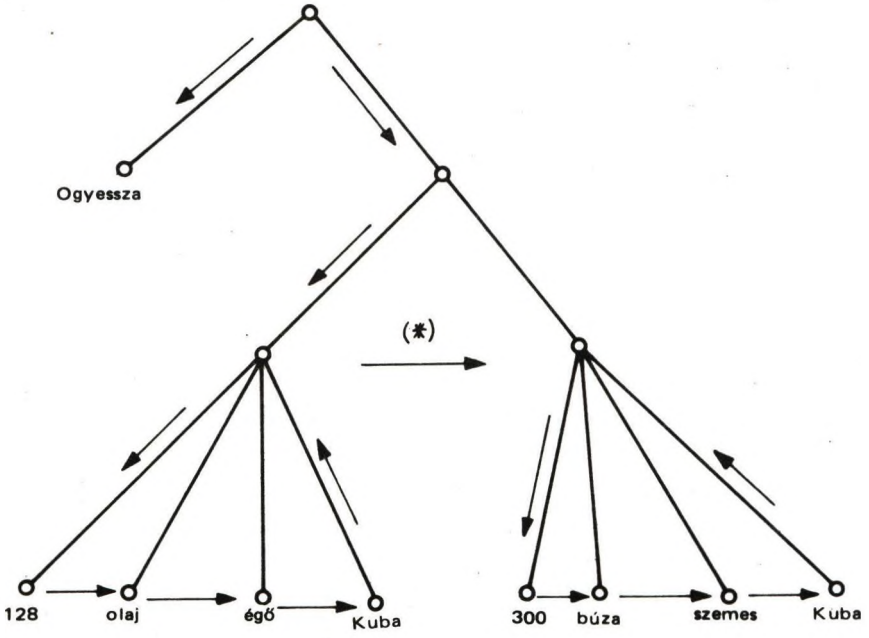
- az adatleírás fáját (ALF)
- az adatok fáját (AF)

Egy adat leírásának megfelel az ALF-ben egy csúcs; a tényleges adatok viszont az AF-ben vannak. Éppen ezért a rendszer mindig együtt használja a kettőt, mindig párhuzamosan mozog bennük.

(Definiál egy ún. megfeleltetett fát – MF – is, amely alatt az együtt tekintett két fát érti.)
A 2. és 3. ábra egy mintapélda ALF-jét és AF-jét ábrázolja.



2. ábra. Egy szállítási feladat ALF-jének részlete



3. ábra. Részlet a mintapélda AF-jéből

A 3. ábrán a nyilak a lekérdezéshez szükséges *mozgásokat* ábrázolják, mint említettük ezek a mozgások általában párhuzamosak az AF-ben ill. A az ALF-ben, kivéve a +-gal jelölt esetet, amikor az AF-ben egy tömb következő elemét veszi elő, viszont az ALF-ben értelemszerűen nincs változás.

Az *input rendszer* lehetőséget nyújt arra, hogy ne csak az adatbázis belső struktúrájának megfelelően vihessünk be adatokat. Definiálhatunk különböző nyomtatványokat és ezután az adatbevitelhez csak a megfelelő formanyomtatvány megnevezésére van szükség. Hasonló lehetőség megtalálható az *output rendszerben* is. A lekérdezés történhet közvetlenül (a rendszer által használt makrók segítségével), de történhet programból is (Assembler, PL/I) és a rendszer rendelkezik egy önálló *lekérdező nyelvvel.*

Bemutatunk egy lekérdezést az alábbi mintapéldához kapcsolódóan:

```
Szállítmányok. export. Ogyessza
Szállítmány  ALL  (cél = 'Kuba')
        (súly, megnev., típus)
              output ('print')
```

A *terminálos kezelő rendszer* alapvetően lekérdezésre szolgál, azonban lehetőséget nyújt módosításra vagy új adat bevitelére is (természetesen ezeket célszerűbb batchben elvégezni).

A *szótár* alrendszer konvencionális: kódok ill. szövegek megfeleltetésére szolgál; a *statisztika* alrendszer – nevének megfelelően – különböző statisztikai szolgáltatásokat nyújt (pl.: százalékszámítás stb.)

4. Hardware és software környezet

Az INES–2 rendszer implementálható tetszőleges ESZR gépen R–20-tól felfelé. Memóriaigénye hozzávetőlegesen 150 K (batch), ill. 350 K (terminálos rendszer). OS operációs rendszer alatt működik; a jelenlegi terminálos rendszer a GAM elérési módszert használja.

5. Összefoglalás

Az INES–2 rendszer az első, hazai felhasználók részéről hozzáférhető, önálló fejlesztésű rendszer a hazai közepes nagyságú ESZR berendezésekre.

Az adatbáziskezelő az interaktív és batch alkalmazásokat támogatva software átmenetet biztosít a magasabb szintű és integritású rendszerekhez.

Az INES rendszer a MNIIPU, VNIISZI és a SZÁMKI együttműködés keretén belül került be, a REI támogatásával, az országba. A hazai adaptáció és kísérletek most folynak. A rendszer hiányosságai között kell megemlíteni a megfelelő dokumentáció és a tervezési segédeszközök hiányát.

A SZÁMKI és a VNIISZI további közös fejlesztéseket tervez, továbbá a SZÁMKI segíteni kívánja a rendszer hazai oktatását és az OSAK-on, valamint a bázisintézeteken keresztüli terjesztését.

AZ R15 SZÁMÍTÓGÉP ALAPVETŐ MŰSZAKI– ÉS RENDSZERTECHNIKAI JELLEMZŐI

Németh Pál – Köves Péter – Mannhardt Endre
SZKI

1. Az R15 általános jellemzői

1.1 Az R15 alapvető felhasználói jellemzői

Az R15 – EC 1015 – számítógép az ESZR „RJAD–2" család legkisebb tagja, a „RJAD–2" többi modelljével együtt az ESZR továbbfejlesztés eredménye.

Számítási kapacitása és input-output teljesítménye alkalmassá teszi az R15 hatékony felhasználását: mind önálló rendszerekben, mind számítógép hálózatok állomásaként.

Műszaki teljesítményén túlmenően az R15 számítógép korszerű ESZR architektúrája tékony alkalmazási lehetőségeket biztosít mind a gazdasági adatfeldolgozásban, mind a mű ki-tudományos számítások területén. A számítógép konfiguráció lehetőséget nyújt lokális é távoli rendszerek kiépítésére, valamint batch és interaktív üzemmódú működésre.

Az R15-öt a következő jellemzők emelik ki különösen:

a) Az R15 központi egység több önállóan és egyidejűleg működő, párhuzamosan sz vezett egységet (processzort) tartalmaz, melyek kezelő- és vezérlő funkciókat látnak el a centralizált feldolgozás (intelligencia) elvének megfelelően.

b) A központi egységhez illesztett képernyős konzolon keresztül a kommunikációs tőségek a felhasználó és a rendszer között lényegesen javulnak, az ember-gép kapcsolatok szerűsödnek.

c) Az R15 felülről lefelé kompatibilis a RJAD–1 modellekkel és programkompatibi a RJAD–2 rendszer modelljeivel.

d) Az R15 központi egység által realizált teljesítmény szintetikus mutatói a követk
- GIBSON I kb. 20.000 műv/sec
- GIBSON III rövid kb. 33.000 műv/sec
  hosszú kb. 18.000 műv/sec
- GPO–WU–II kb. 30.000 műv/sec

e) kibővített pontosságú lebegőpontos műveletek (28 hexadecimális karakter),

f) a dinamikus címfordítás lehetővé teszi a program részére egy 16 Mbyte-os virtu tár használatát,

g) a tárvédelem, a dinamikus címfordítással együtt biztosítja a fő tároló tartalmán delmét, ami lehetővé teszi több program egyidejű futtatását,

h) a gépi hibaelhárításra szolgáló fejlett technika kiterjed a hibák kijavítására, mik sítások megismétlésére, valamint a gép állapotára vonatkozó információk rögzítésére,

i) a rendszer a programok végrehajtásakor ellenőrzi az utasítások és adatok helye és megkülönbözteti a program hibákat a gép hardware meghibásodásától,

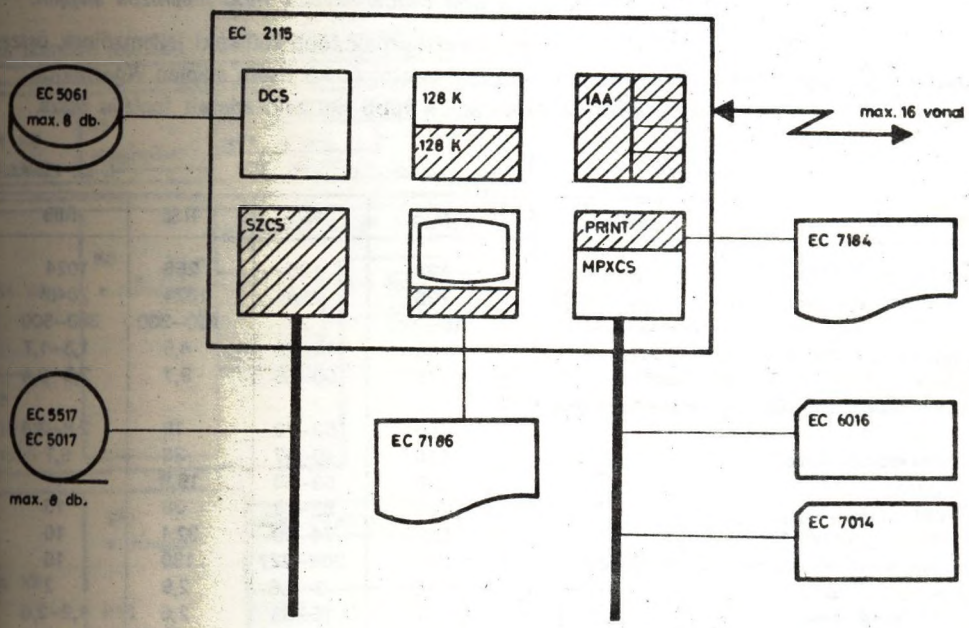
j) a monitor és a PER (program események naplózása) megkönnyítik a tesztelést operátori beavatkozást nem igénylő program belövést anélkül, hogy ez kihatna az egyi leg futó programokra,

k) operatív tár max. 256 Kbyte-ig bővíthető,

l) az R15 utasítás készlete megfelel az ESZR továbbfejlesztés követelményeinek. Magába foglalja a lebegőpontos, bináris és decimális aritmetikai utasításokat, de nem tartalmazza a közvetlen vezérlés és a multiprocesszoros rendszer vezérlés utasításait.

1.2 Alapvető műszaki jellemzői és konfigurátora

Az R15 konfigurátorát az 1. ábra mutatja.



1. ábra Az R15 konfigurátora

A központi egység rendelkezik mindazon decimális, lebegőpontos, dupla pontosságú lebegőpontos utasításokkal, timer-ekkel, virtuális tárkezelési lehetőségekkel, program belövést segítő eszközökkel, amelyek a RJAD–2 architekturát jellemzik.

Az operatív tár 128 kB méretű, ami opcionálisan 256 kB-ig bővíthető. Ez a tárméret az alapvetően alkalmazott DOS/VS típusú operációs rendszer mellett, már lehetőséget ad OS/VS típusú rendszerek alkalmazására is.

A gép standard eszközei közé tartoznak az EC 5061-es 30 MB kapacitású, 50 msec átlagos hozzáférésű cserélhető mágneslemez tárolók, melyből max. 8 darab használható. Ennek megfelelően a maximális online tároló kapacitás 240 MB, ami a rendszer, könyvtár és alkalmazói területek megfelelő megválasztása esetén hatékony feldolgozást tesz lehetővé. A gép operátori konzolja display típusú, amely egyben – lényegében szöveges megadással – az operátorosi panel funkcióit is realizálja. A display képernyőn megjelenő szövegekről az opcionálisan a rendszerhez kapcsolható EC 7186 mátrix nyomtatóval kaphatunk másolatot. A nyomtató latin és cirill karakterek kinyomtatására is alkalmas. Nyomtatási sebessége 180 karakter/sec.

Az EC 7184, 1100 sor/perc nyomtatási sebességű, latin-cirill jelkészletű sornyomtató. A berendezés az R10–R12 konfigurációk bevált egysége, amely az EC 2115-ben programozott formátum kezeléssel is el lett látva. A gép opcionális integrált adatátviteli adaptere max. 16 adatátviteli vonal kezelését látja el, amelyek négyes csoportokban bővíthetők. Az egyes

négyvonalas egységek szinkron vagy aszinkron vonali eljárást realizálhatnak, a szabványos sebességeken. A maximális összesített adatátviteli sebesség 21 Kbit/sec.

A multiplex csatorna, amelyre a kártyás berendezések (EC 6061, EC 7014) vagy egyéb lassú perifériák kapcsolódhatnak, 32 alcsatornát realizál. A maximális adatátviteli sebesség multiplex üzemmódban 17 kB/sec, monopol üzemmódban 27 kB/sec. A gép opcionális szelektor csatornájának adatátviteli sebessége 160 kB/sec.

1.3 Az R15 összehasonlítása a RJAD–2 más modelljeivel a főbb jellemzők alapján.

Az R15, EC 1025, EC 1035, EC 1055 számítógépek főbb műszaki jellemzőinek összehasonlítását az approbációs konfigurációk műszaki adatai és kiépítése alapján végezhetjük.

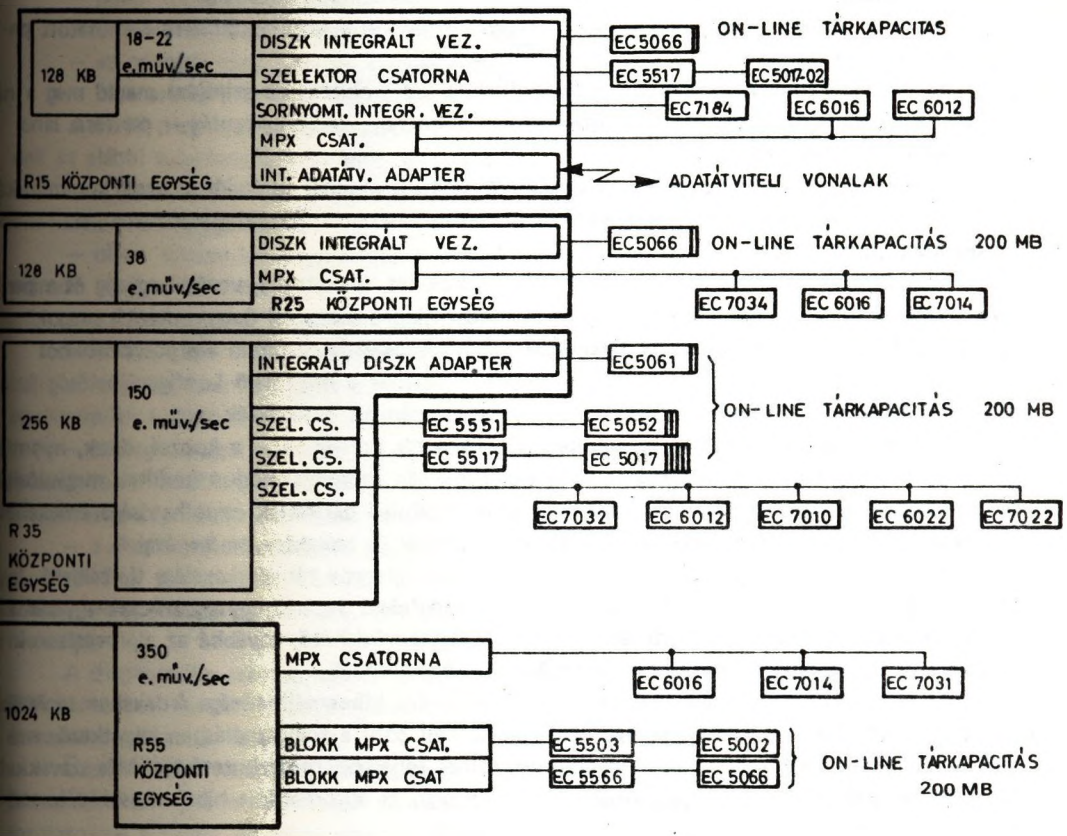
Az 1.sz. táblázat az approbációs konfigurációk főbb műszaki adatait foglalja össze.

1. sz. táblázat

PARAMÉTEREK / MODELL	R15	R25	R35	R55
Memória-kapacitás (Kbyte)	128	128	256	1024
Maximális bővíthetőség (Kbyte)	256	256	1024	2048
Processzor teljesítmény (ezer műv./sec.: GIBSON I.)	18–22	38	150–200	350–500
Fixpontos összeadás/kivonás (μsec.)	18	14–28	4,5	1,3–1,7
Lebegőpontos összeadás/kivonás (μsec.)	76	50–65	9,7	2,8–3,4
Duplapontosságú lebegőpontos összeadás/kivonás (μsec.)	98	53–70	15	2,8–3,4
Fixpontos szorzás (μsec.)	210	40–47	23	6,7
Lebegőpontos szorzás (μsec.)	235	53–69	19,8	11
Fixpontos osztás (μsec.)	290	82–89	38	13
Lebegőpontos osztás (μsec.)	285	74–90	32,1	10
Duplapontosságú lebegőpontos osztás (μsec.)	745	209–227	180	16
Vezérlés átadás (μsec.)	15	3–7,6	2,9	2
Rövid műveletek (μsec.)	17	15–33	2,6	1,2–2,6
Ciklusidő	1,1 μsec.	olvasás:500 nsec	2 μsec	1,4 μs
Csatornák max. száma	3	5	5	5
Csatornák átbocsátó képessége (Kbyte/sec.)	multiplex:17	25	40	multiplex:2 burst
	monopol: 27	74	120	40 mult
	szelektor:160	–	1 M	blokk mult. 1500
				3000/2 by
Adatátviteli adapter vonalkezelése (vonalszám)	S/S: 8	–	–	–
	BSC: 4	–	–	–
Felvett teljesítmény (kW)	7	10	41	35
Alapterület (m^2)	35	75	110	200–250
Max. Súly (kg)	500	500	nincs megadva	nincs megadva

A 2. sz. ábra az approbációs konfigurációk kiépítését szemlélteti.

<OPERATIV TÁR > PROCESSZOR < CSATORNA > VEZ. EGYS. < HÁTTÉR TÁR >
<PRINTER > KÁRTYA OLV. < KÁRTYA LYUK. > SZALAG OLV. < SZALAG LYUK.>



2. ábra Approbációs konfigurációk

2. Az R15 alapvető tervezési elvei és azok megvalósítása

2.1 A struktúra kialakításának elvei

Az R15 meghatározó konstrukciós elvei a számítógép *konfiguráció* kialakításának és üzemeltetésének komplex elemzésén alapulnak, lehetővé téve az adott teljesítményszinten összességében alacsony hardware volument, és ebben a gépkategóriában megszokott szinthez képest a rendelkezésre állás javítását.

Az R15 lényegében ugyanazon az alkatrész- és technológiai bázison épül fel, mint a RJAD–2 sorozat nagyobb teljesítményű tagjai. Ennek megfelelően az R15 kisgépnek megfelelő feldolgozó képessége (sorozatra általában jellemző ár-teljesítmény viszony mellett) a hardware volumen csökkentését eredményező struktúrális megoldásokon alapul. A rendszert – az ésszerűség határáig fokozva – a beépített elemek sokszoros kihasználása jellemzi.

– A főbb műszaki megoldások szekvenciális jellegűek. (1 Byte-os adatutak, nagyfokú mikroprogramozottság)

– Időosztásos üzemmódban több feladat oldható meg egyidejűleg. Jellemző példa: etranse típusú mikrporogramok időosztásos futtathatósága a mikroprocesszorokban, ami ilt tővé teszi ugyanannak a mikroprogramnak egyidőben több, tipizált egység vezérlésére valm felhasználhatóságát. (pl.: adatátviteli multiplexor vonalainak kiszolgálása.)

Az R15 rendszer integráns egységet képez, beleértve a konfigurátorban bemutatott rifériákat is.

Ennek az elvnek megfelelően, csak logikailag (az architektúra szintjén) marad meg gyobb modellekre jellemző tároló- (processzor)- csatorna – vezérlő egység – periféria lá nak megfelelő struktúra.

A nagyobb modellekre jellemző struktúra fenti elemeinek funkciói és azok kommu ciója általában nagy volumenű hardware-ben, esetleg részben mikroprogramozottan, de kisz külön kerül megoldásra.

Elemezve az architektúrából következő funkciókat, a csatorna, vezérlő egység és am féria vezérlés funkciói egyetlen mikroprogramozott alaprocesszorral összevontan is megol tók. Az R15 rendszer az operatív tárolóhoz párhuzamosan kapcsolódó alprocesszorokból (mikroprogramozott processzorokból) áll, amelyek számát a megfelelő konfigurálhatóság ba ellátandó funkcióból következő teljesítmény követelmények határozzák meg.

A legfontosabb, az alkalmazásokban legtipikusabb perifériák – a konzol, diszk, nyo adatátviteli multiplexor – esetében a vezérlési funkciók integrált módon kerültek megvalós ra. A felhasználói szintű vagy speciális igényeket kielégítő perifériák csatolhatósága érdekéb rendelkezik az R15 egy kisebb teljesítményű multiplex és szelektor csatornával is.

A leírt struktúrát funkciók szerint elemezve az operatív tár címkezelése tipizálható az operatív tárkezeléshez szükséges eszközöket a megfelelő vezérlő egységben csak egyszer megépíteni. Ezzel a tárhozzáfordulás algoritmusa egyszerűsíthető, továbbá az alprocesszorok az operatív tár között a szükséges vonalszám is csökkenthető.

Az R15 struktúra kialakításánál a gép minél jobb kihasználhatósága érdekében ten szempont volt, hogy hardware jellegű támogatást kell adni a felhasználásban jelentkező, beri tényezőktől függő szűk keresztmetszetek minél teljesebb megszüntetésére. Más szava az előírt megbízhatóságon, hibafelismerési biztonságon és automatikus hibajavításon túlme en az adott kategóriában a szokásosnál hatékonyabb

– operátor-gép kapcsolat és
– ön-diagnosztikai mechanizmusok

kerültek megvalósításra.

Az operátor-gép kapcsolat műszaki-gazdasági szinten elemzett és meghatározott „int telligensebb" megoldása egy alprocesszor feladata. A realizált display-es konzol és operátor pult szöveges formában ad meg és dolgoz fel minden operátori beavatkozást. Az output üzenetek sebességét nem korlátozzák mechanikus berendezések.

Egy alkalmazásban felhasznált rendszerben fellépő bármilyen fel nem ismert hiba tévedés súlyos gazdasági következményekkel járhat. Ezért azt mondhatjuk, hogy a hibafelis merés és behatárolás ugyanolyan fontos tényező, mint a műveleti sebesség, az átviteli te sítmény, a kihasználtsági tényező stb. Ezért volt számunkra fontos követelmény, a rendsz üzemeltetés és karbantartás hatékony eszközeinek kifejlesztése a rendszer összhatékonysá nak fokozása érdekében. Az ESZ 1015 modell ennek megfelelően hatékony diagnosztik módszereket alkalmaz a helyes működés ellenőrzésére és a hibák behatárolására.

A mikrostruktúra szintjén – függetlenül a realizáláshoz szükséges hardware többlet konzekvensen alkalmazásra kerültek a hibafeltáráshoz, hibakezeléshez tartozó alábbi meg dások:

– paritásellenőrzés az adatutakon:

– inverz bit alkalmazása a vezérlő tárakban (a vezérlő tárban egy bit hiba javítását teszi lehetővé, amennyiben a mikrporogramok futása folyamán az adott bit átírása nem történik meg);

– check bitek alkalmazása (vezérlések ellenőrzése);

– mikroutasítások műveleti kód kombinációinak felhasználása hibajelzésre;

– az operatív tárolóban az 1 bites hibák javítása és a 2 bites hibák jelzése;

– hiba esetén bizonyos esetekben mikroutasítások ismétlése, stb.

Az R15 diagnosztikai rendszerének a kialakításakor abból indultunk ki, hogy a rendszernek az alábbi tulajdonságokkal kell rendelkeznie:

– a hiba detektálását és behatárolását a legkisebb, helyszínen cserélhető egységre, magas színvonalon kell megoldania;

– olyan egyszerűen legyen alkalmazható, hogy kevesebb és a területen kevésbé képzett mérnök is alkalmazni tudja;

– a végrehajtáshoz legyen elegendő a rendszer minimális részének hibátlan működése;

– a hiba detektálását – beleértve a zavarokat is – és behatárolást a lehető leggyorsabban kell végrehajtania és a hiba diagnózist a karbantartó mérnökökkel szabatosan, tömören és egyértelműen kell közölnie;

– a diagnosztikai rendszer legyen olyan mértékben teljes és önálló, hogy a karbantartó mérnök minél inkább mentesüljön a szükséges programcsomag és az ellenőrzendő adatok megválasztásával kapcsolatos döntések alól;

– a diagnosztikai eszközök legyenek függetlenek a rendszer konfigurációitól.

Az R15 struktúrában ezeknek a célkitűzéseknek a megvalósítását szolgálja az alprocesszorok írható mikroprogram tára, és a kiszolgáló alprocesszor, amely a konzol funkciókat is realizálja.

A diagnosztika szempontjából ez az alprocesszor egy külön sínrendszeren keresztül az R15 processzor minden részegységéhez (kb. egy-egy kártyányi részéhez) külön-külön hozzáfér és azon vizsgálatokat folytathat.

Üzem közben lekérdezi, és egy diszketten feljegyzi a beépített ellenőrző áramkörök által jelzett hibákat. Hibakeresés esetén analizálja a feljegyzett hibákat, zavarokat, illetve tesztminta sorozatokkal vizsgálja az egyes részegységek összetevőit, illetve olyan mikroprogramokat tölt be az alprocesszorok mikroprogram tárába, amelyekkel azok a hozzájuk csatolt részegységeket, perifériákat tesztelhetik.

A hagyományos diagnosztikai rendszer ebben az esetben átalakul, a műszaki kiszolgálási programcsomag legtöbb elemét mikrotesztek váltják ki.

Software elemekkel olyan eszközök diagnosztizálása történik, amelyek nem integrált módon kapcsolódnak a központi egységhez és így nem használhatók ki a mikrodiagnosztika előnyei:

a) A mikrodiagnosztika sokkal jobban kapcsolódik magához a hardware-hez, ezért jobb megoldást nyújt a hibák detektálásában és a legkisebb cserélhető egységre történő behatárolásban.

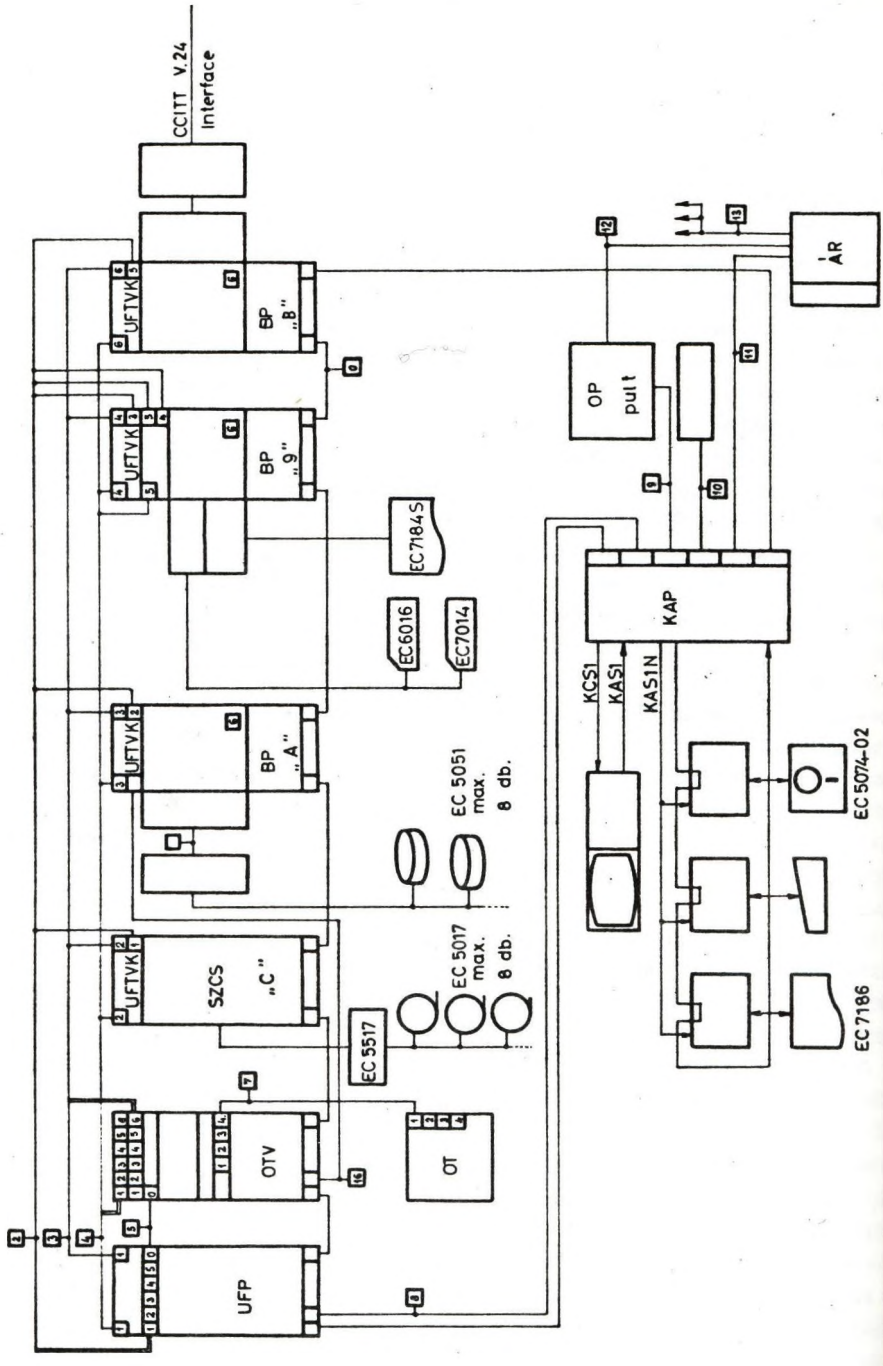
b) A mikrodiagnosztikai eszközök gyorsabban és kevésbé költségesen állíthatók elő és tárolhatók, mint a software eszközök.

c) A mikrodiagnosztika sokkal kevesebb működtetendő „hard core"-t igényel.

A vázolt diagnosztikai rendszerhez nagyvolumenű mikroprogram tartozik. Az R15 struktúrában azonban ez nem követeli meg a funkcionális mikroprogramok által meghatározott mikroprogram tároló méretek növelését, mert az összes mikroprogramot a kiszolgáló alprocesszor által kezelt diskett tartalmazza. A diskettről mindig csak a szükséges mikroprogramok kerülnek behívásra.

2.2 A megvalósított struktúra néhány jellemző részlete

Az R15 alapvető struktúráját a 3. ábra mutatja.



3. ábra Az R15 alapvető struktúrája

A rendszernek két központi eleme van.
Az operatív tárhoz és tárvezérlőhöz (OT, OTV) sugarasan csatlakoznak
– az utasítás feldolgozó alprocesszor (UFP)
– a különböző periféria csatolásokat vezérlő input-output bázis processzorok, valamint
– a szelektor csatorna.
A másik központi elem, a kiszolgáló alprocesszor (KAP), amely diagnosztikai funkció-
ának megfelelően egy külön sínen keresztül a rendszer összes többi elemével kapcsolatban
an.

2.3 A tervezési elvek hatása a hardware-firmware struktúrákra

Az R15 számítógépben megvalósított hardware – firmware arányait a 4. ábra mutatja.
Megállapítható, hogy az összes felhasználói szinten értelmezett „hardware" (OT eredő
ektor)
– Firmware komponensében
  - a hagyományos értelemben vett CPU funkciók 10%-os
  - a periféria csatolások, csatornák 20%-os
  - a mikrodiagnosztika 70%-os
  súllyal szerepel.
– Hardware komponensében
  - a hagyományos értelemben vett CPU 10%-ot
  - a tároló 30%-ot
  - a periféria csatolás, csatorna realizálás 60%-ot
  tesz ki.

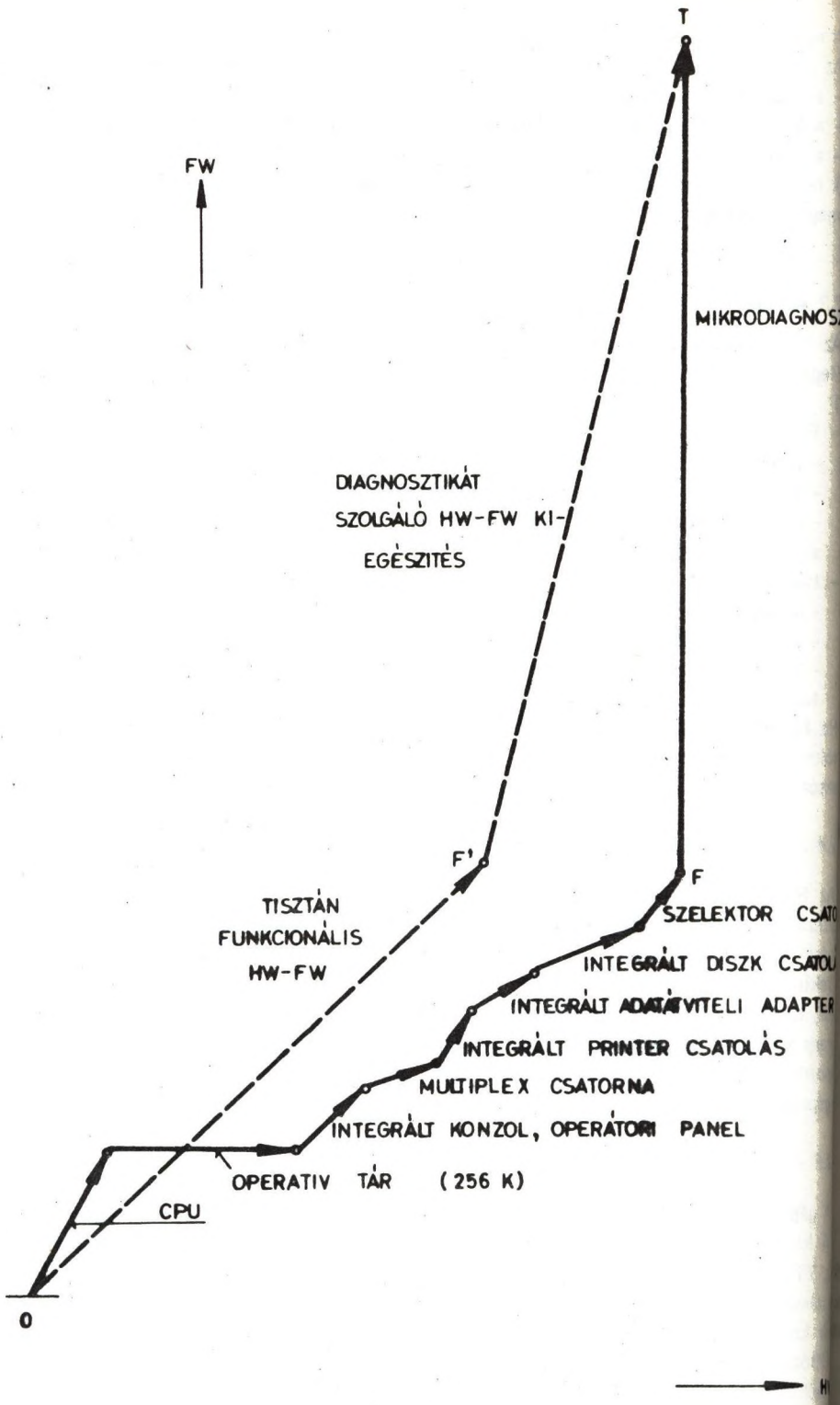
Ha figyelembe vesszük a kifejezetten hibakezelési, detektálási célokból beépített hard-
vare volumenét, akkor a funkcionális és hibakezelő tulajdonságok megvalósítását az OF' és
'T vektor felbontással jellemezhetjük.

2.4 Az R15 megvalósított architektúrája

Az R15 architektúrája minden tekintetben megfelel a RJAD–2 működési elveknek. A di-
ekt vezérlés és a multiprocesszing lehetőségének kivételével a RJAD–2 sorozat összes lehetősé-
e, vonása realizálásra került.
Tekintettel arra, hogy a rendszer általános szervezése, a program végrehajtás, a rendszer
ezérlése, ezen belül a program állapot szó, control regiszterek, tároló védelem, a monitoring,
program esemény feljegyzés, a TOD óra, a timer-ek, a megszakítások, az általános-decimális,
ebegőpontos és dupla lebegőpontos utasítások, a géphiba kezelés, az I/O rendszer tulajdonsá-
ai, a gépcsaládon belül egyformák, itt ezt nem részletezzük.

2.5 Az R15 konstrukciós jellemzői

A gép konstrukciója, áramköri elembázisa az R10–R12 bevált megoldásait fejleszti to-
ább. A kialakított struktúra nem teszi szükségessé az ECL áramkörök alkalmazását, a TTL 1,
TL 2, TTL 3 sorozatok kerültek felhasználásra. A kártyák dupla TEZ méretűek, kétoldalas
yomtatással, direkt csatlakozókkal. A blokkokhoz tartozó hátlapok nyomtatott kivitelűek, a
lokkok közötti kapcsolatok szalagkábelekkel kerültek megoldásra.
Hálózati transzformátor nélküli, kapcsoló üzemű, kb. 80% hatásfokú tápegységek ke-
ültek felhasználásra. Az egész EC 2115 központi egységhez tartozó elektronika egy szek-
ényben helyezkedik el.



4. ábra Az EC 2115 HW és FW arányai

A konstrukció alapját a vonatkozó ESZR szabályok előírásai képezik, mely előírásoknak a berendezés minden tekintetben megfelel, illetve ezen előírásokon túlmenően a konstrukció figyelembe veszi az ESZR C2 SZT Báziskonstrukciós és Ergonómiai Munkacsoport, 1975. VII. 15–18. moszkvai munkacsoport ülés ajánlásait is, biztosítva ezzel a legkorszerűbb konstrukciós követelményeknek és technológiának való megfelelést.

Előadásunkban kísérletet tettünk arra, hogy

– összefoglaljuk az R–15-nek, a RJAD 2 legkisebb tagjának a főbb műszaki, rendszertechnikai jellemzőit,

– kiemeljük azokat a tervezési elveket és megoldásokat, amelyeken a rendszer megbízhatósága, műszaki/gazdasági jellemzői, konfigurációs sajátosságai alapulnak,

– hangsúlyozzuk azokat a megoldásokat, amelyek a RJAD 2 más gépeiben kisebb súllyal, vagy egyáltalán nem szerepelnek.

AZ SZKI TIME SHARING SZÁMÍTÓGÉP RENDSZERÉNEK KIALAKÍTÁSA, OPTIMALIZÁLÁSA

Németi Tibor
SZKI

A számítógép felhasználás széleskörű elterjedése, a növekvő számú képzett szakembe számítóközpontoktól egyre újabb és kényelmesebb szolgáltatásokat vár.

A számítástechnika elterjedésének kezdetén a programozók közvetlenül a „gépen" d goztak. A későbbiekben a számítógépek teljesítményének növekedésével már nem tudta e len programozó a gépet hatékonyan használni, kialakult az úgynevezett zárt üzem, amiko kiadott feladatokat a programozótól függetlenül futtatják a gépen. Ez jó diszpécselésnél b tosítja a gép optimális kihasználását, ugyanakkor az ember-gép kapcsolat jelentős romlásá vezetett.

A fenti ellentmondást oldják fel a time-sharing (TS) rendszerek, amikor a terminál teraktív üzemmódban dolgozó felhasználó úgy látja mintha ő lenne a gépen, ugyanakkor terminálok illetve egyidőben dolgozók felhasználók nagy száma biztosítja a gép jó kihasz lását.

1. TS rendszer eszközei

Az SZKI 1976. I. negyedévében kezdte meg time-sharing rendszerének kiépítését e Siemens számítógép és körülötte VIDEOTON terminálokból álló hálózat kialakítását (1.

A rendszer fontosabb elemei:

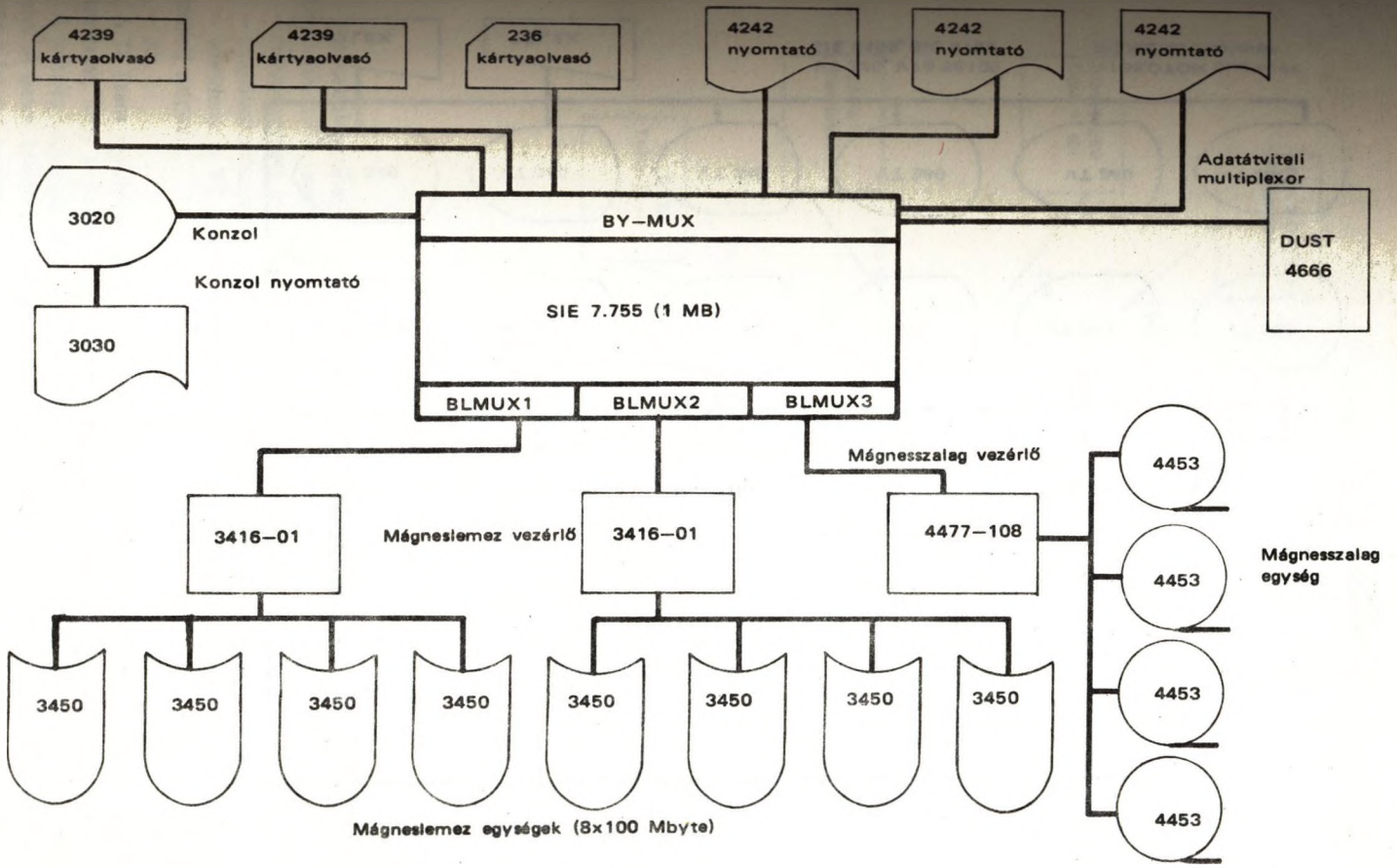
- 7.755 központi egység 1 MB-os memóriával,
- két vezérlőegység 800 MB mágneslemez kapacitással,
- 4 db mágnesszalagegység,
- a szükséges papírperifériák,
- a TAF hálózat egy DUST 4666 típusszámú multiplexoron keresztül csatlakozik számítógéphez (2. ábra).

A csillaghálózat három év alatt fokozatosan bővült, jelenleg több mint 30 terminá (3. ábra) érhető el a számítógép napi 10—12 órában TS üzemmódban. Az éjszakai órák és munkaszüneti napokon a gép batch üzemben működik.

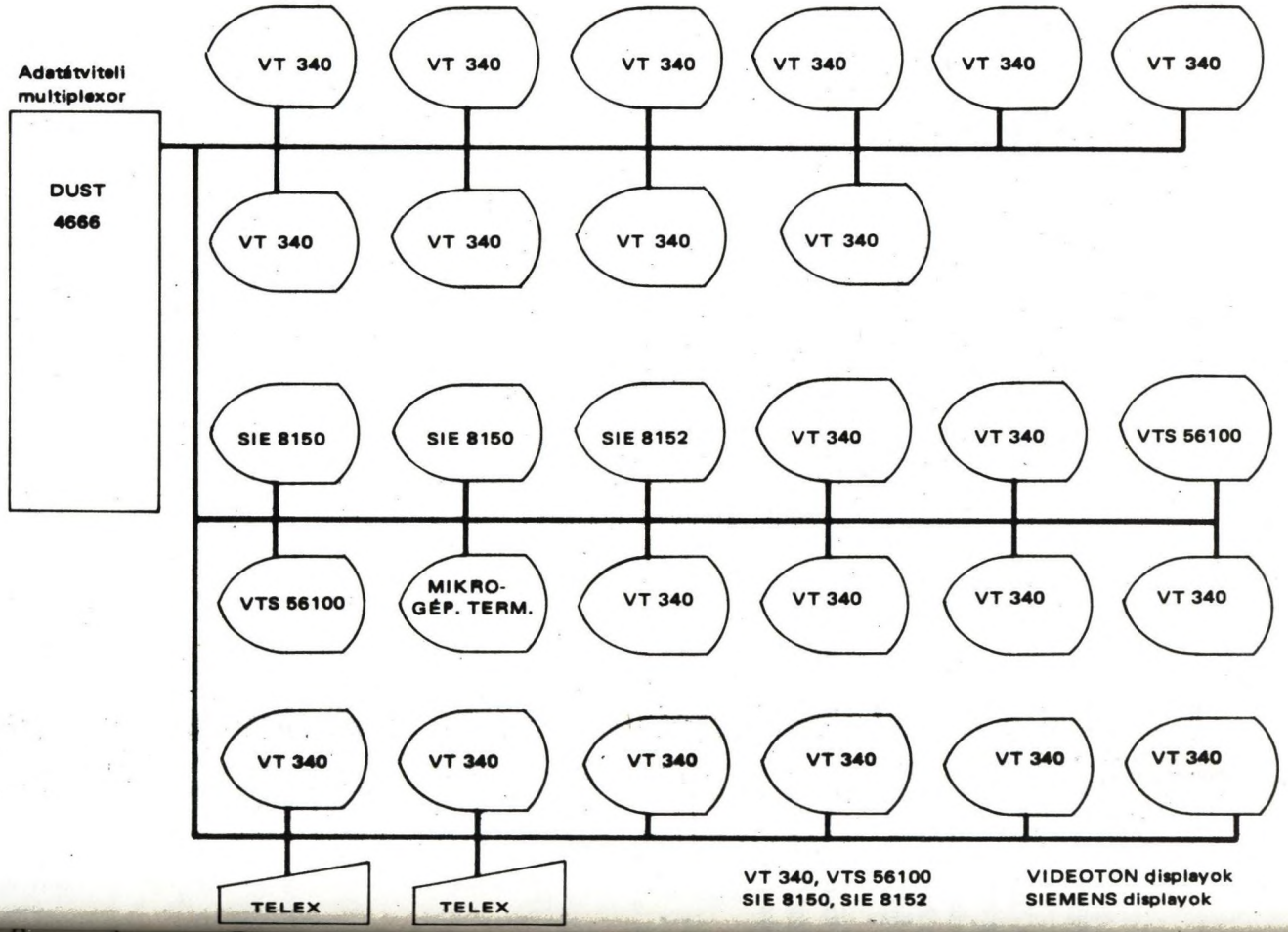
2. A TS rendszer üzemének kialakítása

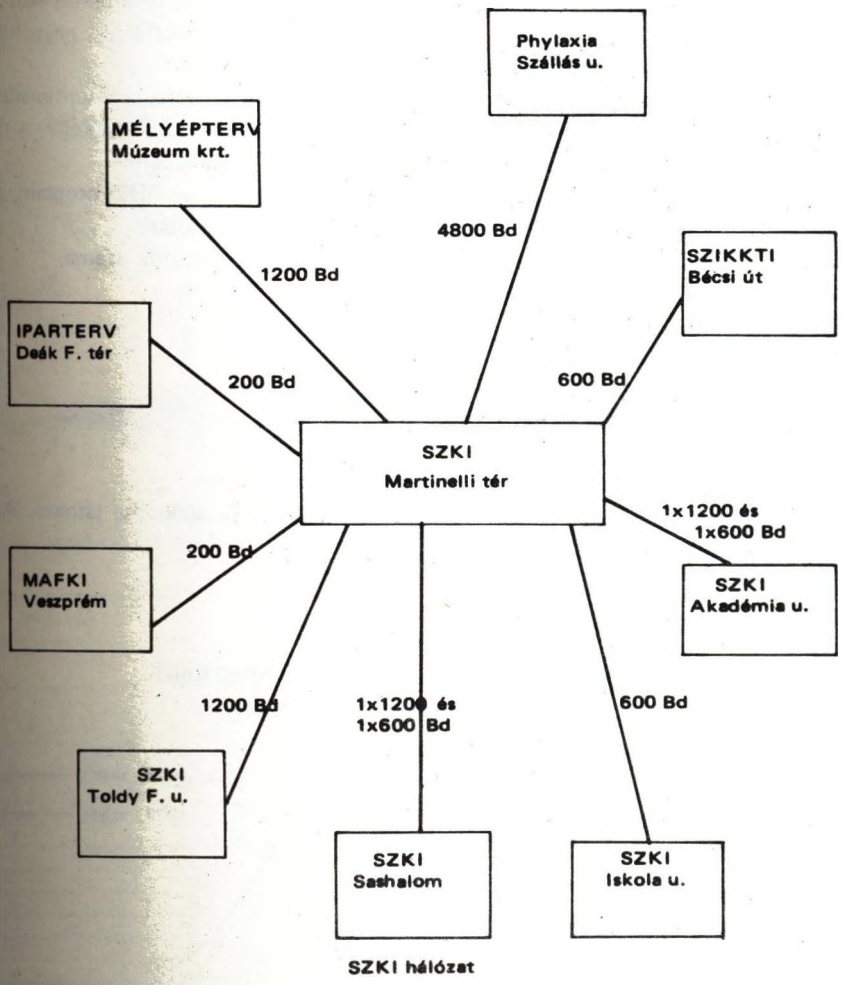
A batch és TS rendszerek munkájának szervezése között — ami a munka optimali sát illeti — az a legnagyobb különbség, hogy a feladatok megoldásának nagy részét term lon keresztül maga a felhasználó és nem a géptermi operátor végzi, ezért a gép terhelés mértéke állandóan — véletlenszerűen — változik. Az operátornak, ill. az ún. rendszerad nisztrátornak lehetősége van, hogy a rendszer terhelését szabályozza, pl.

— operátori paranccsal korlátozhatja a párhuzamosan, egyidőben futó bach, illetv teraktív job-ok maximális számát (batch illetve dialóg limit beállítása),


4239
kártyaolvasó
4239
kártyaolvasó
236
kártyaolvasó
4242
nyomtató
4242
nyomtató
4242
nyomtató
Adatátviteli
multiplexor
3020
Konzol
Konzol nyomtató
3030
BY–MUX
SIE 7.755 (1 MB)
DUST
4666
BLMUX1
BLMUX2
BLMUX3
Mágnesszalag vezérlő
3416–01
Mágneslemez vezérlő
3416–01
4477–108
4453
4453
4453
4453
Mágnesszalag
egység
3450
3450
3450
3450
3450
3450
3450
3450
Mágneslemez egységek (8x100 Mbyte)

1. ábra


Adatátviteli multiplexor
DUST 4666
VT 340
SIE 8150
SIE 8152
VTS 56100
MIKRO-GÉP. TERM.
TELEX
VT 340, VTS 56100
SIE 8150, SIE 8152
VIDEOTON displayok
SIEMENS displayok



SZKI hálózat

3. ábra

– megakadályozhatja, hogy a várakozási területről a hosszan futó job-ok végrehajtása megkezdődjék (CPU limitidő),

– felfüggesztheti az egyes programok futását,

– megváltoztathatja a batch és az interaktív job-ok prioritását,

– leállíthatja a TS üzemet, ha észreveszi, hogy a rendelkezésre álló public-space (lemezterület) megtelik,

– stb.

A TS üzem kialakításánál a gép egyenletes leterhelése mellett egy másik nagyon fontos szempont, hogy a dialóg felhasználók elfogadható válaszidő mellett tudjanak dolgozni.

A terhelés nagyságát elsősorban a batch feladatok számának korlátozásával lehet szabályozni. A batch feladatok indítása a kártyaolvasóról, vagy terminálról történhet. A beolvasás után a feladat a várakozási sorba kerül. Az adott feladatot az operációs rendszer automatikusan indítja, ha a futó batch feladatok száma kisebb, mint a megengedett. Csúcsidőben

(délelőtt és ebéd után) általában 3–4 batch feladat párhuzamos futását célszerű megengedi amelyek – a terminálok tehermentesítésére – elsősorban tesztfutások, fordítások, stb. Az ilyen rövid feladatok kiválasztása automatikusan történik (TIME paraméter), a nagy futások az operátor a csúcsidő utánra halasztja.

Az üzemeltetés egyik legnagyobb problémája a lemezterület-felhasználás optimalizálása és „igazságos" korlátozása, tudniillik, ha a rendelkezésre álló szabad terület elfogy, a rendsz leáll, a futó feladatok eredménye elvész, a feladatokat elölről kell kezdeni.

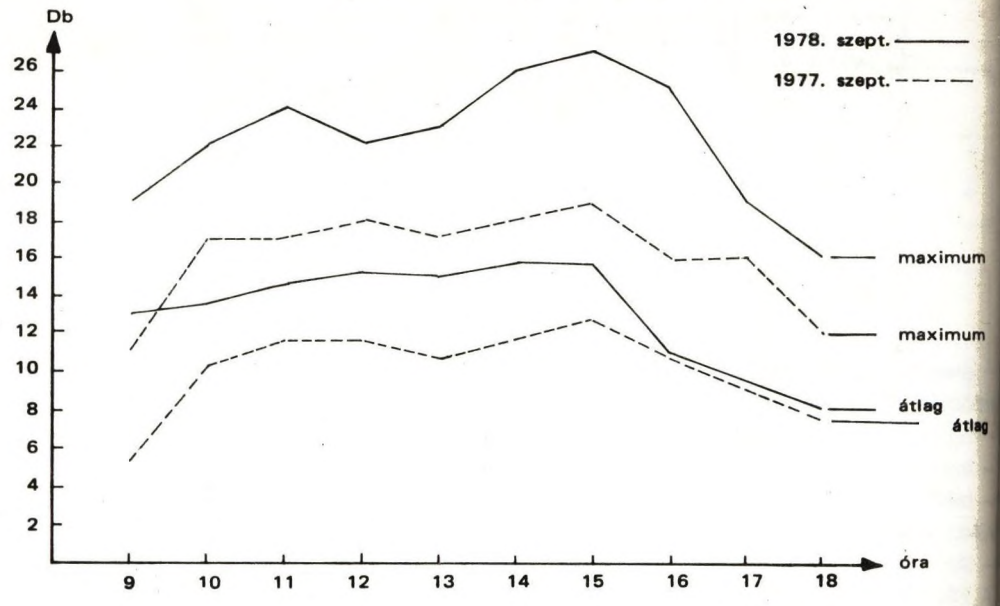
A TS üzem „kézbentartásához" szükséges adatokat részben a számláló program, részb pedig speciális figyelő illetve mérőprogramok adják. Ilyen adatok például:

- a batch, illetve dialóg üzemmódban párhuzamosan futó feladatok száma,
- a feldolgozásra váró (spool-in) feladatok száma,
- a kinyomtatásra váró (spool-out) feladatok száma,
- a terminálok kihasználtsága,
- lemezterület foglalás,
- egy terminálra eső központiegység idő,
- stb.

Az üzem jellemzésére néhány terhelési diagramot bemutatunk.

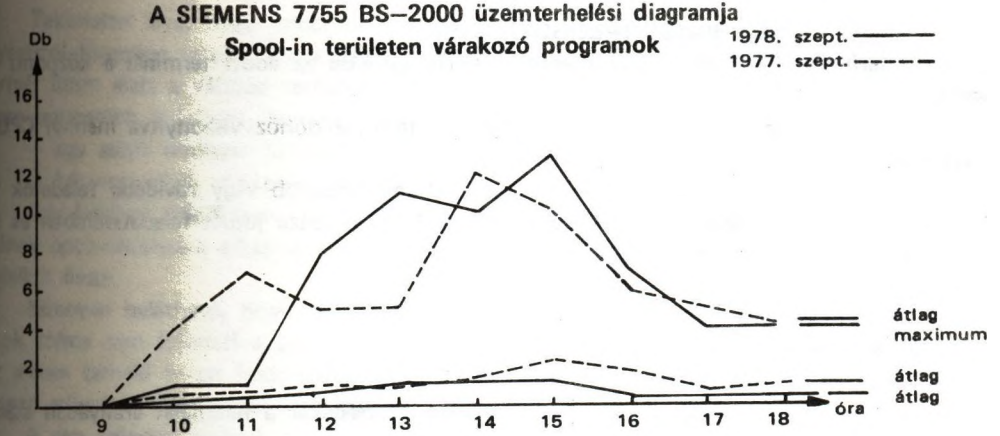
A 4. sz. ábrán a futó batch és dialóg programok összegének alakulása látható. A diagram két év hasonló időszakának adatait hasonlítja össze.

A SIEMENS 7755 BS–200 üzemterhelési diagramjai
Futó batch és dialóg programok összege



4. ábra

Az 5. sz. ábrán a feldolgozásra várakozó feladatok számának alakulása követhető.



5. ábra

A 6. sz. ábra táblázata a terminálok forgalmáról ad tájékoztatást.

1979. APRILIS HO

LINE	TASK		NORM DEFINIALESU	LOGON-LOGOFF IDO		CPU IDO		CPU / CONNECT	
	DB	ATLAG	DB	%	ORA:PERC	ATLAG	SEC.	ATLAG	%	ATLAG
12	19	12	17	89	7:14	10	2 143	42	0,4624	4491
13	316	195	301	95	109:40	199	9 462	169	1,1626	25
14	292	179	285	97	148:00	176	7 201	144	1,1193	22
15	312	193	303	97	109:55	199	15 254	294	2,2424	152
16	9	6	9	100	9:20	10	0	0	0,0000	2
17	33	20	24	72	8:25	9	216	4	0,7714	69
18	19	12	17	89	12:48	13	671	13	1,1455	133
19	49	30	39	80	73:54	77	1 212	24	0,4566	31
20	367	227	338	92	179:49	185	9 667	192	1,0015	1.4
21	83	51	72	87	39:01	41	3 349	67	2,1152	160
22	17	11	17	100	3:40	3	0	0	0,0000	0
23	324	200	310	95	134:52	144	6 952	139	1,1465	92
24	368	227	340	92	190:40	200	7 640	152	1,1115	76
25	335	207	320	96	168:02	170	6 322	126	1,1422	90
26	117	72	108	92	55:09	100	3 792	76	1,1101	76
27	224	140	192	86	147:57	155	6 263	125	1,1259	119
28	137	86	99	95	89:57	94	3 725	74	1,1150	79
29	276	171	266	97	144:50	151	6 125	122	1,1178	91
30	6	4	6	100	7:45	8	434	8	1,1551	103
32	276	171	266	96	137:15	136	6 924	138	1,1472	131
33	374	231	352	94	175:45	137	9 955	199	1,1547	178
34	112	69	110	98	69:27	72	3 266	64	1,1228	55
35	258	159	252	98	149:02	156	7 233	144	1,1248	92
36	1	1	1	100	0:50	0	0	0	0,0000	0
37	212	131	142	76	155:57	160	1 627	32	0,2205	22
38	33	20	26	77	22:01	24	751	15	1,1135	44
39	148	91	134	92	140:23	150	18 974	378	2,1305	2420
40	4	2	3	75	0:15	0	112	2	1,1787	60
41	5	2	5	100	0:25	0	0	0	0,0000	11
(552)	4 692	---	4 391	94	2767:27	---	145 219	---	1,1452	1332

ATL. TASKSZAM		PERIFERIAEGYS.	
PERC	ATLAG	STATIK	TIZED
26	25		3
76	122	2	2
35	93	4	4
27	117	2	2
49	175	2	2
15	42	1	2
46	114	2	2
90	253	3	3
29	72	1	1
21	33	2	1
2	5	1	1
94	97	2	2
59	66	1	1
29	39	1	4
49	139	2	4
42	110	3	3
52	145	2	3
21	29	1	3
78	219	2	4
29	39	6	5
29	33	2	4
37	105	2	3
25	94	1	3
2	2	1	1
44	125	5	4
57	164	1	1
42	122	1	3
49	133	0	1
2	2	2	2
75	170	1,0	1,0

6. ábra

A táblázatból megállapítható a terminál

– forgalma (az indított feladatok száma: TASK),

– kihasználtsága (LOGON–LOGOFF idő),

– terhelés (CPU idő), azaz mennyire vette igénybe az adott terminál a központi számítógépet,

– hatékonysága (CPU/LON–LOF), tehát a terminálidőhöz viszonyítva mennyi CPU időt használtak,

– használatának néhány jellemzője, tehát hogy hosszabb vagy rövidebb feladatok tottak (ÁTL. TASKIDO) illetve, hogy a munka hányad része jutott feladatszámban és ben a készülékre (RÉSZESEDÉS).

3. Számlázás

A jó számlázó rendszer a gépüzemmel járó költségeket a használat arányában szá ja el a felhasználók között.

A rendszer folyamatosan adatokat gyűjt az egyes programok futásáról és azt az nevezett számlafile-be tárolja. Ezekből az adatokból állítja össze a program a számláka

Az SZKI TS rendszere dialóg üzemmódban három erőforrást vesz figyelembe, ne zetesen

– a CPU időt,

– a terminál időt (Logon) és

– az elfoglalt on-line lemezterületet (PAM),

A tapasztalat szerint a költségek a három erőforrás között egyenlő arányban osz janak meg. A terminálidő egységára a várható átlagos terminálkihasználásból és az üze terminálok számából számítható. Az átlagos terminál kihasználás mértékét az üzemi-sta tikai rendszer szolgáltatja. Hasonló módon számítható a CPU idő-egységár is a statiszti ból nyert adatok alapján (egységnyi terminálidő alatt felhasznált CPU idő).

A felhasznált on-line lemezterület az adott TS-rendszer sajátos jellemzője, így arr részletesebben kitérünk. A rendszer alapvető szolgáltatása, hogy a felhasználók bármiko bármelyik terminálról azonnal elérhetik a tárolt adatok egy részét, az ún. on-line tárol adatokat. Természetesen a rendelkezésre álló on-line terület korlátozott, így ez az egyi legkritikusabb „szűk-keresztmetszet". A tapasztalat azt mutatja – mint korábban emlí tük –, hogy ha ez a terület a telítettséghez közel van, a rendszer először lelassul, ma „összeomlik".

A fentiekből az is látható, hogy on-line lemezterület esetén nem fizikai lemezbe ről, hanem a gép egy kritikus erőforrásának felosztásáról van szó.

A TS rendszert használó adatainak, programjának tárolására saját mágneslemezt (inkább is lemezterületről kellene beszélni) vagy mágnesszalagot is használhat, amely azzal a h ránnyal jár, hogy a munka megkezdése előtt az információt egy megfelelő munkaterü be kell vinni, majd a feladat befejezése után – ha nincs on-line lemezterület – akkor mét ki kell íratnia a „saját" adathordozóra.

A fenti számlázási algoritmus alkalmazása batch feladatok esetén nem lenne „iga gos", mert a logon idő (benntartózkodási idő) nem a programozótól, hanem a progra zótól független körülményektől (együttfutó programok száma, a gép terhelése, stb.) fü Ezért logon idő helyett a periféria hozzáférések alapján számított időt vesszük alapul

4. A rendszer (HW, SW) hatékonysága, teljesítménye

Tekintettel arra, hogy mind multiprogramozott, mind time-sharing környezetben több egymástól független feladat fut a gépen egyidejűleg, a rendszer teljesítményének mérése a normál üzem alatt a változó terhelés miatt nehezen vagy egyáltalán nem lehetséges. Lényegesen egyszerűbb a feladat, ha

- egy adott rendszer üzemének optimumát próbáljuk megmérni és beállítani vagy
- két vagy több rendszert méréssel egymáshoz hasonlítunk.

Egy korábbi előadásban [1] részletesen tárgyaltuk a time sharing rendszer teljesítményének optimalizálására alkalmas eszközöket, most csupán a mérés fontosabb eredményeit foglaljuk össze.

Könnyen belátható, hogy time sharing környezetben a gépben egyidejűleg futó feladatok száma nem jellemzi a gép leterheltségét — miután ezen feladatok egyik része a gépet erősen terhelő batch feldolgozás, a másik általában a — nagyobbik fele — pedig a programozó stílusától függő interaktív munka.

A gép teljesítményének egyértelmű meghatározására olyan paramétert vagy paramétereket kell találni, amelyek üzem közben könnyen figyelhetők és jellemzik a gép terhelési állapotát. Művi környezetben, amikor a leadott teljesítmény pontosan mérhető, (pl. elvégzett utasítások száma) — egy sor mérés segítségével meg lehet állapítani, hogy bizonyos paraméterek hogyan alakulnak a terhelés függvényében, ill. hogy ezek milyen értékénél adja le a rendszer a legkedvezőbb körülmények mellett a legnagyobb teljesítményt. Ezután már csak az a feladat, hogy normál üzem közben ezeket a paramétereket figyeljék és software, ill. adminisztratív eszközökkel a terhelést (programok számát) úgy változtassák, hogy ezen paraméterek minél közelebb kerüljenek az előírthoz.

Az adott TS környezetben a legfontosabb figyelt paraméterek a következők:

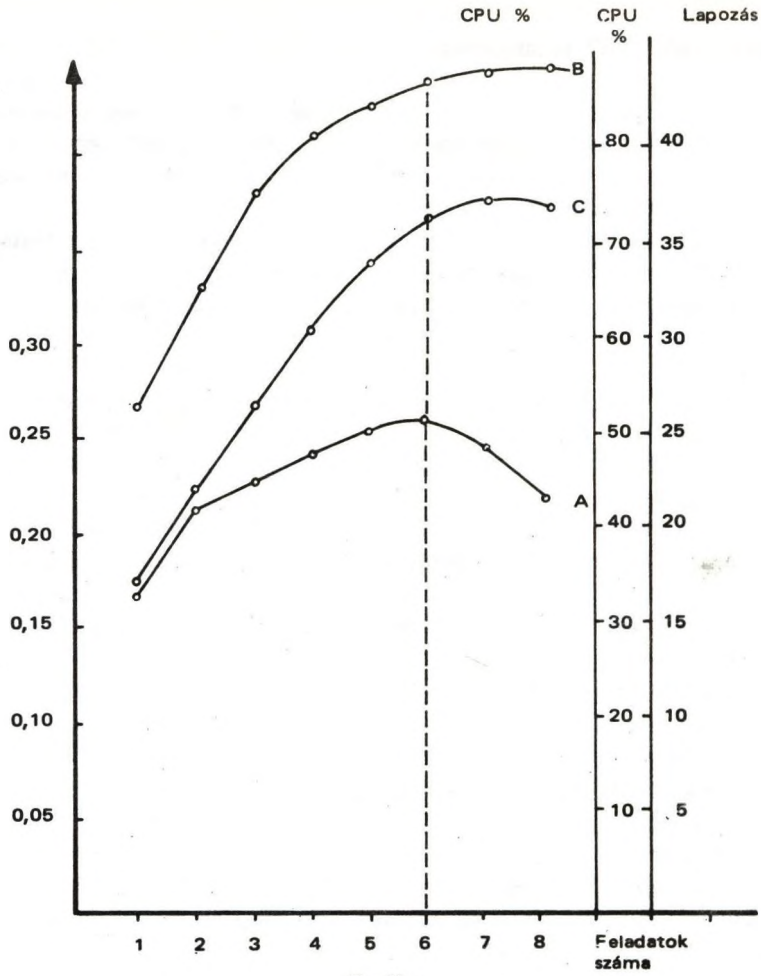
- a központi egység terhelése %-ban,
- a másodpercenkénti lapozások száma —pageing),
- a központi tár és háttértár közti hasznos adatforgalom,
- válaszidő.

A fenti adatok folyamatos figyelésével, ill. a művi környezetben mért adatoknak a napi üzemmel való összevetése alapján lehetségessé válik, hogy

- az üzemi terhelést az optimumhoz közeli értékben tartsuk
- felfedjük a rendszer kritikus pontjait,
- és megállapítjuk a rendszer „szűk keresztmetszetét".

A 7. sz. ábrán látható egy ilyen „művi környezetben" való mérés eredménye. Az egyidőben futó programok számát fokozatosan növeltük 1-től 8-ig és mértük a gép jellemző paramétereit. A leadott teljesítmény mérése céljából a programok egységnyi munka (utasítás MIX) elvégzése után üzenetet küldtek a terminálra. Ezen üzenetek „sűrűsége" jellemezte az össz leadott teljesítményt. A mérés bebizonyította azt a tapasztalatunkat, hogy a túlterhelt gép (több mint 6 feladat) össz teljesítménye csökken. Jól látható, hogy a rendszer kritikus paraméterei (CPU, lapozás) mikor kerülnek a maximum közelébe. A tapasztalat szerint a napi üzem alatt célszerű a maximum alatt maradni.

A már hivatkozott korábbi tanulmányban [1] kimutattuk, hogy a software paraméterek változtatásával és ezen változások hatásának folyamatos figyelésével cca. kétszeresére volt emelhető a rendszer áteresztő képessége.



7. ábra

Irodalom:

[1] „Evaluation of a time-sharing Computer System" Operációs Rendszerek Elmélete V. Téli Iskola, Visegrád, 1979. január.

PROGRAMOK HATÉKONYSÁGÁNAK JAVÍTÁSÁRÓL

Dr. Obádovics J. Gyula
MÜM, SZÁMTI

1. Bevezetés

A kisszámítógépek hardware és software adottságaiban bekövetkezett kedvező változások valamint a számítógépfelhasználók szemléletváltozása együttesen azt eredményezte, hogy a használatba állított kisgépek száma lényegesen meghaladta az előrebecslések számértékét. A magasszintű programozási nyelvek, a nagygépek szolgáltatásait megközelítő operációs rendszereik a kisszámítógépek sokoldalú felhasználását biztosítják. A tapasztalatok azt bizonyítják, hogy mind a vállalati, mind az oktatási-kutatási környezetben működő kisszámítógépek jó hatásfokkal alkalmazhatók olyan területeken is, amelyeken korábban csak nagygépek jöhettek számításba. Az alkalmazások elterjedése, bár más feltétel között, ismét időszerűvé tette az elsőgenerációs számítógépek alkalmazásakor keletkezett — a programok tárigény és futási idő szerinti optimalizálására vonatkozó — problémakör vizsgálatát. Mivel a kisszámítógépekhez használhatók a magasszintű programozási nyelvek, de hatékony programoptimalizáló fordítóprogramok nem használhatók az operatív tár és háttér tár kapacitás korlátjai miatt, ezért elsősorban a programozó és programtervező ismeretein, ötletein múlik, hogy milyen méretű feladat vihető kisgépre és a program mennyi idő alatt fut le azon. A programok tárigény és futási idő szerinti optimalizálását egyidejűleg nem lehet minden esetben megvalósítani, de a programok hatékonyságát javítani akár az egyik, akár a másik szempont szerint gyakran gyakran igen egyszerű elvek alkalmazásával módunkban áll.

2. Általános alapszabályok

A programozás stílus elemeit kell számbavennünk ahhoz, hogy jobb, hatékonyabb programok létrehozását elősegíthessük. A programok szerkezetét összehasonlítva a természetes nyelv írásművészeivel az alábbi megfeleltetést figyelhetjük meg:

Programozási nyelv	*Természetes nyelv*
változók	főnevek
operátorok	igék
aritmetikai és logikai kifejezések	kifejezések
utasítások	mondatok
alprogramok, szegmensek	bekezdések, fejezetek
programok	írásművek (novellák, regények)

A programok a természetes nyelv írásműveihez hasonlóan bevezetésre (beviteli rész), tárgyalásra (az eljárást megvalósító programrész) és befejezésre (kiviteli rész) tagolhatók. A különböző programozók által azonos eljárásokra írt megegyező tagolású programok is a változók,

operátorok, kifejezések, utasítások, alprogramok használata, programok terjedelme lényeges
rést mutatnak. Finomabb elemzéssel a programok futási időre is lényegesen eltérhetnek egy
mástól. A stílus tehát szubjektív.

A programok írására azonban objektív tényezők is hatnak, amelyektől a programozók
legyenithetik magukat. Ilyen tényezők pl. az operációs rendszerek a fordítóprogramok, a
ue konfigurációk és az üzemeltetési környezetek különbözősége. A szubjektív és objektív
nyezők vizsgálata alapján arra a következtetésre jutunk, hogy általános érvényű, ellentmon
mentes és teljes programozási szabályrendszer kidolgozását nem tűzhetjük ki célul. Megfoga
mazhatunk azonban olyan szabályokat, amelyek betartásával vagy azok közül egy-egy sza-
bály tudatos megsértésével az adott feltételrendszerhez illeszkedő, közel optimális program
írására vállalkozhatunk. Ehhez megfelelő programozói magatartásra is szükség van, éspedig
olyanra, amely a rendelkezésre álló források felhasználásával, a rendelkezésre álló időszak ala
a lehető legjobb program megírására törekszik. A jó program pontos (a megvalósított eljárás
által kívánt eredményt adja), számítástechnikailag hatékony, tárolás-igényét tekintve takaré
moduláris, fordítóprogramok speciális szolgáltatásaitól független és jól dokumentált.

A jó program szerkesztéséhez célszerű betartani a következő általános alapszabályokat,
amelyeket a prózai írásművekhez kidolgozottakból kis módosítással nyerhetünk.

— *Előbb tervezz és csak azután programozz!*

Nagyon fontos szabály, melynek megsértése sokszorosan megbosszulja magát. Átgon
programtervezés nélkül írt program mind a hibakereséshez, mind a programmódosításhoz az
elhagyott tervezési idő többszörösét igényli a programozótól. Különösen a kezdő programoz
hajlamos egy-egy programrész gyors elkészítésére, anélkül, hogy átgondolta volna az alprogra
mok és programrészek összefüggéseit, a program modularitásának feltételeit. Fordítsunk gon
a logikus és értelmes programszervezésre, mert a hibakeresést ezzel nagymértékben elősegíti

— *A programod az olvasó számára is érthető legyen!*

A programozó, ha betartja ezt a szabályt, akkor remélheti, hogy a program elkészítése
után több hónap elteltével is használni tudja a saját programját, az újraíráshoz szükséges sz
lemi ráfordítás nélkül. A programba célszerű COMMENT sorok segítségével a felhasznált
módszerről, alprogramokról, speciális változókról stb. a program olvasói, felhasználói számá
tömör megjegyzéseket írni. A programozási munkát nem tekinthetjük befejezettnek mindad
amíg a mintapéldát is tartalmazó teljes dokumentáció nem áll rendelkezésünkre.

— *Ellenőrizz és javíts!*

Minden elkészült programot alaposan ki kell elemezni pontosság, érthetőség, modulari-
tás, hatékonyság szempontjából és a körülmények figyelembevételével, a lehető legjobb pro
ram elkészítéséért, bátran át kell írni a már késznek ítélt programot is.

— *Fogalmazz tömören, de ne az érthetőség rovására!*

Kerüljük a bőbeszédűséget. A program ne tartalmazzon felesleges utasításokat, az uta
tásokból küszöböljük ki a felesleges számításokat. Pusztán a program rövidítésére kiagyalt s
katlan ötletek megvalósítása azonban több kárt okozhat, mint hasznot. A program megérté
segítsük elő COMMENT sorok beültetésével.

— *Ne erőltesd a különleges megoldást!*

Vannak programozók, akik a különleges, ravasz megoldások hívei, mert ezáltal megtal
rítanak egy-egy bitet vagy néhány mikrosecundumot, de nem gondolnak arra, hogy az ilyen
„eredeti" program egy más környezetben vagy egy későbbi futásnál zavart okozhat. A szám

gép a programozó „okosságát" nem tudja értékelni, de a szabványos megoldásokat tartalmazó programokat támogatja.

– *Ne térj el a nyelv szabványaitól!*

A magasszintű programozási nyelvek különböző változatai olyan egyedi utasításokat is megengednek, amelyek előnyösek lehetnek az egyszerűbb, hatékonyabb programok megírására, azonban használatukkal a szabványos nyelv általánosan elterjedt fordítóprogramjai számára érthetetlen programot készítünk. A fordítóprogramtól függő programok szűk körben használhatók, más rendszerre való átvitelük fáradtságos és sok hibalehetőséget rejt magában.

A felsorolt alapszabályokat minden programozó alkalmazhatja a jó program megírása érdekében, de betartásuk nem biztosítja az abszolút jó program előállítását.

3. Speciális hatékonyság javító elvek

Ebben a pontban a futási idő és a tárigény csökkentésének néhány egyszerű elvét fogalmazzuk meg. A példákat FORTRAN IV programozási nyelven írt programrészek szolgáltatják.

A futási idő a számítási időt, a szükséges várakozási időt és a feltételes várakozási időt tartalmazza. A számítási idő az, amely a program utasításainak végrehajtására kell, a szükséges végrehajtási idő az, amit a programnak várnia kell egy elkezdett esemény befejezéséig, a feltételes várakozási idő az, amit a programnak a multiprogramozási környezetben más programok miatt kell várnia. A programozó a számítási időt és a szükséges várakozási időt csökkentheti jó hatásfokkal, a feltételes várakozási időt a programozó nem módosíthatja. A számítási idő csökkentése néha növeli a program tárolási igényét, és a szükséges várakozási időt. A központi tárigény csökkentése viszont nem egyeztethető össze minden esetben a futási idő csökkentésének kísérletével. Általában mindkét törekvés megvalósítása növeli a programozási munkát. A programozási munkát befolyásoló szempontok:

- a programot hányszor kell futtatni,
- mennyi a számítási idő csökkentésével elérhető megtakarítás,
- a programfejlesztésre mennyi idő áll rendelkezésre,
- milyen a program komplexitása,
- mennyi a gépidő költsége,
- mennyi a programozó munkaidejének költsége,
- mennyi a tárolás és a szükséges várakozási idő költsége.

Az egyszer futtatásra kerülő rövid program optimalizálására nem szabad sok időt vesztegetni.

A számítási idő csökkentésére használható általános szabály:

- Az INTEGER műveletek gyorsabbak a REAL műveleteknél, a REAL műveletek pedig gyorsabbak a DOUBLE PRECISION műveleteknél;
- Az összeadás és a kivonás gyorsabb művelet a szorzásnál;
- A szorzás gyorsabb művelet az osztásnál;
- A hatványozás művelete nagyon lassú és általános könyvtári alprogram végzi el.

A számítási idő csökkenthető a program redundáns kifejezéseinek kiküszöbölésével. Alkalmazható elv: Ha egy programban valamely kifejezés értékének megváltoztatása nélkül többször fordul elő, akkor azt egy új változóval jelölve csak egyszer kell kiszámítani, és a kifejezés helyett mindenütt az új változót kell használni.

A szabály alkalmazása előtt ellenőrizzük, hogy a redundáns kifejezés egyes változói az újra felhasználás előtt nem változtatják-e meg értékeiket, mert ha igen, akkor a szabály nem alkalmazható.

Pl:: Az $ax^2 + bx + c = 0$ egyenlet két gyökét kiszámító következő programrészt

```
x1 = (-B + SORT (B * * 2.0 - 4.0 *A * C) ) / (2.0*A)
x2 = (-B - SORT (B * * 2.0 - 4.0 *A *C) ) / (2.0*A)
```

Az elmondott szempontok szerint javítva

```
APLUSA = A + A
GYOKD = SORT (B * B - 4.0 * A *C)
x1 = (-B + GYOKD) / APLUSA
x2 = (-B - GYOKD) / APLUSA
```

programrésszel helyettesíthetjük, s ezzel az alprogramhívás száma négyről egyre, a szorzás és osztás száma nyolcról ötre, az átviteli, tárolási, szervezési utasítások száma 20 felettiről 20 alá csökken, az összeadások száma pedig változatlan marad.

A számítási időt csökkenthetjük olyan hatékony algoritmus kiválasztásával, amely a lehető legkevesebb műveleti lépésből áll.

Pl.: $A\ P(x) = a_1 x^n + a_2 x^{n-1} + \ldots a_n x + a_{n+1}$

polinom $x = x_0$ helyen vett helyettesítési értékének kiszámításához, as számítást formálisan lépésről-lépésre végezve: $\frac{1}{2} n\ (n+1)$ számú szorzást kell végezni, ugyanakkor a Horner eljárás

algoritmusát használva:

$$P(x) = (\ldots (((a_1 x + a_2) x + a_3) x + a_4) x + \ldots) x + a_{n+1}$$

alakban végezve a számítást, mindössze n szorzásra van szükség és az algoritmust megvalósító programrész is nagyon egyszerű;

```
      P = O.
      DO 15 I = 1, NMEG1
15 P = P * X + A (I)
```

Az alprogram H hívás megtakarítását már az előző példában szemléltettük. Az alprogramok közül a nyílt alprogramok (pl. utasításfüggvény, FLOATí IFIX, ABS, stb.) a főprogramba beépülnek minden hívásnak megfelelően, a zárt alprogramok (SUBROUTINE, FUNCTION, SIN, ALOG, EXP, stb.) függetlenek a főprogramtól, egyszer kerülnek tárolásra, ezért tárolás szempontjából használatuk hatékonyabb, de a hívó utasítás hatására létrejövő gépi utasítások a számítási időt növelik és bizonyos tárigényt is jelentenek.

A redundáns kifejezések kiküszöböléséhez hasonlóan fogalmazható meg a szabály az azonos aktuális paraméterekkel többszörösen előforduló alprogramok kiküszöbölésére: Ha egy alprogramot egynél többször hívunk azonos aktuális paraméterrel, akkor azt egy új változóval előlve csak egyszer kell kiszámítani, és a redundáns alprogram helyett mindenütt az új változót kell használni.

Pl.: Az

```
y1 = (EXP (x) – EXP (–x)) /2.0
y2 = (EXP (x) + EXP (–x) ) / 2.0
```

programrész A = EXP (x) , B = EXP (–x) új változók bevezetésével javíthatjuk:

```
A = EXP (x)
B = EXP (–x)     (vagy  B = 1.0/A)
y1 = (A–B) /2.0
y2 = (A+B) /ó.)
```

gy négyszeri hívás helyett csak kétszer, ill. egyszer hívtuk az alprogramot, s ezzel a számási idő lényegesen csökken.

```
x = R/FLORAT (N)
y = P/FLORAT (N)
```

programrészt az FN = FLORANT N vagy FN = N új változó bevezetésével javíthatjuk:

```
FN = N                  vagy FN = FLORAT (N)
x= R/FN                        x = R/FN
y= P/FN                        y = P/FN
```

számítási időt és tárolást is megtakarítottunk az utóbbi programrésszel, mivel a nyílt FLORAT alprogramot csak egyszer építi be a fordítóprogram a programrészbe.

A számítási idő csökkenthető akkor is, ha kiküszöböljük a programból a szükségtelenül használt indexes változókat, valamint ha hatékonyabb indexkezelésre térünk át. Egy tömbelem címét, mivel az adatok fizikai tárolása eltér az adatok logikai szerkezetétől, az index-függvények segítségével lehet előállítani. Az index-függvények gyakori kiszámítása erősen ronthatja a program hatékonyságát, márpedig minden esetben ki kell számítani a címet, ha a tömb bármelyik elemére hivatkozunk.

Ha a tömbnek van olyan eleme, amelyet nagyon gyakran kell elérni, akkor célszerűbb azt index nélküli azonosítóval jelölni, s ezt a skalár változót használni a számításhoz. Ezzel elkerüljük a cím gyakori újraszámítását.

```
Pl. A
K = I
L = I+1
V = A (K, L) /C
W= A (K, L) * C
Z = A (K, L) * * 2.0
```

programrész helyett

a

```
K = I
L = I+1
AKL = A(K, L)
V = AKL /C
W = AKL * C
Z = AKL * * 2
```

programrészt használva csökkentjük az index kiszámítást, sőt az egész értékű második hatvány képzést a fordítóprogramok általában szorzásta fordítják, míg a valós alakban írt második hatvány elvégzése logaritmikusan megy végbe.

Mivel a legtöbb FORTRAN fordító a két indexes tömb elemeit oszloponként balról-jobbra haladva helyei el a tárban lineárisan, ezért a TOMB2 (I, J), elemek lineáris pozícióját

$$(J-1) * NSOR + I$$

képlettel határozhatjuk meg, ahol NSOR a tömb sorainak száma, a három indexes tömb elemei pedig laponként, a harmadik index növekvő értékének sorrendjében, az előző elv érvényesítésével helyezi el a tárban lineárisan, ezért a TOMB3 (I, J, K) elemek lineáris pozícióját a

$$(k-1) * (NSOR * NLAP) + (J-1) * NLAP + I$$

képlettel számíthatjuk ki, ahol NLAP a tömb lapjainak száma.

A számítási idő lényegesen csökkenthető, ha indokolatlanul nem használunk indexes változót, ha azonos indexszel többször felhasználásra kerülő változót skalárral helyettesítünk, ha több többindexes változó azonos indexszel szerepel a programban, akkor csak egyszer számítjuk ki az indexet a lineáris pozíció meghatározására szolgáló képlettel, s a hozzárendelt azonosítóval, mint indexszel, a többindexes változókat egyindexes változóként kezeljük.

Pl. A

```
DIMENSION A (100, 80), B (100, 80), C (100, 80)

. . . . . .
. . . . . .
DO 5 J = 1,80
DO 5 I = 1, 100
5 c (I, J) = A (X, J) + B (I, J)
```

programrész két 100x80-as mátrix összegét képeti. Az A, B, C tömbök azonos indexszel vesznek részt a műveletben. Legtöbb fordítóprogram háromszor számítja ki az (I, J) indexértéket. A feladatot a következő programrész az index-függvény felhasználásával egysoros index-kiszámítással oldja meg:

```
      DIMENSION A ( 100, 80), , B (100, 80) , C (100, 80)
      DIMENSION AL (8000) , BL (8000) , CL (8000)
      EQUITVALENCE (A (1,1) , AL (1) ) , (B (1, 1) , BL (1)) , (C (11) , CL (1) )
      .....
      .....
      DO 5 J = 1,80
      DO 5 I = 1,100
      K = (J–1) * 100 + I
    5 CL(K) = AL (K) + BL (K)
```

vagy egyszerűbben, az utolsó négy utasítás helyett két utasítással is megoldható:

```
      D  5 K = 1,8000
    5 CL(K) = AL (K) + BL (K)
```

Ezt az EQUIVALENCE utasítás teszi lehetővé, mivel így ekvivalenciacsoportonként a tömbelemel címei azonosak,

A programok vizsgálata azt mutatja, hogy a számítási idő nagy részét a programban levő ciklusok használják fel, ezért nagyon fontos a ciklusok oprimalizálása. A ciklus változóitól független változók, kifejezések értékeit a ciklusba való belépés előtt számítsuk ki, s ha kell konstans azonosítókkal használjuk fel a ciklusban.

Pl. Az

```
      ALFA = .314579
      DO 15 I = 1,250
   15 Z (I) = (ALFA * COS (ALFA) ) /ALOG (R/ALFA)
```

programrészben az ALFA * COS (ALFA) kifejezés független a ciklusváltozótól, 249-szer felesleges kerül kiszámításra. A javított programrész a következő:

```
      ALFA = .314579
      C = ALFA * COS (ALFA)
      DO 15 X = 1,250
      R = I
   15 Z (I) = C/ALOG (R/ALFA)
```

A hatékonyság nagymértékben javul, ha a ciklusváltozóitól független indexes változókat a ciklus elé visszük.

Pl. Az

```
      I = N +1
      DO 35 K = 1, L, 4
   35 Z (K) = (W (K) + P (I) ) * P (I)
```

programrész helyett hatékonyabb az

```
      I = N +1
      HELY = P (I)
      DO 35 K = 1, L, 4
   35 Z (K) = (W (K) + HELY) * HELY
```

programrész.

Jó hatásfok javító az azonos ciklusfejjel rendelkező ciklusok összevonása egy ciklussá.

Pl. Az

```
      S = 0.0
      DO 12 I = 1,500
   12 S = S + X (I)
      Z = 0.0
      DO 15 I = 1,500
   15 Z = Z + (X (I) – R) * * 2
```

programrész helyett hatékonyabb az

```
      S = 0.0
      Z = 0.0
      DO 13 I = 1,500
      S = S + X (I)
   13 Z = Z + (X (I) – R) * * 2
```

programrész.

Számítási időt takaríthatunk meg azzal is, ha módunkban áll csökkenteni a ciklus módosító és ellenőrző részének a végrehajtási számát. Ezt egyes esetben a gyorsabban végrehajtható utasítások számának növelésével érhetjük el.

Pl. A

```
      DO 17 I = 1,1600
   17 P (I) = 0.0
```

utasítások helyett a

```
      DO 17 I = 1,800
      P (I) = 0.0
   17 P (I+800) = 0.0
```

vagy

```
      DO 17 I = 1,1596,5
         P (I) = 0.0
         P (I+1) = 0.0
         P (I+2) = 0.0
         P (I+3) = 0.0
      17 P (I + 4) = 0.0
```

utasításokkal nyerhetünk időt.

Számos más lehetőség is rendelkezésünkre áll, a számítási idő és tárolási igény csökkentésére, azonban e cikk keretében nem térhettünk ki mindazok ismertetésére, amelyeket programozói tapasztalat alapján gyűjtöttünk, vagy amelyeket feldolgoztak cikkekben, könyvekben. Mindössze arra hívnám fel a figyelmet, hogy a speciális alakú adatstruktúrák (szimmetrikus mátrix, alsó ill, felső háromszög mátrix; több, azonos elemű sor ismétlésével létrehozott mátrixok sok zérus elemet tartalmazó ún. ritka mátrixok stb.) tárolására kidolgozott módszerek alkalmazásával lényeges tárolási megtakarítást érhetünk el.

Irodalom

USA Standard FORTRAN. ANSI Standard X 3.9–1966.
New York: American National Standards Instituts

Larson, C.: The Efficient Use of FORTRAN, Datamation
1971. VIII. 1. 24–31. oldal.

Lowra, Edward és C. W. Medlock: Object Code Optimazation,
CACM 12, No. 1. 1969. 13–22. oldal.

Allen, F. E.: Program Optimization, Annual Review in Automatic Programming, Volume 5.
(Szerk.: M. I. Halpern, C. J. Shaw) Pergamon Press, 1969.

RENDSZER–INFORMÁCIÓK FELHASZNÁLÁSA EGY HATÉKONY KÉTGÉPES ÜZEM KIALAKÍTÁSÁRA

Páldi Vince–Fóti György
KSH SZIG

A KSH Számítástechnikai Igazgatóságán üzemelő IBM gépek teljesítménye és kapacitása ma az országban működő gépek túlnyomó többségét meghaladja. Várható azonban, hogy egyre több számítógép lesz alkalmas részint OS, részint OS/WS típusú operációs rendszer alkalmazására. Várható továbbá az is, hogy sok számítóközpontban alakul ki olyan helyzet, hogy a régi gép még évekig az újjal együtt üzemel össze nem kapcsolt módon. Ugy gondoltuk, hogy a számítóközpontunkban megvalósított hatékonyság vizsgálati és javítási módszerek hasznosak lehetnek e felhasználók számára.

A konkrétumok előtt azonban le kell szögezni, hogy mi nem önmagában optimális operációs rendszert, nem dugig tömött gépet, nem minimalizált átfutási időt tekintettünk célnak, hanem azt, hogy a számítógépes rendszer a kiszolgáló személyzet és a végfelhasználók együttesen a meglévő erőforrások, a munka jellege és a szellemi kapacitás színvonala alapján optimális rendszert alkossanak. Nem riadtunk tehát attól vissza, hogy egy mérés az általa szolgáltatott hasznos információk mellett egyben az erőforrásokat is terheli, vagy attól, hogy bármilyen mértékű automatizálás gépidő felhasználással jár.

A számítógépek üzemének hatékonyságát software oldalról két, egymástól lényegesen különböző módszerrel tudjuk növelni. Az egyik lehetőség az erőforrások elosztásának felügyelete, a másik a számítógépben végbemenő folyamatok mérése, az eredmények alapján következtetések levonása, változtatások végrehajtása a rendszeren, és ismét mérés. Előadásunk első részében a SZIG-en alkalmazott mérési és eredményértékelési módszereket ismertetjük, a második részben pedig a mágneses adathordozókkal való gazdálkodás főbb eszközeit mutatjuk be.

Számítógépeink üzemének mérését két nagy csoportba oszthatjuk. Az első csoportba tartoznak azok az átfogó jellegű esemény rögzítések, amelyeket az operációs rendszer (OS/VS2 SVS Rel. 1.7) állandóan elvégez. A rögzítés az operációs rendszerhez tartozó állományokba történik. Ezek: az SMF, a LOG és a LOGREC. A mérések másik csoportját mindig a rendszer software felügyeletével megbízott személy vagy személyek kezdeményezik. Ezek a mérések – mivel általában nem fogják át időben a teljes üzemi tartományt – csak mintavételezésnek tekinthetők, így nagyon kell vigyázni a mérési intervallumok kiválasztására, hogy az az egész üzemeltetés vagy a vizsgált jelenség szempontjából megfelelő legyen. Az operációs rendszernek három ilyen célra használható eleme van a GTF, a TSOTRACE és a DSS. A hatékonyság mérésére (pontosabban a szűk keresztmetszet keresésére) szolgáló monitorprogramokat jelenleg csaknem mindegyik software-ház kínálja a vásárlóknak. Mi részletesebben az IBM VS2PT mérő és értékelő programjaival foglalkoztunk, így ezt mutatjuk majd be.

Térjünk vissza az operációs rendszer által rögzített események, adatok értékelésére. Az az eszköz, amelyet feltétlenül elsőnek kell megemlítenünk az SMF (System Management Facility). Az SMF többek között job és job step elszámolási információkat, állományokra és lemezekre vonatkozó adatokat, valamint az operációs rendszer és az aktív alrendszerek működését leíró rekordokat rögzíti. Mi az egyes jobokra és job stepekre vonatkozó adatokat naponta értékeljük. Célunk az, hogy megkeressük egyrészt a rendellenes szituációkat,

másrészt a túl sok erőforrást igénylő jobokat. Az előző napi SMF anyagokból minden elkészítünk két grafikont: az egyiken a CPU WAIT állapotának változása, a másikon a belapozás alakulása látható. Ez a két grafikon segít megtalálni az előző napnak azt a szűkszát (túl sok várakozás, túl nagy lapozás), ahol az egész rendszer hatékonysága lecsökkent. Így elegendő csak ennek a szűkebb anyagot átvizsgálni, de nem kell az operátorok sejtését sem hagyatkozni. Ilyen rendellenes állapotok jöhetnek létre például nagy terjedelmű táblázatoknak TPL programmal történő előállításakor (nagy lapozási érték) vagy nagy I/O igényű jobok párhuzamos futása esetén (CPU WAIT). Ha sikerül felismerni a rendellenes állapotot kiváltó okokat (közös okot), akkor ennek alapján javaslatot tehetünk vagy a feldolgozás gyorsításának vagy a szervezőknek arra, hogy hogyan lehet ezeket a szituációkat elkerülni.

Az SMF-ből nemcsak a rendszer egészének túlzott terhelésére vonatkozó adatokat használjuk fel, hanem leválasztjuk és megvizsgáljuk azokat a jobokat is, amelyek túl sok lépésben állnak, illetve egy vagy több stepjük valamelyik jellemzője egy általunk meghatározott kritikus érték fölé kerül. Ezek a stepre vonatkozó jellemzők a következők: a futási idő, a felhasznált CPU idő, a lapozás (start-stop idő hányados, a start-stop idő) CUP idő hányados, az egyes szalag és lemezhasználat.

Ezeknek az adatoknak alapján egyedi esetekben a szervezőknek teszünk javaslatot, szeres illetve másoknál is ismétlődő jelenségek esetén szabályozzuk ezeknek a joboknak adását az üzemeltetési kézikönyvben vagy a feldolgozásirányításon belül.

Az SMF-ből gyűjtött adatainkat a közeljövőben az egyes JOBQUEUE-k átbocsátóségének (sebességének) mérésével kívánjuk kiegészíteni.

Bár nem tartozik szorosan az előadás témájához, de úgy érezzük az SMF exit lehetőségeiről is szólnunk kell. Az SMF exitek a job feldolgozás minden főbb szakaszában meghatók, az exitekből adott hibajelzésekkel az egyes jobok futása megszakítható. Itt oldottuk meg a házi JCL szabályok ellenőrzését, a SETUP rendszer és a JCL TEST kapcsolatát, a kívánt típusú rekordok írásának letiltását, az IPL-kor a dátumot ellenőrző öröknaptár betését. Szerepet játszanak az exitek akkor is, ha az eredetileg a 155-ös gépre irányított jobot a kb. 3-szor lassabb 138-ason kívánjuk futtatni. Csak itt tudjuk ugyanis automatikusan változtatni a jobra és az egyes stepekre adott időkorlátot.

A másik, általunk is használt állandóan működő adatgyűjtés a LOG. Ez tartalmazza egyes jobok állapotát leíró információkat és az operátoroknak szóló üzeneteket. A LOG formációkat 30 napos intervallumban őrizzük meg és mágnesszalagon tároljuk.

A LOG információkat jelenleg két módon használjuk fel. Az egyik módszer egy adott intervallum eseményeinek kilistázása. A másik módszer segítségével partíció (init) térképet rajzolhatunk egy adott időintervallumra. Mindkét lehetőséget az SMF alapján leválasztott rendellenes időszakok további, részletesebb vizsgálatára használjuk fel.

A LOGREC rögzítés az előforduló hardware hibákat gyűjti. A rögzített rekordok naponta egy HISTORY szalagra kerülnek, ahol 30 napig elérhetők. A naponta történő események egy rövidített lista készül, amely bizonyos SMF információkhoz hasonlóan a kiemelt események feltárására szolgál. Miután job vagy rendszer rendellenességek összefüggnek hardware hibával, a rövidített lista, majd a HISTORY-szalag konkrét lekérdezése segít a változó okok felkutatásában.

Méréseink másik csoportjával egy *előre meghatározott* esemény vagy eseménysorozat hatását vizsgáljuk. Ilyen vizsgálandó folyamat lehet pl. egy új programtermék hatása a rendszerre vagy a TSO (Time Sharing Option) elindítása.

A GTF (Generalized Trace Facility) az operációs rendszerhez tartozó szervízprogram. Segítségével a rendszer szempontjából legfontosabb eseményekről gyűjthetünk adatokat. Start I/O, Supervisor call, task switching és a megszakítások kezelése. Ezen események egyikéből egy rekord készül. Az eredményeket egy másik szervízprogram segítségével ír-

ki. Az adatok szelektálását két módon végezhetjük el: egyrészt a gyűjtendő eseményeket határozhatjuk meg a GTF inputjaként, másrészt a már gyűjtött adatok kiíratása során a listázó programnak adhatunk a kiíratást vezérlő korlátokat (pl. eseménytípus, jobnév, időintervallum stb.).

A másik lehetőség a DSS (Dynamic Support System). Az adatgyűjtést itt nemcsak a különleges események, hanem pl. egy általunk meghatározott címre történő elérés is vezérelheti. Így nyomon lehet pl. követni (nem multiprogramozású környezetben), azt, hogy egy job az inicializálása során milyen rendszermodulokon megy keresztül. A figyelést DSS eljárások írásával automatikussá is lehet tenni, az adatokat pl. mágnesszalagra lehet rögzíteni.

A SZIG-en most vizsgáljuk a TSO TRACE lehetőségét. Ezzel a módszerrel a TSO környezetben lejátszódó események és a TSO Driver működése kísérhető figyelemmel. Az adatgyűjtés során rekordokat nyerhetünk a terminál I/O-k, a TSO SWAP-ek, a TIME SLICE-ok stb. számáról és a hatásukra létrejövő eseményekről.

A leghatékonyabb és egyben legátfogóbb méréseket a VS2PT (VS2 Performance Tool) programok segítségével végeztük. A méréshez előbb a monitor programot kell elindítani, ami az adatokat mágnesszalagra gyűjti. Az adatok értékelését egy külön listázó program végzi el. A végső adatok összegezve (ellentétben a GTF-fel, ahol egymás után) jelennek meg. Ezek különféle rendszerstatisztikákat (az egyes SVC-k gyakoriságát, a hívott programok nevét, gyakoriságát és elhelyezkedését, az inicializált stepek számát stb.) adják meg. A mérőprogram legértékesebb része szerintünk az, amelyik az egyes csatornák, ezen belül a lemezegységek használatát méri. Külön grafikonban szemlélteti a csatornák használatát és a várakozást, ugyancsak külön ábrázolja az egyes egységek kihasználtságát. Ezeknek a méréseknek alapján jutottunk például a rendszerállományok olyan elhelyezéséhez, amelyik az egyes egységeket kb. azonos mértékben terheli. A mérés során adatokat gyűjthetünk az egyes lemezek cilindereinek aktivitásáról is, így lemezen belüli állományátrendezéssel javíthatjuk a gyakran használt állományok elérését.

Meg kell még említeni az LPA-ról (Link Packed Area) térképet adó méréseinket, amelyek segítségével a gyakran és együtt használt modulokat azonos lapokra helyeztük el csökkentve ezzel a lapozások számát és gyorsítva a modulelérést.

Méréseink, illetve elemzéseink első, az operációs rendszer által gyűjtött adatokra támaszkodó csoportját napi rendszerességgel végrehajtjuk. Ezek segítségével általában a hatékonyságromlás *emberi* tényezői ugranak ki (nem teljesített lemezkérés, hibás szervezői vagy feldolgozási munka, rossz programozási stílus). A másik csoport az operációs rendszer software-nek a fizikai korlátokba való ütközését hivatott felderíteni, az ezekből levont következtetések pedig olyan változtatásokat hozhatnak, mintha gépünk „megnőtt" volna, pedig csak egyes software elemek státuszát változtattuk meg. A két vizsgálati rendszer között még egy nagy különbség van: míg az első rendszer elemei az operációs rendszer szerves és állandóan aktív részei, addig a második csoportba tartozó mérések bármelyikének megindítása egy plusz teher és erőforrásigény a rendszer számára, ami ily módon nemcsak a hatékonyságot rontja, ha nem is jelentősen, de megváltoztatja a mérés környezetét is.

Az operációs rendszer által gyűjtött és szolgáltatott információkat valamint a rendszerben tárolt és folyamatosan aktualizált adat-táblázatokat eddig nem említett célra is felhasználjuk.

A SZIG feldolgozási folyamatában a szűk keresztmetszet egyértelműen a külső adattárakkal, a mágneses adathordozókkal kapcsolatos.

A fő problémák:

– a statisztikai feldolgozások nagy adatigénye miatt az egyidőben futó 6–7 job szá-

mára kevés a 240 Mbyte-os on-line felhasználói lemezkapacitás (naponba kb. 100 lemezcs re van)

– kb. 50 db 300 Mbyte-os lemezcsomag tartalmát kell folyamatosan karbantartani (lés, hely-felszabadítás, nyilvántartás)

– kb. 10 ezer mágnesszalag tartalmi és fizikai kezelését kell ellátni, miközben napon mintegy 500 új szalagos állomány készül.

A felsorolt problémák megoldására több rendszert fejlesztettünk ki, melyek működé hez a fentiekben említett operációs rendszer hatáskörébe tartozó információkat használjuk

Ezek:

– SMF (System Management Facility) bizonyos rekordjai
– rendszerkatalógus (SYSCTLG)
– lemeztartalomjegyzék (VTOC)

Az SMF rekordjai között 4 típus van, amelyek a felhasználói jobok input/output te kenységére vonatkoznak: A 14-es rekord minden állomány inputként, a 15-ös minden állo mány outputként való megnyitásakor íródik. A 19-es egy lemezcsomag levételekor, a 21-es pedig I/O hiba bekövetkezésekor íródik az SMF állományba. Több más rekordtípussal egy ezeket a rekordokat gyűjtjük, és egyrészt az élő SMF állományból dinamikus módon, má részt a napi vagy heti kummulált adatokból statikus módon feldolgozzuk.

A rendszerkatalógusnak ugyancsak nagy a jelentősége. Rendszerünkben minden állo mányt kötelezően katalogizálni kell, amely a job control nyelv segítségével rendkívül egys rűen megoldható. A katalógus az állomány nevét, és az azt tároló adathordozó típusát és sorszámát tartalmazza. Mindig az aktuális állapotot tükrözi, így feldolgozása dinamikusan ténik. Az ismertetésre kerülő adathordozó kezelő- és nyilvántartó rendszeren kívül szerepe Automatizált adathordozó előkészítő rendszerben igen nagy jelentőségű (Másik előadás tá gyalja).

A lemeztartalomjegyzék információi közül számunkra a lefoglalt hely és a megőrzé idő a legfontosabb. A katalogizáltsághoz hasonlóan minden állományra kötelező a megőrz idő megadása, és a lemezes állomány esetén méretkorlát betartása. A megoldható időt és méretet típusokba soroltuk, és az értékeket szabályoztuk. Az érdekek dinamikus lekérde révén történhet a lefoglalt helyek felszabadítása.

A mágnesszalag nyilvántartó rendszer (MANYI)

Célja:

– a gépteremben állandóan megfelelő mennyiségű felülírható mágnesszalag legyen
– lehetőség legyen a szalagok folyamatos karbantartására
– felhasználók ne szalagot, csak állományokat ismerjenek
– a mágnesszalagok, illetve a rajtuk tárolt állományok információi hozzáférhetők gyenek.

A rendszer központi állománya minden szalagról egy 200 byte-os rekordot tárol, ben a szalagra vonatkozó összes lényeges információ megtalálható.

Pl.:

– a szalag inicializálásának időpontja, az előfordult I/O hibák száma
– a szalagon tárolt állomány neve, létrehozásának és lejáratásának időpontja, az á mány jellemzői, katalogizáltsága, az állomány szalagon elfoglalt hossza stb.

A rekordok kulcsa a szalag sorszáma. A rekordokat az egy héten kétszer futó MA program aktualizálja, mégpedig az előző aktualizálás óta készült 15-ös és 21-es SMF re dok, valamint az aktuális katalógus alapján.

A 15-ös rekord tartalmazza az újonnan létesített szalagok minden fontos információját, ezeket egészíti ki a 21-es rekord az esetleg előfordult I/O hibák számával, a rendszerkatalógus pedig azzal az információval, hogy az elkészült állomány katalogizálva lett-e. (Előfordulhat, hogy temporális állományt írtak a szalagra, vagy az állomány hibás futás miatt nem katalogizálódott).

Az aktualizálás során a program megvizsgálja az összes rekordot, és az alábbi funkciókat végzi el:

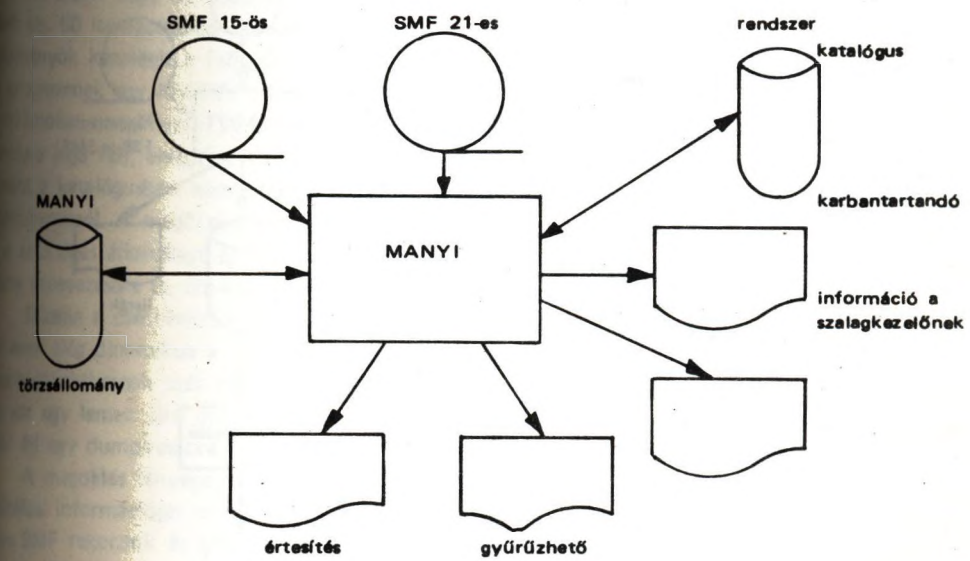
– értesítési listát küld a 10 nap múlva lejáró állományokról

– kiveszi a katalógusból, és gyűrűzhető listára (felülírható szalag) írja a 10 napra értesített állományokat tartalmazó szalagokat

– gyűrűzható listára írja a katalógusban nem szereplő vagy érvénytelen megőrzési idővel rendelkező állományokat tartalmazó szalagokat

– karbantartandó listára írja azokat a szalagokat, melyeken permanent I/O hiba fordult elő, vagy amelyek inicializálása óta 400 nap eltelt.

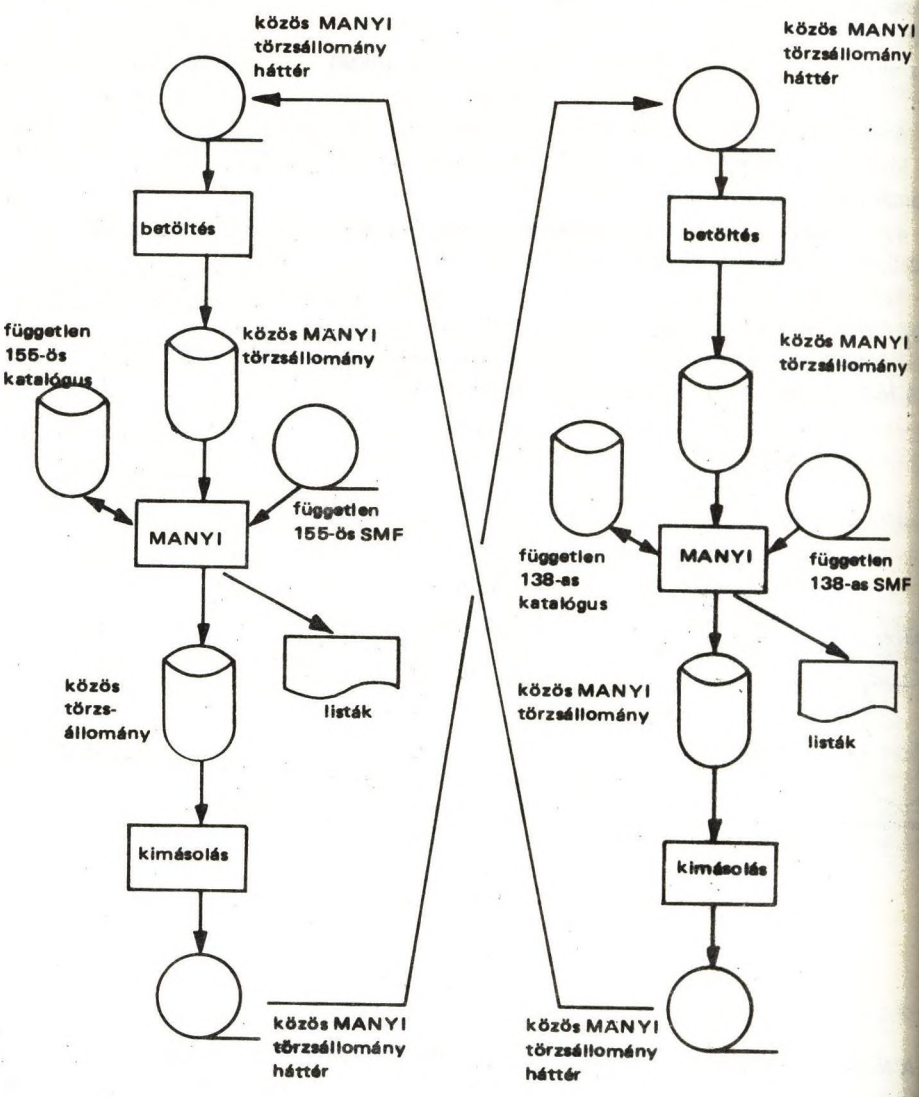
– listára írja a nyilvántartásba nem tartozó szalagokra történő írást (külső szalagok), valamint a MANYI állomány és a katalógus esetleges eltéréseit.



A MANYI rendszer törzsállományában tárolt információk szalagszám és állománynév szerint ad-hoc módon és rendszeresen lekérdezhetők. Ilyen információk például az állomány jellemzői, melyeket a felhasználó főleg dokumentációs célból írat ki, de készülnek rendszeres kimutatások a szalagok kihasználtságáról, a megőrzési időre vonatkozó szabályok betartásáról, az egy osztály vagy munkaszám által adott idő alatt felhasznált szalagok számáról stb.

A mágnesszalag nyilvántartó rendszer mindkét gépen fut, mégpedig közös törzsállományból, de külön SMF és katalógus adatok alapján.

Így a szalagállomány fizikai szinten közös, de az elkészülő állományok kezelése aszerint történik, hogy melyik gépen jöttek létre.



Lemezkezelés

A lemezkezelés rendszerünkben jelenleg több önálló részből áll, és célját, megvalósítá tekintbe többirányú.

A lemezcserék optimálisabbá tétele miatt a használt lemezes állományokat három csoportba soroltuk:

– rövid ideig szükséges teszt vagy TSO állományok
– több felhasználó által rendszeresen használt korlátozott méretű állományok
– periódikusan használt állományok mérettől függetlenül, de felső méthatár definiálásával.

A három kategóriához a gyakorlatban két különböző attributumú és kezelésű lemezes rendszert biztosítottunk. 8 db 30 Mbyte-os lemezünk közül 4-et fix attributummal láttunk el. E lemezek nem cserélhetők, a rendszerben állandóan on-line vannak. E lemezekre kerülnek az első két kategória állományai. A létrehozott állományoknak 0—15 nap megőrzési időt lehet adni, a 15-nél nagyobb helyére a rendszer automatikusan 15 napot tesz. A 15 napos megőrzési idővel rendelkező állomány a rendszerben kitüntetett. Felhasználása esetén (OPEN) egy rutin a megőrzési időt automatikusan átállítja a napi dátum +15-re, így az ilyen állomány mindaddig lemezen marad, míg használatában 15 napnál nagyobb szünet be nem következik. A 15 napnál kisebb megőrzési idő OPEN esetén sem változik, hanem megtartja a job-control-ból kapott értéket.

Egy minden reggel futó technikai program törli és a katalógusból kiveszi a lejárt megőrzési idejű állományokat, és törli a katalógusban nem szereplő, illetve megőrzési idő nélküli (temporális) állományokat. A rendszer rendkívül dinamikusan biztosítja az állandó szabad helyet.

A sok törlés miatt kialakuló „lyukak" eltüntetésére kifejlesztés alatt áll egy lemez sűrítő program, amely reggelenként a törlő program futását követően fogja a sűrítést elvégezni.

A másik négy 30 Mbyte-os lemez cserélhető attributumú, azaz a jelenleg használatban lévő kb. 50 lemezcsomag számára 4 egység áll egyidőben rendelkezésre. E lemezeken tárolt állományok kezelését a felhasználó által adott megőrzési idő alapján a DISCO program végzi. A programnak egy kívülről változtatható táblázata van, amely a hatáskörébe tartozó lemezsorszámokat tartalmazza. A program futásakor végignézi az éppen fentlévő lemezeket, és amelyre joga van, elvégzi a karbantartást. Törli a lejárt vagy érvénytelen megőrzési idejű, valamint a katalógusban nem szereplő állományokat, és értesítést küld a 20 napon belül lejáró állományokról. A megőrzési idő meghosszabbítását egy külön program végzi, amely megnyitja a szükséges állományt és átírja a megőrzési idejét. Ezt a programot lehet használni az állomány átnevezésére is. On-line lemezeken a program nem írja át a megőrzési időt.

Miután a cserélhető lemezen tárolt állományok száma egyre nő (a megőrzési idő alapján nem elég dinamikus a hely felszabadítás), az egy anyaghoz tartozó állományok több lemezre szóródhatnak szét (az éppen felszabaduló szabad helyek szerint), valamint ritkán kerülhet sor egy lemez újre inicializálására (egyszerre egy lemez soha nem szabadul fel) befejezés előtt áll egy dump/restore rendszer kialakítása, amely megoldást ad a fenti problémákra.

A megoldás lényege, hogy egy szótár tartalmazza a lemezre létesített állományok minden allokálási információját munkaszámos csoportosításban. (A szükséges információkat a 14, 15, 19-es SMF rekordok és a lemez VTOC-k adják). A feldolgozások átfutási idejének ismeretében lehetővé válik a használatban nem lévő állományok szalagra dumpolása, és az éppen szükséges állományok visszatöltése. A dumpolás és visszatöltés esetén az összetartozó, illetőleg egy időben szükséges állományok azonos lemezre kerülhetnek. A rendszer a mágnesszalagkezeléshez hasonlóan biztosítja az állományok eszközfüggetlenségét és az állandó szabad területet.

Elszámolás

A rendszer által szolgáltatott információkról eddig úgy beszéltünk, mint az operációs rendszer hatékonyságát javító, és a szűk keresztmetszetek hatékony lekezelését végző alapadatok halmazát. További felhasználása is lehetséges azonban azon az alapon, hogy objektív képet ad a gépe(ke)n folyó munkáról. Ilymódon célszerűen felhasználható a munkák ellenőrzésére, és a munkákat futtató felhasználók orientálására. Miután számítóközpontunkban az elvégzett munka után nincs számlázás, az elszámolás nem nyereség orientált, lehetőség van olyan objektív elszámolásra, amelynek célja kimutatni egy feldolgozás „árát" a szűk keresztmetszetek és az átbocsátóképesség oldaláról. Az effajta SMF adatokra épülő elszámolási rendszer kialakítása most van folyamatban.

A SZÁMÍTÓGÉPES NYOMTATÁS TECHNOLÓGIÁJA ÉS ANNAK HAZAI VONATKOZÁSAI

Rákóczi Ferenc
VIDEOTON FEJLESZTÉSI INTÉZET

1. Bevezetés

A számítógépes információ-feldolgozó rendszerek egyik legfontosabb adatkiviteli perif riája a nyomtató, amely a feldolgozás eredményét az ember számára érthető, olvasható form közli, és arról többé-kevésbé tartós bizonylatot ad. Míg egy-két évtizeddel ezelőtt a nyomtat nak néhány alaptípusa volt csupán elterjedve, az utóbbi években a miniszámítógépek gyors f désének, majd a mikroszámítógépek elterjedésének eredményeként a nyomtatók választéka rohamosan bővült, a teljesítmény- és árszintek lefelé és felfelé egyaránt nagymértékben szétn zódtak. A mini- és mikroszámítógépek alkalmazása olcsóbb nyomtatókat igényel, a kötegel távadatfeldolgozáshoz mérsékelt áru, közepes teljesítményű nyomtatókra, az úgynevezett „klasszikus" számítógéptermi tevékenységhez pedig rendkívül nagyteljesítményű, s ennek m lően költségesebb nyomtatókra van szükség. A számítástechnikai ipar a felhasználói igény kielégítésére törekedve számos különféle fizikai elven alapuló nyomtatási technológiát, s ezek alapján működő nyomtatót fejlesztett ki, amelyek mindegyikének az a célja, hogy az inform ciót papíron rögzítse. Adott rendszerben az alkalmas típus kiválasztásához számos tényezőt figyelembe venni, például a nyomtatás minőségével szemben támasztott követelményeket, gr kus ábrázolás igényét, a nyomtatott anyag várható mennyiségét, a karakterkészlet rugalmas toztatásának lehetőségeit, a fogyó anyagok — papír, festékszalag stb. — árát, s nem utolsó s a rendszer megbízhatóságát és karbantartási költségeit.

A nyomtatókat, amelyek teljesítménye 10 jel/mp-től néhányszor 10 000 sor/perc-ig, é pedig 1000 dollártól néhány 100 000 dollárig terjedhet, felosztható a nyomtatás módja, m ködési elve, vagy a jelalak megformálásának módja szerint; a felosztások természetesen átfe egymást.

2. A nyomtatók felosztása

A nyomtatás módja szerint lehetnek

a) Karakternyomtatók, vagy soros nyomtatók

b) Sornyomtatók, vagy párhuzamos nyomtatók

A karakternyomtatók a sort karakterenként nyomtatják ki, a sornyomtatók egy cikl egy teljes sort nyomtatnak.

A működés fizikai elve alapján vannak

a) Ütőnyomtatók

b) Egyéb fizikai elven működő, nem ütő nyomtatók.

A jelalak megformálásának módja szerint vannak

a) Jelalaknyomtatók

b) Mátrixnyomtatók

A jelalaknyomtatók a nyomdai betűkhöz hasonlóan megformált összefüggő jelalako tatnak; a mátrixnyomtató a kívánt jelformált pontokból, vagy vonalakból rakja össze.

A továbbiakban a nyomtatók jellegzetes típusait a nyomtatás fizikai elve szerint fel ismertetjük, s ezen belül tárgyaljuk a soros és párhuzamos jelalak- és mátrixnyomtatókat.

3. Nyomtató típusok

3.1. Ütőnyomtatók

A számítástechnikában jelenleg alkalmazott karakter- és sornyomtatók túlnyomó többsége ütőnyomtató, ami annyit jelent, hogy a festékanyag mechanikai erő hatására adódik át a papírra. Előnye, hogy közönséges papíranyagra nyomtat, és több példány átütésére alkalmas, tehát több másolatot készít. Hátránya a dinamikus erőhatással szükségszerűen velejáró zaj, valamint a mechnikus szerkezetekkel együttjáró súrlódás és kopás. Ennek ellenére várható, hogy a jövőben is az ütőnyomtató típusok uralják a piacot, s az egyéb fizikai elven működő nyomtatók alkalmazása speciális területekre szorítkozik.

Az ütőnyomtatás elve alapjában egyszerű: a karakterhordozón kialakított kidomborodó karakter, és a vele szemben álló kalapács között helyezkedik el a festékszalag és a papírlap. A karakter és a kalapács közé szorított festékszalagról a festékanyag a mechanikai erő hatására átadódik a papírra, s azon a karakter alakjának megfelelő nyomot hagy. A szükséges erőhatás a mozgó tömegek ütközésekor impulzusszerűen jön létre. Az ütőnyomtatás kétféle módon valósítható meg: vagy a karakter üt rá a festékszalagra és a papírra, úgy mint a közismert írógépben, vagy a kalapács üt rá hátulról a papírra, s nyomja azt neki a festékszalagon keresztül a karakternek. Az előbbi módon működnek a betűkaros és betűfejes villamos írógépek és karakternyomtatók, valamint a mátrixnyomtatók, az utóbbi elven pedig szinte mindegyik elektromechanikus sornyomtató.

Az alábbiakban az ütőnyomtatók jellegzetes típusait ismertetjük.

3.1.1. Karakternyomtatók (soros nyomtatók)

A karakternyomtatók az információt karakterenként nyomtatják ki, sorban egymás után. Egy karakter kinyomtatása után a karakterhordozó egy betűosztással tovább lép, a kijelölt karakter a nyomtatási pozícióba áll, s kinyomtatja a következő karaktert. A teljes sor kinyomtatása után a karakterhordozó visszaáll a sor elejére, s a papír egy sortávolsággal tovább lép. E karakternyomtatók teljesítménye 1—100 karakter/s tartományban van, ára — teljesítménytől függően — 1–7 ezer dollár körül van. A karakterhordozó lehet a sor irányával párhuzamos, vagy arra merőleges tengelyű henger, gömb, küllős tárcsa stb.

A legelterjedtebb típus a 10 karakter/s teljesítményű Teletype; közismert az IBM Selectric gömbfejes karakternyomtató, amely 15 karakter/s sebességgel működik, s nyomtatási minősége az üzleti levelezés követelményeit is kielégíti. A gömbfej könnyen cserélhető, így a karakterkészlet és a betűtípus kívánság szerint változtatható. A Xerox—Diablo és Qume nyomtatókban a „százszorszép"-nek nevezett külső tárcsa a karakterhordozó, 50—60 karakter/s teljesítménnyel nyomtat.

Ezeket a nyomtatókat távközlési célokra, üzleti levelek előállítására, konzolírógépként és terminálokhoz használják.

3.1.2. Sornyomtatók (párhuzamos nyomtatók)

A közepes- és nagyteljesítményű nyomtatók túlnyomó többsége ebbe a csoportba tartozik. E sornyomtatók teljesítménye 100—3000 sor/perc; ára 3000—100 000 dollár határok között van. A technológiai fejlődés tette lehetővé, hogy a nyomtatási sebesség 1960 óta 600 sor/perc-ről a jelenlegi 2000—3000 sor/perc-et elérje, ugyanakkor nőtt e nyomtatók megbízhatósága, és csökkent az ára. A 2000—3000 sor/perc nyomtatási sebesség elérése szinte mechanikai csúcs-

teljesítmény; ennek jelentős növekedésére már nem számíthatunk, sőt, a nyomtatási minőséggel szemben támasztott követelmények – optikai jelalakolvasás – miatt valószínűnek látszik, hogy az ütő elven működő sornyomtatók teljesítménye többségében 1500 sor/perc alatt marad, s nagyobb teljesítmény igénye esetén az egyéb fizikai elven működő nyomtatók alkalmazása kerül előtérbe.

A sornyomtatókban egy teljes sor kinyomtatása egy ciklusban megy végbe. A nyomtató mechanizmusban minden egyes nyomtatási pozícióhoz tartozik egy-egy kalapács s a nyomtatható karakterek a mozgó karakterhordozón e kalapácssor előtt futnak el. A megfelelően időzített kalapács abban a pillanatban üti neki a papírt és a festékszalagot a karakterhordozónak, amikor a kívánt karakter a nyomtatási pozícióban van. A karakterhordozó mozgása folyamatos, a kalapács a nyomtatási pozíción átfutó, nagy sebességgel mozgó karakterre üt rá, ezért szokás ezt a típust „repülőnyomtatónak" is nevezni. A dolog természetéből következik, hogy az ütközés rendkívül rövid időtartama (30–35 μs) alatt is elmozdul a karakterhordozó, s emiatt a nyomtatott karakter kissé elkenődik. Ezt a jelenséget a karakterek alakjának tervezésénél természetesen figyelembe szik, ennek ellenére a nyomtatási minőség a karakter mozgásának sebességével fordítottan arán A nyomtatott sorok – vagy a karakterek közötti osztástávolságok – egyenletessége pedig a ka csok pontos időzítésétől függ.

A leggyakoribb karakterhordozó a karakterdob. A dob tengelye a sor irányával párhuzamos, s a karakterek a henger alkotói mentén soronként vannak elrendezve. Egy-egy sorban a nyomtatott oszlopszámnak megfelelő számú, azonos karakter van (rendszerint 80, 128, 132, 136). A karakterkészlet a sornyomtatóknál általában 64 vagy 96 karaktert tartalmaz, ennek megfelelően a henger palástján 64 vagy 96 sorban helyezkednek el a karakterek. A dob forgása közben a karakterek a sor irányára merőlegesen mozognak a kalapácssor előtt, a kalapácsok pontos időzítése tehát jelentősen befolyásolja a nyomtatott sor egyenességét.

E probléma elkerülhető, ha a karakterhordozó a sorok irányával párhuzamosan mozog. Ez a karakterhordozó lehet lánc, szíj, vagy acélszalag. A karakterlánc alkalmazásában úttörő a Az acélszalag karakterhordozó az utóbbi évek újdonsága. A karakterláncok és szalagok előnye, hogy súlyuk lényegesen kisebb, mint a karakterdoboké, könnyen cserélhetők, a karakterkészlet bővebb lehet, mint a karakterdobnál. Ezzel szemben a karakterdob tartósabb, s egyszerű csapágyazásával megbízhatóbb, mint a bonyolultabb hajtó mechanizmust igénylő, és súrlódó, kopó elemeket tartalmazó lánc vagy szalag.

Lényeges szerkezeti eleme a sornyomtatónak a nyomtatás műveletét végző kalapács. A több sornyomtató típusban elektromágneseket alkalmaznak, amelyek tolórudak közvetítésével viszik át a mozgást a csuklósan vagy rugalmas elemekre felfüggesztett kalapácsra. Az ilyen elektromágneses mechanizmusok mozgó alkatrészei súrlódásnak és kopásnak vannak kitéve, a szenyeződés a súrlódás mértékét befolyásolja, ami a gondosan beállított időzítésekre káros hatással van. Mindezeket a hiányosságokat kiküszöböli a Dataproducts cég MARK IV kalapácsrendszere, amelynek egyetlen mozgó alkatrésze laprugókra felfüggesztve, súrlódás- és kopásmentesen működik, s élettartama a gyakorlatban az 500 millió leütés tervezési célkitűzését többszörösen meghaladja. Licenc alapján ezt a kalapácsot alkalmazza a VIDEOTON Számítástechnikai Gyár is hazai gyártású sornyomtatóiban. Ez a kalapácstípus megbízhatóbb, és kevesebb karbantartást igényel, mint az egyéb konstrukciók.

* A karakterek sorirányú eltolódására az emberi szem kevésbé érzékeny.

3.1.3. Mátrixnyomtatók

Az ütő eleven működő mátrixnyomtatók a kívánt jelformát pontokból állítják össze, a pontokat elektromágnesekkel mozgatott tűk nyomatják festékszalag közvetítésével a papírra. Az ütő eleven működő mátrixnyomtatók általában 5x7–7x9 pontból rakják össze a karaktereket. Nagy előnye e nyomtató típusnak, hogy ugyanaz a nyomtató fej a karaktergenerátor ROM tartalmától függően sokféle karakterkészlet nyomtatására alkalmas, s a ROM cseréjével japán katakana, hindi, arab írásjelek is nyomtathatók.

Mátrixnyomtatókkal 30–600 karakter/s, vagy 120–500 sor/perc nyomtatási teljesítmény érhető el, áruk 2–10 ezer dollár körül van.

Mátrixnyomtatók lényegében kétféle módon működnek. A karakternyomtatóban a nyomófej függőleges elrendezésben tartalmaz egy sor tűt, egyidejűleg egy függőleges pontsort nyomtat ki, s miközben a sor mentén végigfut, kialakul egy teljes sor írásképe. A karakternyomtató és a sornyomtató közötti különbség itt kissé elmosódik, mert bár vannak típusok, amelyek a karaktereket egymás után egyenként is képesek leírni, például billentyűzetről vezérelve, többségükben a fej folytonos mozgással fut végig a soron, s teljes sort nyomtat ki. Működése gyorsítható, ha a fej mindkét irányú mozgása közben nyomtat. Valódi sornyomtatók azok a típusok, amelyekben a pontokat kinyomtató szerkezeti elemek vízszintes sorban helyezkednek el, s például minden karakterhez egy tű tartozik. A tűsort tartó szerkezeti elem vízszintes elmozdulása közben mindegyik tű kipontozza egy-egy karakter legfelső pontsorát. A papír egy függőleges pontozásnak megfelelő távolsággal tovább lép, s a tűsor – visszafelé mozogva – a karakterek második pontsorát üti ki. A papír 7–9 függőleges léptetése után kialakult a teljes sor írásképe. Ilyen elven működik például a R Tally, Okidata és a Printronix nyomtató.

A mátrixnyomtatók alkalmazása az utóbbi években fokozódott, mivel a 30 karakter/s teljesítményű jelalaknyomtatók és a 100–300 sor/perc teljesítményű lassú sornyomtatók közötti tartományban a mátrixnyomtatók olcsóbbak, mint a jelalaknyomtatók. Bár a 300–100 karakter/s teljesítményű mátrixnyomtató drágább, mint egy Teletype Model 33, legalább háromszoros sebessége és jóval nagyobb megbízhatósága miatt mégis versenytársa lett. A 60–120 sor/perc teljesítményű mátrixnyomtatók alacsonyabb árfekvésükkel versenytársai a 100–300 sor/perc teljesítményű sornyomtatóknak, bár megbízhatóságuk kisebb, és több karbantartást igényelnek.

3.2. Különöeges nyomtatók

E címszó alatt a nem ütő elven működő nyomtatókkal foglalkozunk. Az ütőnyomtatók hátrányainak – mozgó mechanizmus, rajos működés, korlátozott nyomtatási sebesség – kiküszöbölésére az ipar különféle kémiai és fizikai jelenségeken alapuló nyomtatókat fejlesztett ki, amelyek némelyike közönséges papírra nyomtat, némelyike különleges papíranyagot igényel, közös tulajdonságuk azonban az, hogy a jelformát pontmátrixból állítják össze, és az ütőnyomtatókkal ellentétben egyidőben csak egy eredeti példányt állítanak elő, több példányos nyomtatára nem alkalmasak.

A technológia és a mikroelektronika fejlődése azonban lehetővé tette a karakterképet kialakító pontok sűrűségének növelését. Az ütő elven működő mátrixnyomtatók 5x7 pontból álló karaktereivel szemben például az IBM 46/40 tintasugaras nyomtatója 24x40x960 pontból rakja össze a jelformát, s ennek minősége a nyomdai betűk minőségével vetekszik.

Az alábbiakban a különleges nyomtatók néhány jellemző típusát ismertetjük, természetesen a teljesség igénye nélkül.

3.2.1. Hőnyomtatók

A hőnyomtató feje 5x7 pontot tartalmaz, amelyek hőhatása a különleges papíron nyomot hagy. Előnye olcsó ára és zajtalan működése. Mivel a papír ára a közönséges papírénak többszöröse, alkalmazása akkor indokolt, ha a nyomtatás volumene csekély. Bár 100 karakter/s sebesség is elérhető, általában 30 karakter/s sebességgel működnek, áruk 1000—6000 dollár körül van.

3.2.2. Elektrolitikus és elektrografikus nyomtatók

Mindkét technológia különleges papíranyagot igényel. Az elektrolitikus eljárás nedves, az elektrografikus száraz technológiai folyamat. Az ilyen működő mátrix-karakter nyomtatók függőleges sorban kilenc tűt tartalmazó feje végigfut a soron, s a tűre kapcsolt feszültség hatására elektrokémiai úton, vagy a papírra felhordott felső vékony réteget leégetve pontszerű nyomokat hagy a papíron. Ezzel a technológiával 250 karakter/s körüli sebesség érhető el viszonylag olcsó néhány száz és néhány ezer dollár közötti áron. Hátránya a nyomtatott szöveg nem egészen kifogástalan külalakja, és a különleges papír ára.

3.2.3. Elektrosztatikus nyomtatók

Különleges papíranyaggal működő mátrix sornyomtatók. A vízszintesen elrendezett tűsor megfelelően vezérelve pontszerűen töltést visz fel a papírra; az előhívó fürdőben az elektromos töltésű pontok festékanyagot vesznek fel. Az elektrosztatikus nyomtatók többségükben 300—3600 sor/perc teljesítménnyel dolgoznak, és 5000—13000 dollár körül van az áruk. Kiemelkedő teljesítményű a Honeywell 18000 sor/perc teljesítményű elektrosztatikus nyomtatója, 50 000 dollár körüli áron.

Rendkívül nagy előnye az elektrosztatikus sornyomtatónak az, hogy igen jó felbontás érhető el vele: az írótűk sűrűsége 200—220 pont/hüvelyk — 8 pont/mm — felbontást tesz lehetővé, az írás minősége tehát nagyon jó, és grafikus ábrázolásra is lehetőséget nyújt. Egyedüli hátránya a magas papírköltség.

3.2.4. Mágneses nyomtató

Inkább csak technikai érdekesség, a piacon szinte egyedülálló a Data Dynamics mágneses nyomtatója. Az írófej a felette elfutó mágnesszalagra 10x12 pontmátrix formájában mágneses pontokkal állítja elő a kívánt karaktereket; szalag felmágnesezett pontjai festékanyagot vesznek fel, majd a papír fölé álló mágnesszalagot a papírra szorítva az írásképet a papírra nyomtatja. 120 és 240 karakter/s teljesítményű típusai cirill, héber, japán írásjelek nyomtatására is alkalmasak.

3.2.5. Tintasugaras nyomtatók

A tintasugaras nyomtatók a karakterképet ugyancsak pontokból állítják össze, a papír kívánt pontjába parányi tintacseppet lőnek ki.

Kétféle típusa ismeretes. Az egyik az ütő mátrixnyomtatóhoz hasonlóan olyan nyomtatófejjel működik, amelyben egy függőleges fecskendősor van, s a tintacseppek elektronikusan vezérelve, piezoelektromos kristályok segítségével lövi ki; a fej a sor irányában fut.

A másik típus folyamatosan lövi ki a tintacseppeket; a villamos téren áthaladó cseppecskék villamos töltést vesznek fel, majd eltérítőlemezek között áthaladva a tintacseppet a lemezekre kapcsolt feszültséggel arányosan az elektrosztatikus tér eltéríti, a katódsugárcsőhöz hasonló módon.

E típusok teljesítménye és nyomtatási minősége tág határok között változik. Az IBM 92 karakter/s sebességgelnyomtat nyomdai minőséggel; a Mead Dijit nyomtató 45 000 sor/perc teljesítménnyel rendkívül nagy mennyiségű információ, pl. üzleti levél kinyomtatására alkalmas.

Előnye a típusnak, hogy közönséges papírra nyomtat. Hátránya, hogy igen kényes a tinta minőségére. Valószínű, hogy a közeljövőben több típusa jelenik meg a piacon. Ára jelenleg néhány ezer és néhány tízezer dollár között van.

3.2.6. Xerografikus nyomtatók

Ezek a nyomtatók a közismert xerografikus másoló eljáráshoz hasonló elven működik, a különbség a képalkotás módjában van. Az 1974-ben megjelent Xerox 1200 a teljes betűképet vetíti rá felvillanó fénnyel a xerografikus nyomóhengerre, az 1976-ban megjelent IBM 3800 típus lézersugaras karaktergenerátorral pontmátrix formájában állítja elő a karaktereket a fotovezető felületen. A karakterkészlet tekintetében ez utóbbi lényegesen rugalmasabb. A fotovezető hengerfelületére festékanyagot juttatnak, azt átnyomatják a papírra, majd hőhatással rögzítik.

A Xerox nyomtató lapokra nyomtat 60 lap/perc, vagyis 4000 sor/perc teljesítménnyel, az IBM nyomtató két szélén perforált papírra 10—14 ezer sor/perc teljesítménnyel. E nyomtatók ára 125 000—310 000 dollár.

E nagyon drága nyomtatók nagy teljesítménye csak akkor használható ki, ha havonta legalább 1,5 millió lapot kell kinyomtatni. Számítógépekhez akkor jöhet számításba, ha legalább három nagyteljesítményű elektromechanikus sornyomtatóra lenne szükség.

4. Hazai eredmények

Az előzőekből kitűnik, hogy a számítástechnikában a — viszonylag nem nagyon távoli — jövőben a piacot továbbra is az elektromechanikus ütő nyomtatók uralják, a legnagyobb darabszámban ezek értékesítése várható. Különösen érvényes ez a megállapítás hazánkra és a környező országokra. A különleges nyomtatási technológiával működő, igen nagy teljesítményű nyomtatók alkalmazása csupán nagy teljesítményű számítógép rendszerekben jöhet számításba. A piaci értékesítési lehetőségeket felmérve a VIDEOTON Számítástechnikai Gyár úgy döntött, hogy saját gyártású kisszámítógép-rendszereiben, és OEM értékesítésre közepes teljesítményű karakterdobos ütő sornyomtató gyártását kezdi meg. Mivel az új gyártmány bevezetésére sürgős igény volt, a várható fejlesztési idő okozta késedelem elkerülésére a VIDEOTON megvásárolta a Dataproducts 2310 és 2410 típusú nyomtatók gyártási jogát, s ezzel korszerű technológiát vezetett be sornyomtatók gyártására. A 300 sor/perc körüli teljesítményű sornyomtatók gyártása az 1970-es évek elején kezdődött. Ezeket a sornyomtatókat EC 7184 kódszámmal ESZR rendszerekben is alkalmazzák, és a Számítástechnikai Gyár egyik legjobban értékesíthető terméke lett.

A licenc keretében gyártott típusok hazai fejlesztésének eredményeként jelentek meg e sornyomtató típus 600 sor/per, 900 sor/perc, 1200 sor/perc teljesítményű változatai.

Folyamatban van a jelenleg gyártott sornyomtató típus továbbfejlesztése is, melynek eredményeként e karakterdobos sornyomtatók várhatóan 1980-ban korszerűbb formatervi kialakítással, kisebb méretekkel, s a felhasználó számára nyújtott több szolgáltatással — pl. diagnisztikai display — kerülnek értékesítésre.

Ezekben a sornyomtatókban a korábban már említett, nagy megbízhatóságú és hosszú élettartamú MARK IV kalapácselemek vannak felhasználva.

A mini- és mikroszámítógépek, valamint a terminálok várható nagyarányú elterjedése olcsóbb és rugalmasan felhasználható nyomtatók iránt fokozott igényt támaszt. Ezt az igényt kívánja kielégíteni a VIDEOTON Számítástechnikai Gyár 300 sor/perc teljesítményű, acél karteszalagos sornyomtató gyártásának bevezetésével.

A jövő igényeit hivatott kielégíteni a Számítástechnikai és Automatizálási Intézetben folyó kutatómunka, amely lézersugaras xerografikus nyomtató fejlesztése terén ért el értékes eredményeket. A nyomtatóban hazai gyártású lézert használnak fel, s a kutatómunka eredményeként jelentős szabadalmak is születtek.

A hazai felhasználók mátrixnyomtatót a Német Demokratikus Köztársaságból (Soemtronic) és Lengyelországból (MERA) szerezhetnek be.

5. Zárszó

A fentieket összefoglalva megállapítható, hogy a számítógépes adatkivitel fő eszköze a nyomtató. Túlnyomó többségben elektromechanikus ütőnyomtatókat alkalmaznak, az egyéb fizikai elven működő különleges nyomtatók lassabban terjednek el, főleg speciális alkalmazási területeken. A jövőben a technológia további fejlődésével nagyobb arányú elterjedésük várható.

Meg kell azonban kérdőjelezni a jelenlegi nagy volumenű információ kinyomtatásának reális szükségességét. Véleményem szerint megengedhetetlenül sok papíranyagot használnak fel olyan információ kinyomtatására, amely csak esetlegesen kerül felhasználásra, és rövid időn belül elévül, kidobásra kerül. Szervezés kérdése, hogy az ilyen természetű információ ne papírra, hanem például mágneses információhordozóra kerüljön, ahonnan — szükség esetén — display-re lehívható, s ha kell, erről a töredékről „hard copy" is készíthető. Ezzel is hozzá lehet járulni az egész világon sürgetett takarékossági intézkedésekhez.

AZ ÉPÍTÉSI GEOTECHNIKAI ADATTÁR, MINT A SZÁMÍTÓGÉPPEL SEGÍTETT MÉRNÖKI TERVEZÉS GÉPI INFORMÁCIÓS RENDSZERE

Reményi Péter—dr. Abaffy József—Lukács Márta
FTV—MTA SZTAKI—MTA SZTAKI

1. Műszaki előzmények

Az építőipar az államosítás óta eltelt időszakban szervezeti, technikai-technológiai és műszaki felkészültség vonatkozásában is ugrásszerű fejlődést mutatott. Az építésiparosításban elért eredmények, az új szerkezeti megoldások és építési anyagok alkalmazása az építéstervezésben és a beruházásokat előkészítő különböző mérnöki munkákban is mennyiségi és minőségi változásokat követeltek meg. Ezek a hagyományos módszerekkel és szemlélettel természetesen nem voltak teljesíthetők.

Az építésügyi miniszter utasítása már 1954-ben nemzetközi viszonylatban is újszerű és azóta is egyedülálló országos műszaki adattár kialakítását tette lehetővé, amikor elrendelte az építkezésekhez végzett talajmechanikai és hidrológiai vizsgálatok központi archiválását.

A szocialista fejlődés kezdeti szakaszában meghatározott alapvető célkitűzések:

- a rendkívül idő- és költségigényes feltárási, helyszíni és laborvizsgálati eredmények megőrzése az ország különböző tájegységeinek, településeinek, illetve tipikus építési talajainak függvényében jelentkező eltérő beépíthetőségi (műszaki és gazdasági) feltételek minél pontosabb megismerése céljából;
- a különböző talajtípusokhoz, illetve eltérő adottságú területekhez kapcsolódó gyakorlati építési tapasztalatok tömeges gyűjtése és értékelése az építési folyamat jobb szervezése és hatékonyságának javítása céljából;
- a feltáró-vizsgáló-szakértési kapacitás növelése az ismételt adatfelhasználás lehetőségének megteremtésével;

gyakorlatilag napjainkban is helytállóak. Az építésügyi és városfejlesztési miniszter 1978. X. 1-én hatályba lépett rendelete csupán korszerűsítette és kiegészítette az eredeti intézkedést, amikor az Építési Geotechnikai Adattár működését újra szabályozta.

Az adattárban jelenleg 94 000 komplett műszaki dokumentációban kereken 800 000 geotehnikai fúrás adatait, illetve az ezekhez végzett több millió talajmechanikai és vegyészeti laboratóriumi vizsgálati eredményt, s végül több százezer helyi építési tapasztalatot (pl. épületkárok, pincevízelöntések, felszín- és talajmozgások, próbaterhelési eredmények stb.) archiválunk. Az adattár jelenlegi árszinten *2 milliárd Ft. vizsgálati költséget reprezentál,* s ezáltal valós nemzeti vagyont képez.

Az adattár megalakulása óta nyilvános közkönyvtárként áll a beruházók, tervezők és kivitelezők, illetve az építésigazgatási szervek rendelkezésére. Fennállása óta kerekítve 40 000 látogató vette igénybe az adattárat, s összesen közel 250 000 dokumentációt kértek ki betekintésre, ismételt adatfelhasználásra. A forgalom bizonyítja, hogy az adattár információtartalma az építőiparban közvetlenül hasznosítható.

Mindezek alapján az ÉVM 1971-ben kutatási-fejlesztési feladatként jelölte meg a *Földmérő és Talajvizsgáló Vállalatnak* (FTV) az adattár technológiai-technikai korszerűsítését.

2. A korszerű számítógépes adattároló-kezelő rendszer kialakítása

A feladat megoldásához közreműködői szerződést kötött az FTV a MTA Számítástechnikai és Automatizálási Kutató Intézetének Numerikus Módszerek Osztályával.

A korszerűsítés során megoldandó feladatokat az alábbiak szerint rögzítettük:

- helytakarékos tárolás és állagvédelem;
- teljeskörű és hibamentes nyilvántartás;
- gyors és hibamentes hozzáférhetőség;
- szelektált adatelhívás, illetve származtatott és integrált célinformációk szolgáltatása;
- a legkülönbözőbb formájú információs igények széles körű kielégítése;
- folyamatos műszaki információ szolgáltatás feltételeinek megteremtése.

Az adattároló-kezelő-feldolgozó- és információs rendszer kialakítása, szervezése szempontjából tudomásul kellett vennünk, hogy:

1. Műszaki szempontból az adatok információ-tartalma, értékelhetősége és ismételt felhasználhatósága nem egyenértékű, ugyanis vannak:

- az idő függvényében változatlan, tartósan azonos értékű adatok;
- lassú, uralkodóan azonos változási trendet mutató adatok, melyek egy folyamat meghatározására vagy prognosztizálására alkalmasak;
- időben gyorsan változó, általában határok között ingadozó értékek, állapotjellemzők;
- természeti vagy művi hatásokra a vizsgálat végrehajtása óta alapvetően megváltozott adatok, melyek csak az adott hatások milyenségére vagy mértékére szolgáltatnak informá

Ezen kívül tudomásul kellett venni számos műszaki szabályozó és a sok évtizedes tervezési gyakorlatból és megszokásból származó korlátozó tényezőt is.

2. Számítástechnikai oldalról a hardware adottságok és software lehetőségek által behatárolt gépidő-optimum megkövetelte, hogy:

- tömegesen ismétlődő szöveges információkat kódoljuk;
- az alapadatokból számítással származtatható paraméterekkel a tároló kapacitást ne terheljük;
- az adatkérés gyakoriságának legjobban megfelelő adatcsoportosítást és file szervezést tosítsuk;
- a műszaki és számítástechnikai követelményeket optimálisan teljesítő azonosítási és kapcsolati rendszert dolgozzunk ki.

3. A fejlesztési munka eredményei

A rendszerszervezés alapján elfogadott fejlesztési célkitűzés egyrészt a komplex számítógépes programcsomag kidolgozását, másrészt a mikrofilmes adatrágzítés párhuzamos megvalósítását tette. Az ország időközben megváltozott gazdasági helyzetére tekintettel a jelentős devizát igénylő mikrofilm géppark beszerzését el kellett halasztani. Ugyanakkor a modellezés, az algoritmizálás, a kódolás és a programozás terén az elmúlt öt évben jelentős előrehaladást sikerült elérni. Ennek nyilvánvaló alapfeltétele volt, hogy a több évtizedes hagyományos ügymenetben egyrészt kialakult és bevált egy logikai rendszer és folyamat, másrészt rendelkeztünk az igények tartalmára, formájára és jelentkezési gyakoriságára vonatkozó tapasztalatokkal.

Ezideig elkészültek:

- a komplex számítógépes rendszer adattároló és adatbeviteli programjai;
- a leggyakrabban jelentkező információigényeknek megfelelő adatlehívó programok (fejlesztésük szintén folyamatban van);
- a legszokványosabb igényeket kielégítő táblázó kiíratő programok (fejlesztésük és bővítésük szintén folyamatban van);
- izovonalas térképek szerkesztési és rajzoló programja;
- a talajvízjárás értékelő és építési vízszint prognosztizáló programok.

Az ÉVM egységesítési és tipizálási törekvéseinek megfelelően a „FOGAS"-nak (File Organization Geotechnikai Algoritmusok Struktúrája) nevezett rendszer (Ábra) különböző file-jeinek adattöltéséhez kidolgoztuk és nyomtatva kiadtuk az adatlapokat.

A kutatás-fejlesztésére fordítható pénzügyi lehetőségeink korlátot szabtak a programozási munka ütemének. Ennek ellenére ma már abban a helyzetben vagyunk, hogy a kereken 30 milliós adattömeg géprevitelének előkészítése megkezdhető. A manuális adatelőkészítő és adatbeviteli munka, illetve a tárolt adatok helyességének ellenőrzése több évet vesz igénybe. Ezalatt a tervszerű programfejlesztés is befejezhető, s előreláthatólag a 6. ötéves terv végére a gépi adatkezelő rendszer az építőipar rendelkezésére állhat.

Gyakorlati használatba vétele révén biztosítja:

- a területi tervezés, a területfelhasználási-telepítési döntések mérnöki-gazdasági megalapozottságának jelentős növelését;
- a területhasználat és az építés előkészítés döntései megbízhatósági fokának javítását;
- a geotechnikai feltáró és vizsgáló kapacitás jelentős növelését, illetve az átfutási idő és a költségek csökkentését;
- a műszaki tervezés és kivitelezés előkészítettségi fokának emelését a hatékonyság (tervmódosítások minimalizálását, kivitel jobb előkészítése, mélyépítési megtakarítások, várakozási idők csökkentése stb.) növelését;
- az építésigazgatási jellegű hatósági munka műszaki megalapozottságának javítását;
- az ÉVM intézkedései alapján működő különböző szakági információs és szaktanácsadó szolgálatok tevékenységének átfutási idejének, színvonalának fokozását.;

Mindezek által a rendszer közvetlenül hozzájárul az MSZMP KB 1978. X. 12-i építőipari határozatának végrehajtásához, miszerint

„Alapvető követelmény a vezetés, az üzem- és munkaszervezés színvonalának emelése, a rendelkezésre álló eszközök és munkaerő ésszerűbb felhasználása. — A gazdasági és műszaki tervezésben, a kivitelezésben egyaránt szigorúan érvényesíteni kell a takarékosságot."

Ezzel kapcsolatban feltétlenül utalnunk kell arra, hogy a Földmérő és Talajvizsgáló Vállalat — amely az országos geotechnikai feltáró-vizsgáló kapacitás kb. 40%-át teszi ki — az 1965—1978 közötti 14 év alatt a szakértési-tervezési feladatainak 35,7%-át teljesítette ismételt adatfelhasználással. Ezzel 10,7 év fúrási kapacitást váltott ki, meggyorsítva a beruházások előkészítését, továbbá kereken 30 millió Ft feltárási költséget (energiatakarékosság!) takarított meg a beruházónak.;

Könnyen belátható, hogy a hagyományos üzemmóddal szemben a számítógépes adatkezelő rendszer a hatékonyságot megsokszorozza, s ezáltal közvetlenül szolgálja a beruházások gyorsabb előkészítését és az energiatakarékosságot is.

Az eddig elkészült programokat eseti adatbevitellel teszteltük. Közvetlen gyakorlati célok végrehajtása érdekében a teljes programcsomag komplex működését Budapest főváros talajvízjárásának értékelésével ellenőrizzük. Ugyancsak a fővárosi lakásépítés elősegítését szolgálja a XVIII—XIX. kerületeknek mint reprezentatív minta feladatnak a komplex építésföldtani értékelése. Az ehhez szükséges adatelőkészítést ebben az évben fejezzük be, a futtatásra 1980-ban kerül sor.

4. Következtetések

Az eddigiekben — rendkívül érintőlegesen — vázolt helyzet is egyértelműen bizonyítja, hogy a számítógépes adattári rendszer közvetlen termelési feladatokat szolgál ki az építőipari ágazatban. Központi szolgáltató funkciót lát el a műszaki tervező, a beruházó és a kivitelező szervek felé, de a tudományos kutatásban is új lehetőségeket és utakat nyit.

Értelemszerű, hogy a geotechnikában, s azon belül az alapozási-mélyépítési munkában adott építési altalajok, a talajvíz, s még számos más helyi természeti és művi adottság képezi rendszerkomponenseket, melyek összessége egyben determinálja a beépíthetőség műszaki és daági feltételeit. Az építőipari műszaki közéletben ma már nem szorul bizonyításra, hogy a b ruházásokés az építési folyamat hatékonyságának mind dominánsabb tényezőjét képezik a sz alatti munkák. A számítógépes adattároló-feldolgozó rendszer tehát valós népgazdasági köve mények közvetlen megoldását szolgálja.

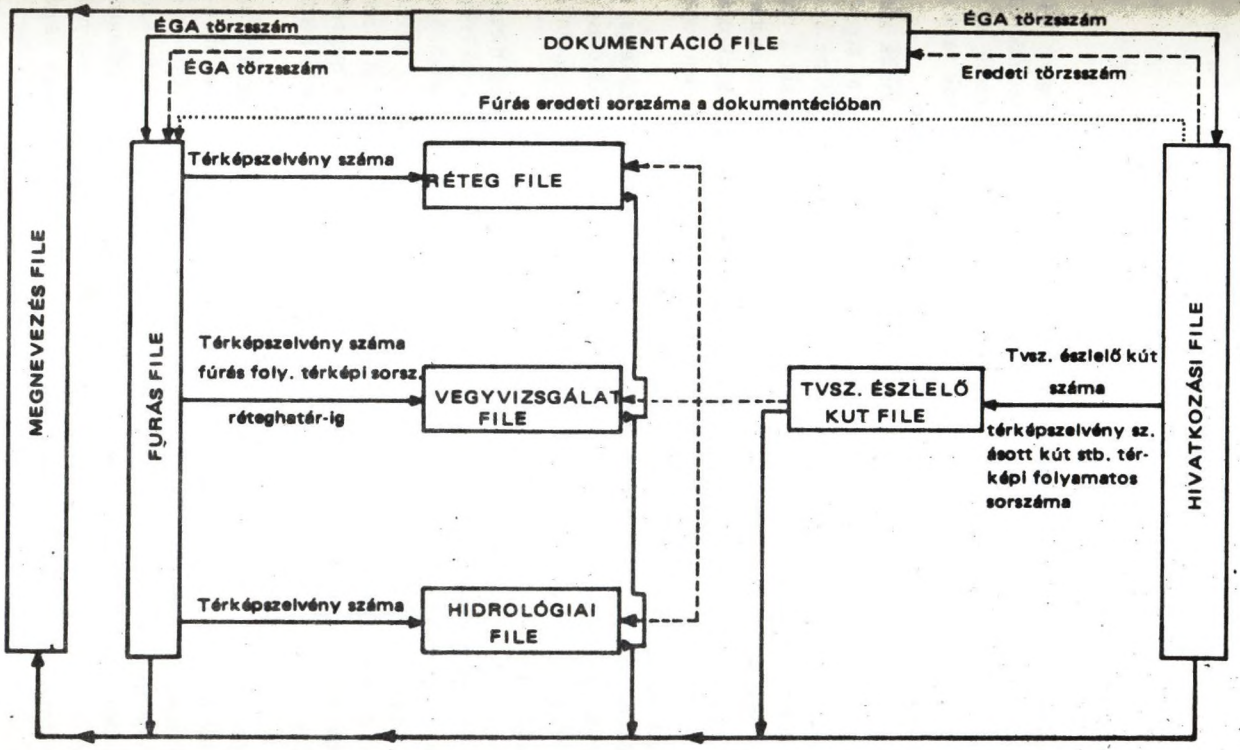
Tájékoztatásként megemlítjük, hogy az *Országos Környezet- és Természetvédelmi Hiv* kezdeményezése alapján elvi szinten foglalkoztunk a rendszernek az országos környezetvéde információs rendszernek az országos környezetvédelmi információs rendszerbe való csatlako nak lehetőségével is.

A technikai fejlődés időközben jelentősen módosította feltáró, helyszíni gyorsvizsgálat berendezéseinket, műszereinket és labor eszközeinket is. Ezek automatikus adatregisztrálás közvetlen számítógépi adathordozóra ugyancsak folyamatba tettük kutatási-fejlesztési téma Az így nyert adatok közvetlenül az adattári rendszerbe is betárolhatók lesznek, teljesen kikü bölve az emberi beavatkozást.;

A bármely speciális célfeladat megoldására alkalmas Építési Geotechnikai Adattár önál rendszert alkot. Már 1971-ben a rendszerszervezés és kialakítás kezdetekor figyelembe vettü azonban, hogy egyúttal a számítógéppel segített alapozástervezés háttér adatbázisának köve nyit is kielégítse. Ezen tervezési rendszer elvi felépítésének előkészítését ugyancsak megkez

Mindezek eredőjeként egy olyan önálló alrendszerekből felépített komplex rendszer üz meltetésére nyílik lehetőség, melyben már valóban érvényesül az a megállapítás, hogy:

„Az ismeretek csak a tudás anyagát alkotják, értelmét a *törekvés* adja, mihez kezdünk ismeretekkel."


ÉGA törzsszám
DOKUMENTÁCIÓ FILE
ÉGA törzsszám
ÉGA törzsszám
Eredeti törzsszám
Fúrás eredeti sorszáma a dokumentációban
MEGNEVEZÉS FILE
FÚRÁS FILE
Térképszelvény száma
RÉTEG FILE
Térképszelvény száma
fúrás foly. térképi sorsz.
réteghatár-ig
VEGYVIZSGÁLAT FILE
TVSZ. ÉSZLELŐ KUT FILE
Tvsz. észlelő kút száma
térképszelvény sz. ásott kút stb. térképi folyamatos sorszáma
HIVATKOZÁSI FILE
Térképszelvény száma
HIDROLÓGIAI FILE

A TIPUSMENTES λ-KALKULUS ALKALMAZÁSA PROGRAMNYELVEK SZEMANTIKÁJÁNAK LEÍRÁSÁRA

Révész György
MTA SZTAKI

1. Bevezetés

A függvény fogalma kétségtelenül a matematikai gondolkodás egyik legsikerültebb a sztrakciója, amely természetesen nem készen pattant ki egy nagy matematikus fejéből, h hosszabb fejlődésnek az eredménye. Ez a fejlődés nyilván ma sem zárult le, sőt a számí technika térhódítása nyomán ezt a fogalmat újabb és újabb revíziónak vetik alá különös kintettel a kiszámíthatóság gyakorlati megvalósításának nehézségeire.

A függvénytan kialakulásának kezdeti szakaszában a matematikusok kétségtelenül zárt alakban felírható függvényekre gondoltak, melyeknek az értéke az argumentumból nyos elemi lépések véges sokszor történő alkalmazásával nyerhető. Persze az elemi függv között mindjárt ott voltak az ún. transzcendens függvények, amelyeket a négy alapművegv segítségével nem lehet véges sok lépésben kiszámítani. A továbbiakban azután éppen ezek az elemi függvényeknek a sorfejtései mintájára képzett végtelen sorok okozták a legtöbb dot. Voltak, akik az ilyen végtelen sorokkal definiált függvényeket nem fogadták el, ha nem lehetett valamilyen zárt alakra hozni azokat.

A következő lényeges változást a halmazelmélet kialakulása jelentette, melynek sö létrejött az ún. Dirichlet-féle függvényfogalom, ahol már semmiféle kikötést nem teszün ra nézve, hogy milyen módon származtatható az argumentumból a függvényérték. Ez u függvényfogalom terjedt el azóta a matematikában.

Volt azonban egy probléma, nevezetesen a halmazelmélet szilárd megalapozottság a hiánya, ami megfontolásra késztette a matematikusok egy részét a tekintetben, hogy den további megszorítás nélkül elfogadják-e a végrelen halmazokon értelmezett függvén fogalmát. Kétségtelen, hogy ha egy ilyen függvényt meg akarunk adni, akkor végtelen helyen kell definiálnunk az értékét, s ez végtelen hosszú leírást igényel, hacsak nincs v milyen véges sok szabályba sűríthető számítási előírásunk a függvény megadására. A szá tástudomány nyelvén ez utóbbi nem más, mint egy program, mellyel a kérdéses függvé gépi úton is kiszámíthatjuk.

Az ún. extenzív, vagyis halmazelméleti függvényfogalommal szemben tehát megkü böztetjük az intenzív – vagy inkább intencionális – függvényfogalmat, mely utóbbi m egy véges terjedelmű leíráshoz, ha tetszik programhoz kapcsolódik. Nyilván szükségünk mindkét fogalomra, hiszen csak extenzív értelemben mondhatjuk például azt, hogy kétl lönböző program ugyanazt a függvényt számítja ki. Másrészt viszont számossági megfo sokból rögtön látható, hogy az extenzív értelemben vett függvények túlnyomó többsé ve kiszámíthatatlan. (Véges hosszúságú program csak megszámlálhatóan végtelen sok l

Egy program szemantikáján valójában azt az extenzív értelemben vett függvényt amelyet a kérdéses program kiszámít. A definíció szerint önmagában nem okoz különö problémát, a kérdés viszont az, hogy mire használható ez a definíció. Nem elég ugya ben tetszőleges program szemantikáját definiálnunk, hanem az itt felvetődő számtalan kérdésre is meg kell próbálnunk válaszolni. Ez pedig távolról sem könnyű feladat. Ehe

mindjárt megkérdezhetjük, hogy egy adott programról hogyan dönthető el, hogy egyáltalán kiszámít-e valamilyen függvényt. Sajnos akármilyen érdemleges programfogalommal dolgozunk, mindjárt kiderül, hogy általában be kell érnünk részlegesen, azaz parciálisan definiált függvényekkel. Egy tetszőleges programról ugyanis nem lehet eldönteni, hogy minden inputra véges sok lépésben megáll-e vagy sem. Ugyanilyen nehézségekbe ütközik annak az eldöntése, hogy két különböző program ugyanazt a függvényt számítja-e ki vagy sem. A számos negatív eredmény mellett az utóbbi időben kezdenek kibontakozni olyan eredmények, amelyek a szemantika egyik vagy másik fontos kérdésére pozitív választ tudnak adni. Az egyik legérdekesebb fejlemény e tekintetben az, hogy sikerült a kiszámíthatósághoz extenzionális értelemben szükséges feltételeket megadni, s ezáltal a kérféle függvényfogalom közötti kapcsolatot mélyebben megérteni. Ide azonban hosszú út vezetett, melyet a jelen dolgozatban nem célunk bejárni.

A továbbiakban azt kívánjuk bemutatni, hogy a szementikai problémák fent vázolt megközelítéséhez hogyan lehet a λ-kalkulust formális eszközként felhasználni.

2. Az extenzív függvényfogalom és a λ-kalkulus

A halmazelméleti függvényfogalom értelmében a függvény fogalmához hozzátartozik a függvény értelmezési tartományának (domain) és értékkészletének (codomain) a megadása. Ezt szokás az

$$f : D \to C$$

jelöléssel kifejezni, ahol tehát az f függvény egy a D halmazból a C halmazba való leképezés. A D-ből C-be ható függvények halmazát, vagy terét C^D-vel jelöljük. Erről rögtön látható, hogy nagyobb számosságú mint D, hacsak C számossága legalább 2. Ha most a D egy tetszőleges elemét x-szel jelöljük, akkor a hagyományos jelölés szerint $f(x) \in C$, feltéve hogy $f \in C^D$. Ez azt mutatja, hogy a változók jelölésére használt szimbólumokat meg kell különböztetnünk a függvények jelölésére használt szimbólumoktól, hiszen különböző halmazoknak az elemeit jelölik.

Egy változó *típusán* azt a halmazt értjük, amelyet a változó értékei befutnak. Pontosabban egy függvényváltozó a megfelelő D és C halmazokból képzett rendezett pár, [D→C] jelöli a függvény típusát. Ha most f [D→C] típusú, g pedig [C→E] típusú, akkor értelmezhető az $f \circ g$ *kompozíció*, mely [D→E] típusú. Az egyöntetűség kedvéért megállapodunk abban, hogy egy egyszerűbb $x \in D$ változót [1→D] típusúnak tekintünk, ahol 1 az egyelemű „egységhalmazt" jelenti.

A kompozíció így nem más mint egy függvénynek a behelyettesítése az argumentum helyére. Így viszont máris olyan összetett kifejezéseket nyerhetünk, amelyekben több különféle változó szerepel, s adott esetben kérdésessé válhat, hogy mit mivel kell helyettesítenünk. Ha csak az f o g kompozíciót vesszük, s itt a g helyére egy [C→F] típusú h függvényt helyettesítünk, akkor már egy [D→F] függvényt kapunk. A függvényeket jelölő változókat ugyanis egyformán kell kezelnünk a közönséges változókkal, ha azt akarjuk, hogy a rendszerünk valamennyire is általános legyen.

A programozásban ezt az igényt függvénydeklarációk használatával igyekszenek kielégíteni, de ez többnyire csak félmegoldást jelent, mivel a függvénynevek használatát sokkal szigorúbb megkötések korlátozzák mint a közönséges változókét. Elvárhatnók ugyanis, hogy ha egy összetett függvény deklarációjában szereplő valamely függvénynév helyére más függvényt helyettesítünk, akkor egy újabb függvényt kapjunk. Nos az ilyen — függvényre alkalmazva ismét függvényt eredményező függvények — használata a legtöbb programnyelvből hiányzik. Pedig minden fordítóprogram lényegében egy ilyen összetett függvény.

Az f o g függvény alkalmazását a x-re g(f(x)) formában szokás kifejezni. Ha mo helyébe egy h függvényt helyettesítünk, akkor mindjárt megváltozik az egész kifejezés me. A típusok összhangját betartva a g(f(h)) kifejezésben bármely változót helyettesíth egy újabb változóval, sőt összetett függvénykifejezéssel, s ezért a helyettesítések előír explicitté kell tennünk. Erre jó a λ-jelölés, mely szerint

$$\lambda f \,.\, g\ (f(h))$$

egy olyan függvényt jelöl, melynek az f a behelyettesítendő változója, míg a

$$\lambda g \,.\, g\ (f(h))$$

kifejezésben a g az ilyen változó. Ha most az elsőt F-fel a másodikat pedig G-vel jelö akkor

$$F(G) = g(h(f(h)))$$
$$G(F) = g(f(h(h)))$$

Ez az egyszerű példa is mutatja, hogy milyen bonyodalmakkal kell szembenézni lönösen ha a típusokra is tekintettel kell lennünk. Jó lenne olyan típusokkal dolgozni mindenféle kompozíció meg van engedve beleértve az önmagára való alkalmazást is, ami lehetősen problematikusnak látszik. Egyszóval egy olyan univerzális tartományt kellene nunk, melyen minden függvény értelmezve van, s amelyből egyik sem vezet ki. Előve hogy ilyen tartomány valóban létezik, használhatjuk a λ-kalkulusnak az ún. típusmen tozatát, ahol a típusokkal nem kell törődnünk.

A λ-jelölésmódot arra használjuk, hogy extenzionális értelemben adottnak tekint függvényekből újabbakat konstruáljunk. A konstrukcióhoz mindössze két műveletet ha lunk, az absztrakciót és az applikációt, mely utóbbi lényegében azonos a kompozíció absztrakció viszont nem más, mint valamely korábban adottnak tekintett változónak változóvá tétele, azaz lambdásítása. A lambdásított változókat a továbbiakban kötött zóknak, a többieket pedig szabad változóknak nevezzük.

A λ-kifejezések halmazát formálisan az alábbi környezetfüggetlen grammatika ál nerált nyelvnek tekintjük:

$$\langle \lambda\text{-kifejezés} \rangle ::= \langle \text{változó} \rangle \mid \lambda \langle \text{változó} \rangle \,.\, \langle \lambda\text{-kifejezés} \rangle \mid$$
$$(\langle \lambda\text{-kifejezés} \rangle)\langle \lambda\text{-kifejezés} \rangle,$$

ahol a változók halmaza mondjuk az ALGOL 60 értelmében megengedett azonosítóknak Fel kell hívnunk a figyelmet arra, hogy a fenti szintaxis szerint mi a hagyományos jelö tól eltérőleg a függvényapplikációban nem az argumentumot tesszük zárójelbe, hanem „applikátort", vagyis az applikáló függvényt. Ennek csupán a zárójelek balról jobbra felbontásával kapcsolatos technikai okai vannak.

A következő kérdés most már az, hogy különböző módon konstruált λ-kifejezé lölhetnek-e extenzionális értelemben azonos függvényeket, és ha igen, akkor ezt hogy het e kifejezések formális vizsgálata alapján eldönteni.

Az első kérdésre a válasz nyilván pozitív, hiszen például a (λy.(x)y)z kifejezés ugyanazt a függvényt kell jelentenie, mint a (z)y-nak. A második kérdés vizsgálatára is bevezetünk olyan elemi transzformációkat, melyek a λ-kifejezéseket más, lehetőleg szerűbb szerkezetű, de az eredetivel azonos értelmű kifejezésekbe viszik át. Ezeket a transzformációkat konverziós szabályoknak szokás nevezni, s a pontos megadásukhoz

günk van a változók kötött, ill. szabad előfordulásainak a megkülönböztetésére. Egy P kifejezésben előforduló x változó valamely előfordulása kötött a P-ben, ha annak egy $\lambda x.Q$ alakú részében szerepel. A szabad előfordulást induktív módon definiáljuk:

(1) x szabadon fordul elő az x-ben;

(2) ha x valamely előfordulása szabad egy P kifejezésben és $y \neq x$, akkor az x szóban forgó előfordulása szabad a $\lambda y.P$-ben,

(3) ha x valamely előfordulása szabad egy P kifejezésben, akkor tetszőleges Q-ra az x szóban forgó előfordulása szabad mind a (P) Q mind a (Q) P kifejezésekben.

A fentiekből következik, hogy egy P kifejezésben egy adott x változó szabadon is és kötötten is előfordulhat (persze más-más helyen). A kötött változók konkrét jelölése nem befolyásolja a kifejezések értelmét, így azokat bármikor átkeresztelhetjük, s ezt fejezi ki az első konverziós szabály, melyet α-val jelölünk. Jelöljük először $[y/x]P$-vel az x P-beli szabad előfordulásainak y-nal való helyettesítését, pontosabban a P-ből így nyert kifejezést.

(α) $\lambda x.P \rightarrow \lambda y.[y/x]P$, ha y sem szabadon sem kötötten nem fordul elő a P-ben.

Az alábbi ún. β-szabályok az applikációra vonatkoznak, s lényegében egy kijelölt helyettesítésnek a részleges vagy teljes végrehajtását fejezik ki.

(β1) $(\lambda x.x)\ Q \rightarrow Q$

(β2) $(\lambda x.y)\ Q \rightarrow y$, ha $x \neq y$

(β3) $(\lambda x.\lambda x.P)\ Q \rightarrow \lambda x.P$

(β4) $(\lambda x.\lambda y.P)\ Q \rightarrow \lambda z.\ (\lambda x.\ [z/y]P)Q$, ahol z nem fordul elő sem szabadon, sem kötötten a P-ben és a Q-ban

(β5) $(\lambda x.(P_1)P_2)\ Q \rightarrow ((\lambda x.P_1)\ Q)\ (\lambda x.P_2)\ Q$

Ha egy P kifejezésre vagy annak egy részére valamilyen konverziós szabályt alkalmazva egy P' kifejezést kapunk, akkor azt mondjuk, hogy a P egy lépésben redukálható P'-re. Ennek kézenfekvő általánosítása a véges sok lépésben való redukálhatóság.

Egy λ-kifejezésről azt mondjuk, hogy normálalakban van, ha semmilyen β-szabályt nem lehet benne alkalmazni.

Ha most P és Q mindketten redukálhatók valamely R kifejezésre, akkor ugyanazt a függvényt definiálják. Ily módon egy ekvivalencia reláció értelmezhető a λ-kifejezések között. Az ekvivalencia eldöntése tetszőleges P és Q kifejezésekre nézve azonban sajnos lehetetlen. Abban az esetben ha mind P mind Q valamilyen P', ill. Q' normálalakra hozhatók, akkor az ekvivalencia könnyen eldönthető, hiszen P' és Q' csak úgy lehet ekvivalens ha csupán α-szabályok segítségével egymásba transzformálhatók. (Ezt a kötött változók szisztematikus cseréjével véges sok lépésben meg lehet állapítani.) Az ugyanis nem fordulhat elő, hogy ugyanazt a kifejezést két lényegileg különböző normálalakra lehessen redukálni. A normálalak egyértelműsége a λ-kalkulus egyik legfontosabb tételének, a Church-Rosser tételnek a folyománya, melynek bizonyítására itt nem térünk ki.

Fontos azonban megjegyezni, hogy a fentiekben definiált ekvivalencia reláció nem meríti ki a szemantikai egyenlőség minden esetét. Mélyebb megfontolások révén kimutatható, hogy vannak olyan normálalak nélküli kifejezések, melyek azonos értelműek bizonyos normálalakban levő kifejezésekkel. Normálalak nélküli kifejezésre egyszerű példa a

$$(\lambda x.(x)x)\ \lambda x.(x)x$$

kifejezés, melyre mindig alkalmazható valamilyen β-szabály, de mindig önmagába alakul vissza. Divergens módon viselkedik viszont a

$$(\lambda x.(x)(x)x)\ \lambda x.(x)(x)x$$

kifejezés. Ezeknek a kifejezéseknek szerencsére semmilyen használható értelmet nem kell tu-

lajdonítanunk. Az önalkalmazhatóságról, tehát az (x)x alakú kifejezések értelmes voltáról ban általában nem kell lemondanunk, sőt ennek lényeges szerepe van a rekurzív definíció

A λ-kifejezések körében tehát rendelkezünk egy olyan formális kalkulussal, melynek gítségével összetett függvénykifejezések sok esetben normálalakra hozhatók, s ezáltal külö ző kifejezések ekvivalenciája is sok esetben eldönthető. Sajnos viszont általában nem eldö tő az, hogy egy adott λ-kifejezés normálalakra hozható-e vagy sem. Mégis ez a kalkulus moly segítséget nyújt bonyolult kifejezések vizsgálatánál, és gépi úton történő feldolgozás

3. Rekurzív függvények

A λ-kalkulus fenti tárgyalásában megmaradtunk egy extenzionális függvényfogalom Ha most konkrét függvényeket akarunk definiálni, akkor először is meg kell adnunk egy rét tartományt, és konkrét kiinduló függvényeket, melyekből a többit felépítjük. Egyszerűség kedvéért vegyük először a természetes számok halmazát, s jelöljük ezt ω-val. Egészítsük ω-t egy speciális új elemmel, melyet a ⊥ jellel jelölünk, és azt mondjuk, hogy az a definiálatlan számértéket jelenti.

Ezekután terjesszük ki a primitív rekurzív függvények definícióját erre a kibővített mazra. Először is az ún. rákövetkező (successor) függvényt, S-et úgy definiáljuk, hogy le

$$(S)\ \bot = \bot,$$

továbbá az

$$(S)\ O,\ (S)\ (S)\ O,\ \ldots$$

sorozat tagjai legyenek mind különbözőek, és merítsék ki a ω-t. A megelőző (predecessz függvényt a

$$(P)\ O = O$$
$$(P)\ (S)\ x = x$$

primitív rekurzióval definiáljuk. Megjegyzendő, hogy ezzel a (P) ⊥ = ⊥ összefüggést is m adtuk.

Az összeadást mint kétváltozós függvényt olyan egyváltozós függvényre vezetjük melyet valamely természetes számra (t.i. az első argumentumra) alkalmazva egy függvén kapunk eredményül, s ez utóbbit alkalmazva egy természetes számra (második argumen megkapjuk a két szám összegét.

$$((A)\ O)\ y = y$$
$$((A)\ (S)\ x)\ y = (S)\ ((A)\ x)\ y$$

Ezzel az ((A)x)y értékét minden x,y értékpárra definiáltuk, beleértve az x=⊥, ill. esetét is. Teljes indukcióval rögtön belátható, hogy ((A)x)⊥=⊥ minden x-re. Másrészt a dik egyenletből azt kapjuk, hogy minden y-ra

$$((A)\ (S)\bot)y = (S)\ ((A)\bot)y$$

azaz

$$((A)\bot)y = (S)\ ((A)\bot)y;$$

amiből ((A)⊥)y = ⊥ következik, hiszen a z=(S)z egyenletnek nincs megoldása.

A definícióból az is belátható, hogy minden x,y értékpárra teljesül az ((A)x)y=((A)y)x egyenlőség.

A szorzás definíciója a következő:

$$((M)O)y = O$$
$$((M)(S)x)y = ((A)((M)x)y)y$$

Ebből rögtön adódik, hogy minden y-ra teljesül az

$$((M)(S)O)y = y$$

összefüggés, továbbá $x \neq \perp \neq y$ esetén az

$$((M)x)y = ((M)y)x$$

egyenlőség. A nyilvánvaló $((M)O)\perp = O$ egyenlőséggel szemben az $((M)\perp)O = ((A)((M)\perp)O)O$ egyenlet megoldásaként bármely szám megadható, s ezért azt meghatározatlannak tekintjük, és az $((M)\perp)O = \perp$ megállapodást tesszük. Ha $y \neq O$, akkor az $((M)\perp)y = ((A)((M)\perp)y)y$ egyenletnek már csak egy megoldása van, azaz $((M)\perp)y = \perp$. Nyilván $((M)x)\perp = \perp$ is igaz minden $x \neq O$-ra. A szorzat értékét tehát az első tényező zérus volta predeterminálja akkor is, ha a második tényező értéke definiálatlan. A második tényező zérus volta esetében viszont ez a predeterminálás már nem egészen egyértelmű. Mindez konzisztens a szorzás klasszikus definíciójával.

Definiáljuk most a faktoriális függvényt.

$$(F)O = (S)O$$
$$(F)(S)x = ((M)(F)x)(S)x$$

Itt most az

$$(F)\perp = ((M)(F)\perp)\perp$$

összefüggés alapján az $(F)\perp$ értékére megint két megoldás adódik, $\perp$ és O.

Ha a definíció második sorát $(F)(S)x = ((M)(S)x)(F)x$ alakban adjuk meg, akkor az $(F)\perp = ((M)\perp)(F)\perp$ összefüggést kapjuk, melynek csak egy megoldása van. A P függvény felhasználásával a faktoriális definícióját egyetlen sorral is megadhatjuk minden x-re:

$$(F)x = (S)(P)((M)x)(F)(P)x$$

Ebből

$$(F)O = (S)(P)((M)O)(F)O = (S)O$$

Az M definíciója szerint, és

$$(F)\perp = ((M)\perp)(F)\perp = \perp$$

adódik.

A fenti definícióban a zérusvizsgálat a P-be van elbújtatva. Az S, P, M függvények ismeretében tehát a primitív rekurziót függvénykompozícióra vezettük vissza. A fenti egyenletben az S, P és M a konstansok szerepét töltik be, az F pedig az éppen definiálandó függvényváltozó. Miután az összefüggésnek minden x-re teljesülnie kell, a következő függvényegyenletet is felírhatjuk:

$$F = \lambda x.(S)\ (P)((M)x)\ (F)(P)x$$

Ezt az F-re, mint ismeretlen függvényre vonatkozó egyenletet

$$F = (\lambda y.\lambda x.(S)(P)((M)x)(y)(P)x)F$$

alakba írva egy

$$F = (H)F$$

alakú egyenletet kapunk, melyből az F a H függvény fixpontjaként adódik. Ezt a függvény-

egyenletet pontonkénti egyenletnek is tekinthetjük, s akkor egy adott n értékre az e

$$(F)n = ((H)F)n = (S)(P)((M)n)(F)(P)n$$

egyenletet kapjuk. Ez most csak n=O és n=⊥ mellett ad fixpont egyenletet, de általá

$$F = (H)F$$

alakú függvényegyenlet pontonként is adhat olyan fixpontegyenletet, ahol a meghatá ismeretlen mindkét oldalon szerepel.

Az általános rekurzív függvényeket éppen ilyen egyenletekkel, illetve egyenletre rekkel szokás definiálni.

4. Programnyelvek szemantikája

A szokásos programnyelvekkel kapcsolatban a szemantika definíciójának a kérd lában nehezen kezelhető. Ennek fő oka az, hogy ezeket a nyelveket még jóval a szer kutatások elmélyülése előtt tervezték. Másrészt azonban ma sem mondhatjuk, hogy m probléma meg van oldva, sőt rengeteg nehézség tornyosul még ma is egy gyakorlatba használható szemantika kidolgozása előtt.

Mi itt csak azt akarjuk bemutatni, hogy a programnyelvekben használt bizonyo strukciókat hogyan lehet λ-kifejezésekre átírni, s így a szemantika definícióját a λ-ka szemantikájára visszavezetni.

Vegyünk egy egyszerű példát. Az alábbi ALGOL típusú nyelven írott program egy egyszerű függvényt számít ki:

```
read(x);
y:=x+2;
x:=(x*y)+1;
z:=(3*y)+xí
print(z);
```

A read és print valamint az alapműveletek most ismert függvényeknek felelje A read(x) azt jelenti, hogy az x a független változó, tehát a megfelelő λ-kifejezés λ kezdődni. Az y:=x+2 értékadó utasítás jobboldalát

$$((A)x)2$$

alakba írhatjuk. Ha most feltesszük, hogy a hátralévő programrésznek egy K kifejez meg, akkor ez az értékadás azt jelenti, hogy a K-ban y helyére mindenütt ((A)x)2-t behelyettesítenünk. De ezt egyszerűen a

$$(\lambda x.K)((A)x)2$$

λ-kifejezéssel lehet megadni. Hasonlóan folytatva az átírást végül is a

$$\lambda x.(\lambda y.(\lambda x(\lambda z.(\text{print})z)((A)((M)3)y)x)(S)((M)x)y)((A)x)2$$

kifejezést kapjuk, melyet a konverziós szabályok segítségével normálalakra hozhatun ilyen átirogatást automatikusan csináljuk, akkor közben nem vesszük észre azt, ha

tozóink maradnak a kifejezésekben, amelyeknek nem adunk értéket. Ha viszont minden szabad változónak $\perp$ értéket adunk, akkor kiderül, hogy van-e értelme a programnak vagy sem.

Egy függvénydeklarációt hasonlóan kezelhetünk mint egy értékadó utasítást, csak itt a jobboldalon egy λ-val kezdődő kifejezés fog állni, amint azt a faktoriális függvény definíciójában is láttuk. Ezt a kifejezést kell a programban a függvény azonosítója helyére mindenütt behelyettesítenünk. Eszerint például az

$$f:=\lambda x.\ (S)(P)((M)x)(f)(P)x$$

függvénydeklarációt egy K kifejezéssel megadott programhoz a

$$(\lambda f.K)\lambda x.(S)(P)((M)x)(f)(P)x$$

formában illeszthetjük hozzá. Igen ám, csakhogy egy ilyen rekurzív definícióban a jobboldalon is szerepel az f, s így nem küszöbölődik ki automatikusan. Ha nem rekurzív deklarációról van szó, akkor minden rendben van, mert végül is csak ismert függvények maradnak benn a kifejezésben.

A rekurzív definíciókkal tehát külön kell foglalkoznunk. Nyilván a kölcsönös, (egymásba ágyazott) rekurzió lehetőségét is figyelembe kell vennünk. Ez azt jelenti, hogy adva van egy többismeretlenes egyenletrendszerünk

$$\begin{aligned}
F_1 &= (\ldots(\lambda f_1.\ \lambda f_2 \ldots \lambda f_n.H_1)F_1)\ \ldots)F_n,\\
F_2 &= (\ldots(\lambda f_1.\ \lambda f_2 \ldots \lambda f_n.H_2)F_1)\ldots)F_n,\\
&\ \ \vdots\\
F_n &= (\ldots(\lambda f_1.\ \lambda f_2 \ldots \lambda f_n.H_n)F_1)\ldots(F_n,
\end{aligned}$$

melyet F_i-kre meg kell oldanunk. Általános módszert adni erre nem igen lehet, de ha nem minden függvényváltozó szerepel ténylegesen mindegyik jobboldal normálalakjában, akkor a fokozatos kiküszöböléssel esetleg boldogulunk. Ha például az F_1 definíciójában nem szerepel az F_1, akkor ezt az összes többi egyenletből is kiküszöbölhetjük.

Formálisan viszont az ún. fixpont kombinátorral tetszőleges fixpontegyenletet megoldhatunk. Tekintsük az alábbi kettős rekurziót:

$$\begin{aligned}
F&=((\lambda f.\lambda g.H)F)G\\
G&=((\lambda f.\lambda g.K)F)G
\end{aligned}$$

A másodikat egy G-re vonatkozó fixpontegyenletként kezelve a megoldást a

$$G=(\lambda x.(\lambda y.(x)(y)y)\lambda y.(x)(y)y)(\lambda f.\lambda g.K)F$$

kifejezés szolgáltatja. Belátható ugyanis, hogy az

$$y_\lambda = \lambda x.(\lambda y.(x)(y)y)\lambda y.(x)(y)y$$

kifejezést bármely H-ra alkalmazva az $(Y_\lambda)H$ kifejezés mindig megoldása az f=(H)f fixpontegyenletnek. (Vegyük észre, hogy az $(Y_\lambda)H$-ban már nem szerepel az f.) Ezt az összefüggést úgy igazolhatjuk, hogy redukáljuk az $(Y_\lambda)H$ kifejezést.

$$(Y_\lambda)H \equiv (\lambda x.(\lambda y.(x)(y)y)\lambda y.(x)(y)y)H =$$
$$= (\lambda y.(H)(y)y)\lambda y.(H)(y)y =$$
$$= (H)(\lambda y.(H)(y)y)\lambda y.(H)(y)y = (H)(Y_\lambda)H.$$

Visszatérve a fenti két egyenletünkre, most a G-re kapott kifejezést behelyettesítjük elsőbe:

$$F = ((\lambda f.\lambda g.H)F)(\lambda x.(\lambda y.(x)(y)y)\lambda y.(x)(y)y)(\lambda f.\lambda g.K)F,$$

majd ennek a jobboldaláról az F-et kiemelve most egy F-re vonatkozó fixpontegyenletet rünk, s ezt is a fenti módon oldjuk meg.

Látjuk tehát, hogy elvben minden rekurzív egyenletrendszert lefordíthatunk valamil λ-kifejezésre. Kérdés, hogy mit tudunk kezdeni ezzel a λ-kifejezéssel. A szomorú, de csa sem meglepő helyzet az, hogy az ilyen fixpontegyenleteknek a megoldásai általában nem hozhatók normálalakra, hiszen éppen ezért lehet (Y_λ) H-t (H) ... (H) (Y_λ) H-ra redukál Egy hibátlan program végrehajtásakor persze mindig csak véges számú iterációnak szabad lépnie, de az iterációk száma a program bemenő adataitól is függhet.

A λ-kifejezések használata tehát nem oldhat meg egycsapásra minden problémát. M úgy vélem, a fentiekből kitűnik, hogy nagyon hasznos eszköznek tekinthetjük a λ-kalkul szemantikai kérdések vizsgálatánál. A konverziós szabályok segítségével egyrészt esetleg kive mítható olyan kifejezések egyenlősége is, amelyek nem hozhatók normálalakra,másrészt bo zonyos kifejezéseket egyszerűbb alakra hozhatunk, miáltal a számítások hatékonyságát n hetjük. Számos kiaknázatlan lehetőség van tehát még a λ-kalkulus számítástudományi al mazására, és elsősorban az a nagy előnye, hogy egzakt matematikai módszerek használat teszi lehetővé e bonyolult kérdések vizsgálatánál.

Irodalom

[1] Church,A: The calculi of λ-expression, AMS No. 6, Princeton 1941.

[2] Curry,H.B. and Feys,R: Combinatory Logic, Vol.I, North Holland, Amsterdam, 1958.

[3] Scott,D: Logic and Programming Languages, ACM Turing Award Lecture, CACM 197 pp. 634–641

[4] Stoy,J.E: Denotetional semantics: The Scott-Strachey Approach to Programming Lan Theory, MIT Press, 1977.

[5] Milne,R. and Strachey,C: A Theory of Programming Language Semantics. Chapman London and Wiley, New York, 1976.

[6] Hindley,J.R, Lecher,B, and Seldin,J.P: Introduction to Combinatory Logic, London matical Society Lecture Note Series 7, Cambridge University Press, 1972.

[7] Wadsworth,C.P: The Relation between Computational and Denotational Properties fo s Models of the Lambda-calculus, SIAM J. Comput, Vol. 5. No. 3. 1976. 488–521

[8] Révész,Gy.: λ-calculus wothout substitution, Computer and Automation Institute of rian Academy of Sciences, 1978.

SÍKBELI MIKROSZKÓPOS RÉSZECSKÉK ALAKJÁNAK MORFOLÓGIAI JELLEMZÉSE KÉPELEMZÉSI MÓDSZERREL, KÜLÖNÖS TEKINTETTEL A KVANTITATÍV METALLOGRÁFIAI ALKALMAZÁSRA

Réti Tamás — Küllős József
GÉPIPARI TECHNOLÓGIAI INTÉZET

1. Bevezetés

A fémötvözetek szövetszerkezetének elemzése és minősítése az esetek jelentős hányadában próbatestek síkmetszetéről készített fény- illetve elektronmikroszkópos fényképvelvételek, az ún. szövetképek alapján történik. Az anyagszerkezet, a tulajdonságok és a szövetképek közötti fémtani összefüggések tanulmányozásával, a szövetképi információk mennyiségi értékelésével önálló tudományos irányzat, a kvantitatív metallográfia [1] foglalkozik. Ez utóbbi tárgykörébe tartozó, többek által vizsgált, de mind ez ideig megoldatlan probléma: a szövetképi — síkidommal reprezentálható — mikroszkópos részecskék alakjának kvantitatív jellemzése és alak szerinti automatikus osztályozása.

Az elmúlt évtizedben a matematikai alakfelismeréssel valamint az automatikus képfeldolgozással kapcsolatos kutatás eredményeképp több olyan eljárást is kidolgoztak, amelyek a vázolt problémák újszerű megközelítésére kínálnak lehetőséget [2-5].

Elsődlegesen a szakirodalmi eredményekre, részben pedig saját elemzésekre támaszkodva, az említett — zömmel karakter felismerés céljára kifejlesztett — módszereket megkíséreltük adaptálni a metallográfiai vonatkozású alakjellemzési problémák megoldására.

Célkitűzéseink a következők voltak:

- A síkmetszeti részecskék alakjának kvantitatív jellemzésére alkalmas számítógépes képelemzési módszerek kidolgozása.
- A metallográfiai gyakorlatban előforduló mikroszkópos részecskék (fázisok, szövetelemek) morfológiai elemzése nyomán a gyakoribb alaktípusok (típusalakzatok) kiválasztása.
- Az alak szerinti osztályozás céljára „alakosztályok" definiálása a típusalakzatok felhasználásával.

Vizsgálataink eredményei három különböző típusú módszer alkalmazásához fűződnek. Ezek hatóerejüket tekintve kiegészítik egymást. Kiválasztásuk a vizsgálandó részecskék eltérő geometriai-topológiai tulajdonságainak, a rendelkezésre álló számítástechnikai lehetőségeknek, illetve korlátoknak, valamint a megoldandó problémák speciális fémtani-metallográfiai hátterének együttes figyelembevételével történt. A következőkben a tanulmányozott módszerek elvét és az alkalmazással kapcsolatos főbb eredményeket ismerterjük.

2. Az alkalmazott alakjellemzési módszerek

Vizsgálatainkat síkidommal reprezentálható részecskékre korlátoztuk, kizárva az elemzésből a pont és vonalszerű alakzatokat. Két síkidom „azonos alakúságát" a szemléletnek megfelelően a következőképp definiáltuk. Az A és B két tetszőleges síkidomot azonos alakúaknak nevezzük ($A \sim B$ jelöléssel élve), amennyiben hasonlósági transzformációval (nagyítás, kicsinyítés) és térbeli mozgatással fedésbe hozhatók.

Minden síkidomhoz – alakjának számszerű jellemzése végett – egy N komponens

$$\bar{V} = \{V_1, V_2, \ldots V_N\}^T$$

ún. „alakvektort" rendeltünk egyértelműen. Az alakvektor komponenseivel szemben kö ményként támasztottuk, hogy a síkidom hasonlósági transzformációjával, síkbeli eltolás elforgatásával és tengelyes tükrözésével szemben invariánsak legyenek.

Megjegyezzük, hogy az alakvektor ismeretében a síkidom geometriájára nem köve tethetünk egyértelműen, ugyanis a morfológia jellemzésére véges sok számadat általában elégséges.

Tetszőleges A és B síkidom alak szerinti hasonlóságának mérését a megfelelő $\bar{V}_A$ alakvektorok távolságának mérésére vezettük vissza, ehhez a

$$\varrho_p(\bar{V}_A, \bar{V}_B) = \left\{ \sum_{i=1}^{N} | V_{Ai} - V_{Bi} |^p \right\}^{\frac{1}{p}} \qquad (1 \leqslant p < \infty)$$

típusú távolságmérő függvényt alkalmaztuk. A fentiekből következik, hogy az (1) távo rő függvény ismert metrikus tulajdonságai ellenére sem biztosítja a

$$\varrho_p(\bar{V}_A, \bar{V}_B) = 0 \quad \Longrightarrow \quad A \sim B$$

összefüggés teljesülését.

A vizsgálatainkhoz felhasznált és a következőkben röviden ismertetett alakjellemz járások közös vonása, hogy alkalmasak olyan alakvektorok előállítására, melyeknek ko nensei a kívánt invariancia tulajdonságokkal rendelkeznek.

2.1 Alakjellemzés a síkidomok geometriai adatai alapján

E módszer jellegzetessége, hogy az alakvektor előállításához a síkidomnak és ko burkának mérhető geometriai jellemzőiből képzett dimenzió nélküli mennyiségek szol alapul. Általában olyan alapadatok felhasználására célszerű törekedni, melyek szemléle tartalmúak, geometriailag vagy fizikailag könnyen értelmezhetők és meghatározásuk k fáradsággal jár. Ilyenek többek között: a síkidom F területe, P kerülete (értve ezen a roló vonalak tejles hosszát), a síkidom D átmérője (ez ekvivalens definíciók szerint ér hető, mint a síkidom pontjai közötti távolságok legnagyobbika, vagy a síkidom két p mos támaszegyenese közötti távolságok legnagyobbika), továbbá a síkidom E szélessé mely a két párhuzamos támaszegyenes közötti távolságok legkisebbike [7]. Tetszőlege idom A_H konvex burkára vonatkozó – az előbbiekkel analóg módon értelmezett jell közül (ezek: a konvex burok F_H területe, P_H kerülete, D_H átmérője, ill. E_H szélessé pán az F_H és a P_H ad új információt az eredeti A síkidom alakjára nézve, tekintett $D \equiv D_H$ és $E \equiv E_H$ azonosságokra [7]. Vizsgálatainkban az alakvektor előállításához rólag az F, P, F_H, P_H, D és E geometriai jellemzőket vettük figyelembe, ugyanis ezek rendelkezésre álló Quantimet 720 típusú [6] automatikus képelemző berendezéssel kö lenül meghatározhatók voltak. Az alakvektor komponenseinek származtatásakor a ha metriai jellemző között fennálló egyenlőtlenségek formájában megfogalmazott összefü re támaszkodtunk. Az ismertebbeket, a szakirodalom alapján [1, 4, 5, 7] rendszerez 1. táblázat összesíti.

	P	F_H	F_H	D	E
F	$4\pi F \leqslant P^2$ ●	$F \leqslant F_H$ X	$4\pi F \leqslant P_H^2$ ●	$\frac{4F}{\pi} \leqslant D^2$ ●	
P		$4\pi F_H \leqslant P^2$ ●	$P_H < P$ X	$2D < P$	$\pi E \leqslant P$ ■
F_H			$4\pi F_H \leqslant P_H^2$ ●	$\frac{4F_H}{\pi} \leqslant D^2$ ●	$\frac{E^2}{F_H} \leqslant 3$ +
P_H				$2D < P_H \leqslant \pi D$ ■	$\pi E \leqslant P_H$ ■
D					$E \leqslant D$ ■

Jelölések

F terület
P kerület
F_H konvex burok területe
P_H konvex burok kerülete
D átmérő
E szélesség

A fenti kifejezésekben egyenlőség akkor és csak akkor teljesül, ha a síkidom:

● körlemez

X konvex

■ állandó szélességű

\+ szabályos háromszög

1. táblázat Síkidomok néhány mérhető geometriai jellemzője közötti összefüggés

A geometriai egyenlőtlenségek felhasználása azzal az előnnyel jár, hogy az alakvektor komponenseinek nagysága megbecsülhető, és egyúttal információ nyerhető az alakzat típusára vonatkozóan is. A Quantimettel végzett vizsgálatok során a mikroszkópos részecskék alakjának jellemzésére többnyire a

$$\overline{V} = \left\{ \frac{4\pi F}{P^2}, \quad \frac{4F}{\pi D^2}, \quad \frac{2D}{P}, \quad \frac{P_H}{P}, \quad \frac{E^2}{\sqrt{3}\, F_H} \right\}^T \qquad (2)$$

típusú alakvektort alkalmaztuk. E vektor dimenzió nélküli komponenseinek értéke a [0, 1] intervallumba esik.

Hasonló típusú alakvektorok származtatására elvileg korlátlan lehetőség kínálkozik. A módszer előnye főleg szemléletességében rejlik. Hátrányai között említendő, hogy megfelelő képelemző célberendezés híján a geometriai adatok gyors és pontos meghatározása speciális algoritmusok felhasználását és a kép finom és alakhű raszterezését teszi szükségessé.

2.2 Alakjellemzés kontúrkövetésen és Fourier-analízisen alapuló módszerrel

A kontúrkövetés és Fourier-analízis együttes alkalmazásán alapuló módszerek [2, 4, 5, 8, 9, 10] topológiailag összefüggő, „lyukat nem tartalmazó" síkidomok morfológiai jellemzé-

sére használhatók. E módszer elvét, bármelyik változatát tekintsük is, a következőkben fo lalhatjuk össze. Normáljuk a síkidomok kerületét a mérethatás kiküszöbölése végett azono nagyságúra. A síkidom alakjának, – pontosabban ez utóbbit határoló síkbeli zárt görbe al jának – jellemzésére definiáljuk a tetszőleges A síkidomhoz tartozó $\alpha_A^{(j)}$ alakfüggvények osztályát. Az alakfüggvények közös vonása, hogy a síkidom kerülete szerint periódikusak, ka a kontúr körüljárási módjától, a görbe paraméterezési kezdőpontjának megválasztásától füg en különböznek egymástól. Az $\mathcal{A}$ alakosztályba targozó $\alpha_A^{(j)}$ alakfüggvény komplex vagy Fourier sorának együtthatóiból képezhetők az alakvektor származtatására hivatott mennyi gek. Vizsgálataink folyamán a módszer három különböző változatának alkalmazásával próbá koztunk, ezeknek elvét, különös tekintettel a harmadikra, az alábbiakban vázoljuk.

a) Tételezzük fel, hogy a vizsgálandó síkidomok mindegyikéhez egyértelműen hozzá delhető egy ún. centrum, amely az alakzat kitüntetett pontja [8]. Ennek kiválasztására tö féle lehetőség is kínálkozik, a hangsúly a kiválasztás egyértelműségén van. Kézenfekvő me dás az, ha centrumnak a síkidom súlypontját választjuk. A centrum kitűzését követően a tároló görbe egy tetszőlegesen választott pontjából kiindulva meghatározzuk rendre ezek centrumtól mért távolságát. Ily módon olyan periódikus alakfüggvényekhez juthatunk, me az ívhossz függvényében megadják a kerületi pontok centrumtól mért távolságát. Az alak tor komponensei az alakfüggvény Fourier sorának együtthatóiból nyerhetők.

b) Granulund, G.H. [2, 10] alakfüggvények származtatására az előbbitől némileg elté megoldást javasolt. Ennek lényege az a felismerés, hogy a síkidomot határoló görbe valós paraméterű komplex függvényekkel reprezentálható. Ez utóbbiak töltik be az alakfüggvén szerepét, és az alakfüggvények komplex Fourier sorának együtthatóiból képezhetők az al jellemzés céljára szolgáló mérőszámok.

c) A harmadik, gyakorlati szempontból általunk érdeklődésre leginkább számottart változat [5, 9] jellemzője, hogy az alakfüggvények származtatásában kitüntetett szerepe v a síkidomot határoló görbe ívhossz (s) függvényében meghatározott χ (s) görbületének.

Az alakfüggvények előállítása az alábbi, síkbeli zárt görbék görbületére vonatkozó t renciál geometriai tételen alapszik [11]. Legyen C egy ívhossz szerint paraméterezhető, s zárt görbe, mely véges sok C_i „síma" görbeszakaszból tevődik össze (1. ábra).
Jelölje a C_i görbeszakaszok $P(s_i)$ csatlakozási pontjaiban a „féloldali érintők" által bezárt szö $\Delta\alpha_i$ ez utóbbiak és a görbület a felvett körüljárási iránynak megfelelően legyenek előjeles mennyiségek. A fenti feltételeket tekintetbe véve, fennáll a

$$\sum_i \int_{C_i} \chi(s)\, ds + \sum_i \Delta\alpha_i = \pm 2\pi \qquad (3$$

összefüggés.

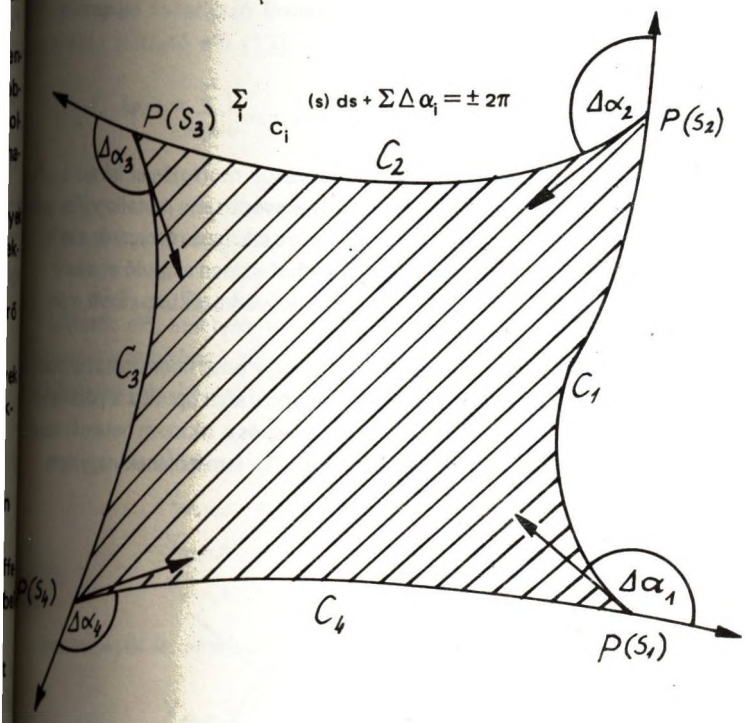
Az alakfüggvények származtatása végett a síkidomot határoló zárt görbére nézve a rüljárási irányt tekintsük pozitívnak, ha a (3) kifejezés bal oldala 2π-vel megegyező. Normáljuk a kerület nagyságát 2π értékűre és legyen a továbbiakban pozitív a válasz körüljárási irány. Legyen P_j a határoló görbe azon pontja, amely az $s = 0$ ívhossz szer paraméterezési kezdőpontnak felel meg. Definiáljuk a P_j kezdőponthoz tartozó $\hat{\alpha}_A^{(j)}$ fü segédfüggvényt (3) alapján az

$$\hat{\alpha}_A^{(j)}(s) = \hat{\Theta}_A^{(j)}(s) - s - \frac{1}{2\pi}\int_0^{2\pi}\left\{\hat{\Theta}_A^{(j)}(s) - s\right\} ds$$

kifejezéssel, ahol

$$\hat{\Theta}_A^{(j)}(s) = \int_0^{2\pi} \chi(t)\,dt + \sum_{s_i \leqslant s} \Delta\alpha_i \qquad [0 \leqslant S \leqslant 2\pi]$$

szakaszonként folytonos, a görbe s_i paraméterű csatlakozási pontjainak megfelelő ugráspontokban pedig megállapodás szerint jobbról folytonos függvény. Ekkor az A síkidomhoz rendelt $\alpha \in \mathfrak{A}$ alakfüggvényt a (4) alatti $\hat{\alpha}_A^{(j)}$ segédfüggvény teljes számegyenesre történő, 2π szerint periódikus kiterjesztésével származtatjuk [5, 9].



1. ábra Vázlat a konturvonal görbületének elemzésén alapuló alakjellemzési módszer elvéhez [11]

Feltételezve, hogy az alakfüggvények Fourier sorukkal előállíthatók, az $\alpha_A^{(j)} \in \mathfrak{A}\,(-\infty < S < \infty)$ valós Fourier sorával megadható az

$$\alpha_A^{(j)}(s) = \sum_{M=1}^{\infty} V_n^{(j)} \sin\left[ns + \varphi_n^{(j)}\right] \tag{5}$$

alakban, ahol

$$V_n^{(j)} = \sqrt{[a_n^{(j)}]^2 + [b_n^{(j)}]^2} \tag{6}$$

$$\operatorname{tg} \varphi_n^{(j)} = \frac{a_n^{(j)}}{b_n^{(j)}} \tag{7}$$

és

$$a_n^{(j)} = \frac{1}{\pi} \int_0^{2\pi} \alpha_A^{(j)}(s) \cos ns\,ds, \qquad b_n^{(j)} = \frac{1}{\pi} \int_0^{2\pi} \alpha_A^{(j)}(s) \sin ns\,ds \tag{8}$$

$(n = 1, 2, \ldots)$

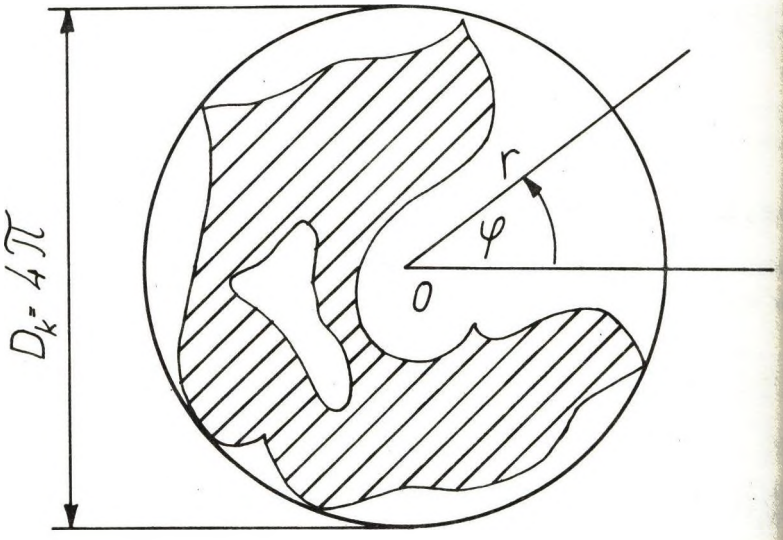
A (6) szerint $V_n^{(j)}$ együtthatókról belátható, hogy kielégítik az alakvektor komponen szemben támasztott követelményeket, függetlenek j-től, azaz a görbeparaméterezéstől ($V_n^{(j)}$ ha $\alpha_A^{(j)} \in \mathfrak{Q}$). A Fourier együtthatók tulajdonságából következően a V_n mennyiségek $n \to \infty$ e zérushoz tartanak, ezért a nagyobb indexűek alakjellemzésben betöltött szerepe elhanyago Vizsgálatainkban az első 16 darab V_n (n = 1, 2, ... 16) mennyiségből képzett alakvektort ha tuk osztályozási feladatok megoldásához.

Az alakfüggvények komplex, ill. valós Fourier együtthatóiból kiindulva különféle ala ra alkalmas mérőszámok képezhetők, ilyenek a szakirodalomban [9, 10] nagy számban talá A módszer előnye, hogy az alakvektor előállítása előzetes adatszelekció elvégzését nem igén számítógépre jól programozható.

2.3 Alakjellemzés kétdimenziós Fourier analízis alapján

A 2.2 pontban taglalt módszer többszörösen összefüggő alakzatok morfológiai leírás alkalmazható. Ennek igénye azonban, ha ritkábban is, de felmerül a kvantitatív metallográfi korlatban. Többszörösen összefüggő tartománnyal reprezentált mikroszkópos részecskék a lemzésére a kétdimenziós ortogonális függvényrendszer szerinti sorbafejtésen alapuló eljárá [3, 4, 5] alkalmazása tűnt célravezetőnek. Ezek közé sorolható a következőkben vázolt m szer is [12].

Helyezzük a síkidom súlypontját a 2. ábra útmutatása szerint egy polárkoordináta re pólusába. A mérethatás kiküszöbölése végett egy körlemezzel, melynek középpontja a polá szer pólusával megegyező, fedjük le a síkidomot úgy, hogy a körlemez és a síkidom határol lainak létezzen közös pontja. Végül normáljuk a befoglaló körlemez D_k átmérőjének nagys 4π-re.



2. ábra Vázlat a kétdimenziós Fourier-analízisen alapuló alakjellemzési módszer elvé

Definiáljuk az A síkidomra vonatkozóan a $[0 \leqslant \alpha \leqslant 2\pi] \times [0 \leqslant r \leqslant 2\pi]$ intervallumban értelmezett karakterisztikus függvényt a szokásos módon

$$\hat{f}_A(\varphi,r) = \begin{cases} 1 & \text{ha } (\varphi,r) \in A \\ 0 & \text{ha } (\varphi,r) \notin A \end{cases} \tag{9}$$

Legyen $f_A(\varphi,r)$ az $f_A(\varphi,r)$ karakterisztikus függvény teljes síkra való 2π szerint kétszeresen periódikus kiterjesztése. Az $f_A(\varphi,r)$ négyzetesen integrálható $[0 \leqslant \varphi,r \leqslant 2\pi]$-ben, ezért a négyzetintegrára vonatkozó konvergencia értelmében kétdimenziós Fourier sorával a következő alakban állítható elő [13]:

$$f_A(\varphi,r) = \sum_{m,n=0}^{\infty} \lambda_{m,n} \left\{ a_{m,n} \cos m\varphi \cos nr + b_{m,n} \sin m\varphi \cos nr + \right.$$
$$\left. + C_{m,n} \cos m\varphi \sin nr + d_{m,n} \sin m\varphi \sin nr \right\} \tag{10}$$

Az $a_{m,n}$, $b_{m,n}$, $c_{m,n}$ és $d_{m,n}$ együtthatók [13] felhasználásával olyan mennyiségek állíthatók elő, amelyek invariánsak a síkidom súlypont körüli elforgatásával és a súlyponti egyenesre vonatkozó tükrözésével szemben. Belátható, hogy ilyen tulajdonságúak az alábbi kifejezésekkel definiált

$$\begin{aligned} V_{m,n,1} &= a_{m,n}^2 + b_{m,n}^2 \\ V_{m,n,2} &= c_{m,n}^2 + d_{m,n}^2 \\ V_{m,n,3} &= a_{m,n} C_{m,n} + b_{m,n} d_{m,n} \\ V_{m,n,4} &= |a_{m,n} d_{m,n} - b_{m,n} C_{m,n}| \end{aligned} \tag{11}$$

mennyiségek is, amelyekre fenn áll a

$$V_{m,n,1} V_{m,n,2} = V_{m,n,3}^2 + V_{m,n,4}^2$$

összefüggés. Következésképp a (11) alatti mennyiségek alakvektor komponenseinek képzésére felhasználhatók [12]. Az eljárás hátrányai között meg kell említeni a polárkoordináta rendszer alkalmazásával járó szokásos negatívumokat, továbbá azt, hogy a polártartomány nagyobb számú részre való felosztásával és a Fourier együtthatók számának növelésével a módszer meglehetősen számításigényessé válik.

3. Gyakorlati alkalmazási eredmények

Vizsgálataink fontosabb gyakorlati eredményei zömmel a 2.2 pontban taglalt alakjellemzési módszer – görbület analízisen alapuló – c) típusú változatának alkalmazásához fűződnek, ezért a következőkben csak ezek ismertetésére szorítkozunk.

A számítógépes vizsgálatokhoz előzetesen digitalizált, kéttónusú képeket használtunk. A digitális képek elemzéséhez számítógépi programot készítettünk, amelyet TPA I típusú számítógépen futtatunk.

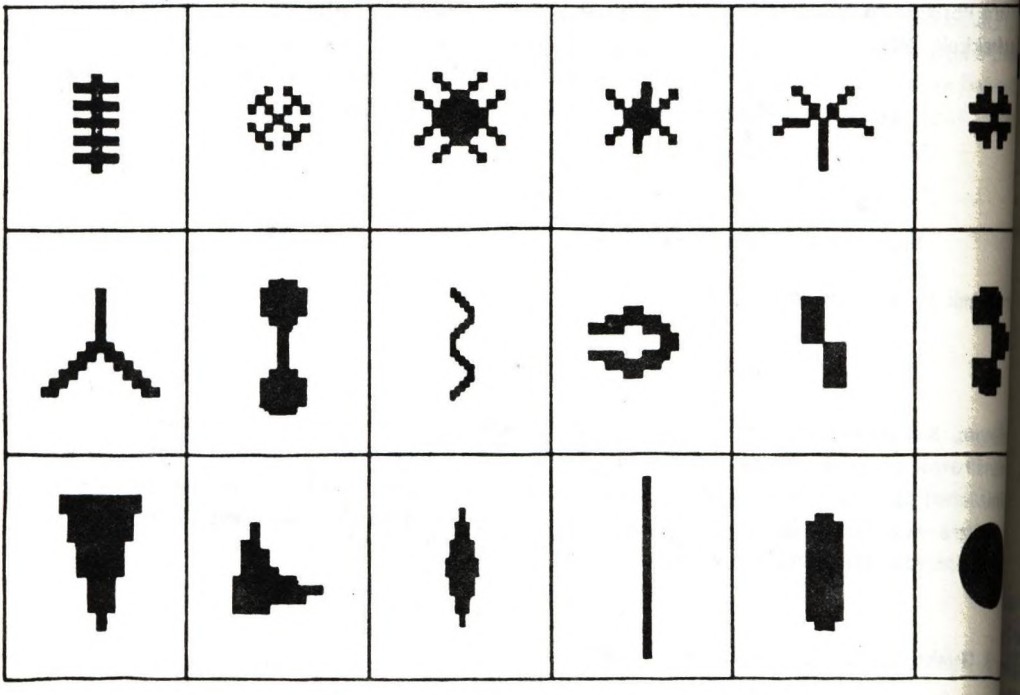
A program segítségével elvégezhető a képen egymáshoz képest izoláltan elhelyezkedő,

topológiailag összefüggő részecskék (síkidomok) azonosítása (egyedi címzése); kontúrkövető algoritmussal meghatározhatók a (4) kifejezéssel definiált alakfüggvények ill. a megfelelő alakvektorok.

Mint korábban említettük, a vizsgálatokhoz 16 komponensű alakvektorokat alkalmaz... Az alakvektor előállításakor a (6) szerint definiált V_n együtthatók szükséges minimális szá... többek között a Parseval formulára támaszkodva becsültük. Tapasztalataink szerint 16 Fou... együtthatópár figyelembevételével a bonyolult alakú mikroszkópos részecskék képe is kielé... mértékben rekonstruálható.

Gyakorlati célú vizsgálataink középpontjában a kvantitatív metallográfiában leggyakra... előforduló jellegzetes alaktípusok kiválasztása, valamint az automatikus osztályozás céljára ... gáló „alakosztályok" származtatása állt. E kettős feladat megoldására a numerikus taxonó... ból [14] ismert hierarchikus osztályok képzéséra alkalmas módszereket alkalmaztuk. A vizs... latok eredményeképp sikerült a gyakoribb alaktípusokat (jellegzetes geometriájú síkidomok... kiválasztani, majd morfológiai elemzésüket követően olyan „típusalakzatokat" konstruálni, ... amelyek a valóságos mikroszkópos részecskék főbb alaktípusait képviselik.

A 2. táblázatban foglalt 18 típusalakzat mindegyike egy-egy alakosztályt definiál. A ... pusalakzatokat, – abból a célból, hogy geometriájuk egyszerűen megadható legyen, és alak... vektoraik komponenseit pontosan számíthassuk –, szándékosan négyzet elemekből állítottu...



2. táblázat Az alakosztályokat definiáló típusalakzatok

Az ismert alakvektorral rendelkező típusalakzatok és az (1) szerinti távolságmérő függ... felhasználásával lehetőség nyílik arra, hogy a részecskék alak szerinti osztályozását a szokás... korlattól eltérően kvantitatív ismérvek alapján és objektív körülmények között valósíthassuk ...

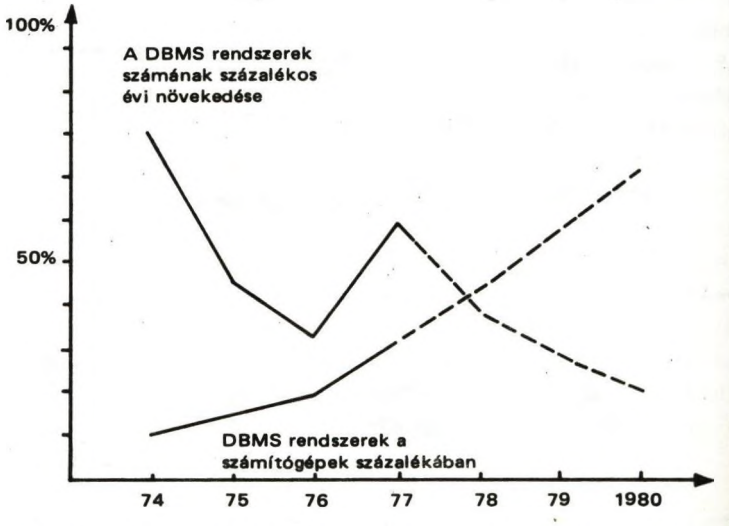
1] Underwood, E.E.: Quantitative Stereology, Addison–Wesley, Menlo Park, 1970.
2] Young, T.Y., Calvert, T.W.: Classification, Estimation and Pattern Recognition, American Elsevier Publishing Co., Inc., New York, 1974.
3] Rosenfeld, A., Kak, A.C.: Digital Pictura Processing, Academic Press, New York, 1976.
4] Rosenfeld, A.: Picture Processing by Computer, Academic Press, New York, 1969.
5] Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis, John Wiley and Sons, New York, 1973.
6] Fischer, C.: The New Quantimet 720, The Microscope, 19 (1971) 1–20.
7] Jaglom, I.M., Boltyanszkij, V.G.: Vüpuklüe figurü, Gosztyeizdat, Moszkva, 1956.
8] Beddow, J.K., Philip, G.: On the Use of Fourier Analysis Technique for Describing the Shape of Induvidual Particles, Planseeberichte für Pulverment. 23 (1975) 3–14.
9] Zahn, Ch. T., Roskies, R.Z.: Fourier Descriptors for Plane Closed Curves, IEEE Trans. Computers, C–21. 3 (1972) 269–281.
0] Granlund, G.H.: Fourier Preprocessing for Hand Print Character Recognition, IEEE Trans. Computers, C–21. 2 (1972) 195–201.
1] Sternberg, S.: Lectures on Differential Geometry, Princetic Hall Ind. Englewood Cliffs, 1964.
2] Réti, T.: Publikálatlan kézirat
3] Tolsztov, G.P.: Rjadü Furje, Gaszudarsztvennoe Izdatyelszvo Fiziko-matematicseszkoj Literaturü, Moszkva, 1960.
4] Sokal, R.R., Sneath, P.H.: Principles of Numerical Taxonomy, San Francisco, 1963.

AZ ADATBÁZISKEZELÉS LEHETŐSÉGEI ÉS PROBLÉMÁI MAGYARORSZÁGON

Simonfai László
SZÁMKI

1. Bevezetés

Aligha kétséges, hogy az adatfeldolgozás jövőjét döntő mértékben befolyásolni fog datbáziskezelés. Az USA-ban 1976–77-ig mintegy 4000 adatbáziskezelő rendszert instal az alkalmas konfigurációk 20%-át jelenti. [1] Világszerte installált rendszerek száma 5- tehető. Egyes források szerint [2] az USA-ban 1982-re az alkalmas konfigurációk 75% nálnak adatbáziskezelő rendszert. Más adatok szerint [3] 1979-ben a nagyobb IBM gép án használják a technológiát, szemben az 1976-os 20%-kal. (1. ábra.) Várható a kisgép báziskezelő rendszerek növekvő szerepe is. [5] [6] [7].



1. ábra Közepes és nagy IBM számítógépeken installált vagy installálásra tervezett adatbáziskezelő rendszerek

A kezdeti próbálkozások, hibák után tehát az adatbáziskezelés gyors elterjedés mert a növekvő komplexitású feladatok megoldásához szükség van az adatok – mint erőforrások – központosított tárolására és ellenőrzésére; a jelenleg különálló, de nem len alkalmazások integrálására; a tárolt adatok osztott használatára; a változó és bővü mazási igényeknek a változó hardware/software környezetben történő, viszonylag zök tes kielégítésére; az adatok integritásának és konzisztenciájának fenntartására.

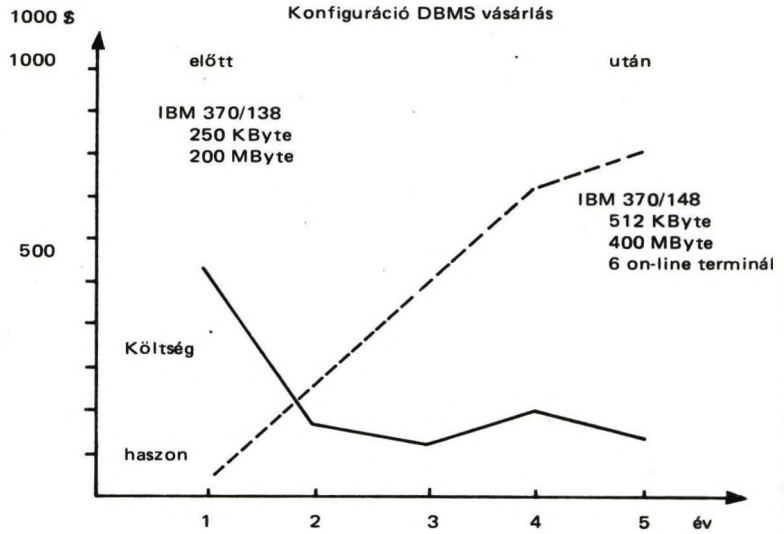
A számítógépalkalmazás növekedését és elterjedését vizsgálva az USA-ban, Nol fázisból álló modellt alkotott [3] [4]. Az egyes fázisok legfontosabb jellemzőit a Tá foglalja össze.

A számítógépalkalmazás fázisai

		1. fázis	2. fázis	3. fázis	4. fázis	5. fázis	6. fázis
		Kezdeti szakasz	Járványszerű terjedés	Szabályozott terjedés	Integrálás	Adminisztráció	Érett szakasz
Általános jellemzés		Első költségcsökkentő alkalmazások. Specializálódás, a technológia megtanulására.	Költségcsökkentő alkalmazások elterjedése. Felhasználó orientált programok.	Meglévő alkalmazások konszolidálása, dokumentálás, szabványok.	Átállás adatbáziskezelésre, on-line alkalmazások.	Integrált alkalmazások.	Információ áramlást tükröző alkalmazás-integrálás
Az információ felhasználása (%)	operatív vezetés	100	85	80	65	55	45
	vállalati irányítás	–	15	20	30	35	40
	távlati tervezés	–	<1	<1	5	10	15
Felhasználók szerepe		Korlátozott: a számítógép több, jobb, gyorsabb információt szolgáltat mint a manuális módszerek.		Hajtóerő: a végfelhasználó részt vesz az adatbevitelben és az adatokat közvetlenül használja. Felelős az adatok minőségéért.		Részvétel: a végfelhasználó és az adatfeldolgozó részleg együttesen felelős az adatok minőségéért és a hatékony alkalmazások tervezéséért.	
Az adatfeldolgozó részleg feladata		Központosított adatfeldolgozás. „Closed shop" üzem		Adatfeldolgozás és adattárolás. Szolgáltatás, megbízható üzem.		Az adaterőforrás kezelése. Többszintes, decentralizált szervezet	
Számítógép használat (%)	bacth	100	80	70	50	20	10
	RJE		20				
	adatbázis kezelés			15	40	60	60
	mini/mikro számítógépek				5	15	25
	egyéb			15	5	5	5

A táblázatban feltüntetett jellemzők alapján megállapítható, hogy egy adott vállalat fejlődésének – számítástechnikai szempontból – melyik fázisában van, és ez támpontot ad a reális továbblépéshez szükséges legfontosabb döntések meghozásához.

A modell szerint a 3. fázisban a célkitűzésekben lényeges változás következik be, a hangsúly a számítógépre orientált vezetésről az adaterőforrásra orientált vezetésre tolódik át, napirendre kerül az adatbázis technológia konkrét bevezetése. Az USA vállalatainak mintegy fele található ma a 3. vagy 4. fázisban. A 2. ábrán adatbáziskezelő rendszer vásárlásának becsült költségei és várható haszna látható a közepes (50 millió dollár évi forgalmú) amerikai vállalat esetében. [2]



2. ábra Adatbáziskezelő rendszer vásárlásának becsült költségei és várható haszna

2. Áttérés adatbáziskezelésre

Érdemes behatóbban megvizsgálni, hogy miért jelentkezik egyre kényszerítőbb er alapvetően konszolidációs jellegű 3. fázisban az adatbáziskezelésre való áttérés igénye. A ző fázisban egyedi alkalmazások készültek, általában tapasztalatlan programozók munk Független, összehangolatlan, nagy méretű redundanciát tartalmazó file-ok jöttek létre, lyek egyre növekvő mennyiségű, ugyanakkor egyre kevésbé pontos és időszerű adatot maznak. A file-ok csak statikus, rutinszerű információs igényeket támogatnak, karbantartásuk egyre nehezebb. Az alkalmazási programok karbantartása a 3. fázisban rendelkezésre álló munkaerő 70—80%-át veszi igénybe. Ugyanakkor az operatív irányít szintjein lévő alrendszerek nem alkalmasak magasabb szintű funkciók (pl. rendelés fel zás, termelésirányítás) támogatására. Ez az a pont, ahol mind a vállalatvezetés, mind az adatfeldolgozó részleg oldaláról megkezdődik a felkészülés az adatbáziskezelésre va résre. A vezetésnek fel kell ismernie, (és megfelelő intézkedéseket kell hozni) hogy:

— az adatbázis a vállalat egésze számára készül, nem pedig az egyes funkcionál gek vezetői részére;

— az adatbázis létrehozatalát és fejlesztését tervezni kell;

— a vezetés feladata az információ szükséglet meghatározása;

— az adatfeldolgozó szervezet feladata az információk előállításához szükséges a meghatározása és tárolása;

— az adatfeldolgozó szervezetet megfelelő hatáskörrel kell felruházni, hogy az i adatok pontossága, időszerűsége és konzisztenciája biztosítható legyen;

— az adatbáziskezelés hasznát azon kell lemérni, hogy a vállalat céljainak elér hatékony működéséhez mennyiben járul hozzá [8].

A folyamatban lévő licenc vásárlás eredményeként az adatbázis-kezelés lehetősége az eszköz, a technológia szintjén adott. Ugyanakkor, azok az okok, amelyek a növekedés 2. fázisában a jelenlegi alkalmazási rendszerek kialakulásához vezettek, nem szűntek meg. Ezért olyan környezet, vállalati és adatfeldolgozási szervezet kialakítására van szükség, amely képes az adatbáziskezelés lehetőségeinek kihasználása – ennek hiányában csak a már meglévő problémák termelődnek újra, a jelenleginél lényegesen magasabb hardware/software és személyi költségek mellett. Az új környezet létrehozása, a megfelelő szemléletmód kialakítása az adatbáziskezelés első számú problémája hazánkban.

3. Az adatbázis adminisztrátor

Az adatbáziskezelésre való áttérés során az adatfeldolgozó szervezet is mélyreható változáson megy át, új funkció jelentkezik, az adatbázis adminisztrátor funkciója. [9] [10] [11] [12]. Az adatbázis adminisztrátor tevékenységi körébe az alábbiak tartoznak:

1. Adatbázistervezés és tanácsadás.

Az adatbázis adminisztrátor feladata olyan adatstruktúra kialakítása, amely egyrészt kielégíti a felhasználási igényeket, másrészt biztosítja a rendszer-erőforrások hatékony kihasználását.

2. Érdekegyeztetés.

Az adatbázis létrehozatala és használata során fellépő, gyakran erősen ellentmondó felhasználói igények összhangba hozatala, az érdekek egyeztetése, az adatbázis adminisztrátor segítségével valósul meg, aki a teljes adatbázist, – az Egészt – képviseli a partikuláris érdekekkel szemben.

3. Adatbiztonság, helyreállítás és újraindítás.

Az adatbázis adminisztrátor felelősségi körébe tartozik azon programtechnika és szervezési eljárások kifejlesztése, amelyek garantálják a szükséges mértékű adatbiztonságot, továbbá az adatok integritását a felmerülő hardware, alapsoftware és alkalmazási software hibák esetén is.

4. Kapcsolat a szállítókkal.

Az adatbázis adminisztrátor tartja fenn a kapcsolatot a hardware és software szállítókkal az adatbázissal összefüggő problémák vonatkozásában.

5. Képzés és továbbképzés.

Az adatbázis adminisztrátor biztosítja az adatbázist használó személyzet házonbelüli kiképzését és továbbképzését.

6. Szabványok, rendszer bevizsgálás.

Az adatbázis tervezésének, programozásának dokumentálásának és tesztelésének felügyelete, illetve a munkákban való részvétele útján az adatbázis adminisztrátor adatbázis szabványokat hoz létre, gondoskodik a szabványok betartásáról és betartatásáról. Új rendszer integrálása során ellenőrzi azokat az előírásokat és specifikációkat, amelyeknek a kielégítése az üzemszerű használat előfeltétele.

7. Konverzió és integrálás.

Meglévő alkalmazások adatbázisra történő áttelepítése során az adatbázis adminisztr gondoskodik a konverzió elvégzéséről, az integrálás zökkenőmentes lebonyolításáról. Enn kapcsán szükség lehet az adatbázis újrastruktúrálására, a dokumentáció update-elésére, a lévő szabványok betartásának ellenőrzésére, illetve a szabványok módosítására.

8. Hatékonyságvizsgálat.

Az adatbázis adminisztrátor folyamatosan vizsgálja az adatbáziskezelő rendszer hatá ságát, és a rendszer hangolása révén gondoskodik a hatékony működésről.

9. Költségtervezés, fejlesztés.

Az adatbázis adminisztrátor vizsgálja, szétosztja és szabályozza az adatbázissal kapc tos költségeket. Együttműködve az adatbázis felhasználóival és a vállalat vezetőségével, r vesz az adatbáziskezelő rendszer közép- és hosszútávú fejlesztési terveinek kidolgozásában

10. Karbantartás.

Az adatbázis adminisztrátor felelős az adatbázis folyamatos karbantartásáért, elvégz szükséges software javításokat és módosításokat, és szabályozza az új változatok használ

A felsorolt tevékenységek ellátásához az adatbázis adminisztrátornak (illetve a funk betöltő szervezetnek) elsősorban jó adatbázis szakembernek kell lennie [1] [3] [5] [6] [7] [8] [10], ugyanakkor rendelkeznie kell vezetési [2] [4] [6] [9] és adminisztratív [5] [6] képességekkel.

Az adatbázis adminisztrátor általában nem egyetlen személy, hanem szervezet. Meg nésével mind az adatfeldolgozó szervezetnek a vállalaton belül elfoglalt helye, mind pedi adatfeldolgozó szervezet maga mélyreható változáson megy át. A lehetséges szervezeti m dások tekintetében az irodalomra utalunk [10] [11].

Az adatbázis adminisztrátori funkció betöltésére alkalmas szakemberek kiképzése, a séges tapasztalatok összegyűjtése és továbbadása, a megfelelő szervezeti keretek kialakítás lenti az adatbáziskezelés második nagy problématerületét hazánkban.

4. Módszerek és eszközök

Nem nehéz felismerni, hogy az adatbázis adminisztrátornak — lehetőség szerint szá géppel segített — eszközökre és módszerekre van szüksége funkciója sikeres betöltéséhez.

1. Az adatszótár

Az adatbázis adminisztrátor legfontosabb eszköze az adatszótár [9] [10] [12]. Az a szótár tartalmazza valamennyi adatelem definícióját, jelentését, származását, kapcsolatait, lamint a felhasználás helyét és módját. Jól használható mind a szándékolt változtatások tásának vizsgálatára, mind pedig a felfedett hibák eredetének behatárolására. Az alábbiak összefoglaljuk az adatbázis adminisztrátor legfontosabb tevékenységeihez felhasznált adat tár adatokat.

Tevékenység	Felhasznált adat, keresztreferencia adatelemek, és kapcsolataik, jelentésük, a külső file-ok leírása, sémák, alsémák;
tervezés	
integritás, konzisztencia	adatelemek szerkesztési előírásai, az adatelemek előállításáért felelősök neve;
titkosság	titkossági kulcsok és zárak, hozzáférési jogok, alsémák;
konverzió	külső file-ok leírása, séma;
hangolás	futási és helyfoglalási statisztikák;
átszervezés	érintett alsémák, programok, titkossági követelmények;
hibajavítás	a hibás adatokat létrehozó, ill. megváltoztató programok azonosítása, a készítő megnevezése, az utolsó módosítás dátuma, programleírások;
programfutások ütemezése	alsémák, az érintett AREA-k, a futások szükséges periodicitása.

Minthogy az adatszótár az adatbázisra vonatkozó adatokat tartalmazza, meta-adatbázisnak tekinthető. Előnyösen használható nemcsak az adatbáziskezeléssel, hanem hagyományos file-rendszerekkel kapcsolatban is.

2. Tervezési eszközök

Az adatbázis tervezés részproblémáinak megoldására viszonylag sok módszer alakult ki, de – mint Sundgren megállapítja [13]: „... (a vázolt eljárás) feltételezi, hogy az adatbázis tervezésének létezik jól integrált, általánosan elfogadott, normatív elmélete. Ez nem igaz..."

A tervezési folyamat a követelményanalízisből indul ki. Adatbázis-specifikus software engineering módszerek még nem alakultak ki, bár alkalmazásuk kívánatos lenne. [14] A logikai tervezés célja – ANSI (X3) SPARC terminológiáját használva [15] – a fogalmi séma és a külső sémák létrehozása. A belső séma a fizikai tervezés eredménye. A fejezetben néhány, a gyakorlati alkalmazáshoz közel álló eszközt, ill. módszert ismertetünk.

A fogalmi sémához vezető út alapvető fontosságú lépése a normalizálás.

Az eljárást a relációs adatmodell kapcsán fejlesztették ki, de teljesen általános érvényű, jól használható mind hagyományos file-rendszerek, mind pedig adatbázisok tervezésénél. Ha a tervezés további lépései során – hatékonysági megfontolások miatt – el is kell térni a 3. normál formától (pl. ismétlő csoportok bevezetése miatt), a lényeges funkcionális függések már ismertek, ezért megsértésük elkerülhető. A 3. normál forma előállítására hatékony algoritmust fejlesztettek ki [16], és ennek adatszótárra épülő implementálása kívánatosnak látszik. A területen élénk kutató munka folyik, mind az algoritmusok hatékonyságának megjavítására, mind pedig a 3. normál forma hiányosságainak kiküszöbölésére (4. normál forma, hierarchikus dekompozíció) [17].

A logikai adatbázis-tervezés területén egyetlen üzemszerű használatban lévő automatikus segédeszköz ismert: az IBM Database Desing Aid (DBDA) nevű programcsomagja [18]. A termék az IMS adatbáziskezelő rendszerhez illeszkedik, hierarchikus adatstruktúrák létrehozatalát támogatja batch üzemmódban. A tervezés kiinduló pontját az egyes alkalmazások által kívánt adatelemek, az elemek közötti funkcionális függések és az adatok használati gyakoriságára vonatkozó adatok alkotják, végeredmény egy hierarchikus séma.

Az Irani [19] által javasolt eljárás lényegében a DBDA-hoz hasonló input alapján optimális CODASYL sémát generál. Az egyetlen befolyásolható paraméter az adatkapcsolatok megvalósításának módja: duplikálás, ismétlő csoport vagy CODASYL halmaz, összesen 24 le-

hetőség áll rendelkezésre. A módszer átfogja a tervezés logikai és fizikai fázisát, céljai azon túlságosan szűkkörűek.

A gyakorlati alkalmazhatósághoz legközelebb azok a módszerek állnak, [20], [21] amelyek elegendően sok paramétert vesznek figyelembe, de optimalizálásra nem törekednek. Ezek a módszerek egy adott tervváltozatot értékelnek ki, háttértár és feldolgozási idő szükséglet kintetében. Az adatbázis adminisztrátor feladata annak eldöntése, – célszerűen on-line üzemmódban dolgozva – hogy mely paraméterek megváltoztatásával lehet a tervváltozatot javítani. A Gerritsen által javasolt eljárás a fogalmi sémából és a feldolgozási igényekből indul ki, a változtatható paraméterek: elhelyezési mód, halmaz implementálás, szinguláris halmazok alkalmazása, a felhasznált operatív, – és háttértár mérete. Vizsgálhatók tehát a fizikai terv variánsai adott fogalmi séma mellett, de a fogalmi séma módosításával további tervváltozatok is előállíthatóak.

A CODASYL 78 tárolási sémája az eddigieknél lényegesen több lehetőséget ad az adatbázis adminisztrátornak a futási hatékonyság befolyásolására. A jelenleginél átfogóbb, többet nyújtó segédeszközökre lesz szükség a fizikai tervezéshez. [22]

Az eszközök és módszerek területén található nyitott kérdések megválaszolása, koherens az adatszótár köré csoportosított, egymásra épülő eszközöket felhasználó módszertan kidolgozása és gyakorlati alkalmazása az adatbáziskezelés harmadik problémája.

5. Összefoglalás

Az adatbáziskezelés lehetősége a folyamatban lévő licenc vásárlás eredményeként az egyetemek szintjén adottnak tekinthető hazánkban. A technológia bevezetésével kapcsolatban az alábbiak három problématerületet emel ki:

1. az adatbáziskezelésre való áttérés vállalati feltételei;
2. az adatfeldolgozó szervezet átalakulása, az adatbázis adminisztrátori funkció megjelenése;
3. az adatbázis adminisztrátor számítógépes eszközei.

A számítógépalkalmazás Nolan-féle modellje támpontot nyújt annak meghatározásához, hogy a vállalatoknál mikor jönnek létre a technológia bevezetésének reális lehetőségei. Az adatbázis adminisztrátor funkcióinak összefoglalása a humán feltételek biztosításához, egy új típusú adatfeldolgozó szakember kialakításához kíván hozzájárulni. A befejezés az adatbázis adminisztrátor eszközei közül azokat tekinti át, amelyek számítógéppel segíthetők, és már a közeli jövőben gyakorlati alkalmazásba kerülhetnek.

[1] Martin, D.: Qu'est-ce qui freine le développement des SGBD? ol informatique. avril 1979. pp. 38-42.
[2] McFadden, F.R.; Suver, J.D: Costs and benefits of a data base system. Harward Business Review. Nº–1. 1978. pp. 131-139.
[3] Nolan, R.: Managing the crises in data processing Harward Business Review. Nº–2. 1979. pp. 115-136.
[4] Nolan, R.: Managing the Computer Resource: A Stage Hypothesis. CACM July 1973. pp. 399-405.
[5] Simonfai, L.; Horváth J.: MADAM adatbáziskezelő rendszer R10 számítógépen. SZÁMKI tanulmányok. Nº–1. 1977. pp. 125-145.
[6] Bánkfalvi J.; Bánkfalvi Zs.; Békéssy, P.; Buzder L.J.; Horváth, J.; Simonfai, L.: Adatbázis-kezelés R10 COBOL-ból. Programozási Rendszerek '78. I. kötet pp. 58-65.
[7] Slonim, J.; Farrel, M.W.; Fisher, P.S.: A survey of mini-data base management systems in 1977. SIGMINI Newsletter. Aug. 1978. pp. 26-34.
[8] Appleton, D.S.: What Data Base Isn't Datamation, Jan. 1977. pp. 85-91.
[9] Lyon, J.K.: The database administrator Wiley & Sons, 1976.
[10] Halassy, Béla Dr.: Adatbázisok kezelésének alapvető kérdései. SZÁMOK, Budapest, 1978.
[11] –: IDMS Database Design and Definition Guide Cullinane Corp. 1977.
[12] Lefkovits, H.C.: Data Dictionary Systems Q.E.D Publication, 1977.
[13] Sundgren, B.: Data Base Design Theory and Practice 4th International Conference on Very Large Data Bases West Berlin, 1978. pp. 3-16.
[14] Wasserman, A.I.: A Software Engineering View of Data Base Management. 4 th International Conference on Very Large Data Bases West Berlin, 1978. pp. 23-35.
[15] Tsichritzis, D.; Klug, A.: The ANSI/X3/SPARC DBMS framework Information Systems, Nº–3. 1978. pp. 173-191.
[16] Beeri, C.; Bernstein, P.A.: Computational Problems Related to the Design of Normal Form Relational Schemas ACM TODS, March, 1979. pp. 30-59.
[17] Beeri, C.; Bernstein, P.A.; Goodman, N.: A sophisticate's introduction to database normalization theory. 4 th International Conference on Very Large Data Bases West Berlin, 1978. pp. 113-124.
[18] Raver, N.; Hubbard, G.U.: Automated logical data base design: Concepts and applications. IBM Systems Journal, Nº–3, 1977. pp. 287-312.
[19] Berelian, E.; Irani, K.B.: Evaluation and optimization. 3 rd International Conference on Very Large Data Bases Tokio, 1977. pp. 545-555.
[20] Gambino, T.J.; Gerritsen, R.: A Data Base Design-Decision Support System. 3 rd International Conference on Very Large Data Bases Tokio, 1977, pp. 534-544.
[21] Merten, A.; Oberlander, L.: A view of database structure design. 1 st Int. Computer Software & Application Conference Chicago, 1977. pp. 252-257.
[22] –: Report of the CODASYL Data Description Language Committee. Information Systems, Nº–4, 1978. pp. 247-320.

NYELVORIENTÁLT SZÁMÍTÓGÉPEK UTASÍTÁSKÉSZLETÉNEK TERVEZÉSE*

Simor Gábor
SZKI

Kivonat:

Minden magasabbszintű programozási nyelvhez meghatározható annak ún. Kanonikus É telmezési Formája, amely alapján egy gépi utasításkészletet realizálva az ezen a nyelven írt pro ramokat optimális áteresztőképességgel végrehajtó számítógéphez juthatunk. A cikk megfoga mazza a Kanonikus Értelmezési Forma szerepét és szabályait, beszámol e megközelítésnek egy COBOL architektúra kialakítására irányuló alkalmazási kísérletéről és leírja az erre a megköze lítésre alapozott utasításkészlettervezés általános folyamatát.

1. Bevezetés

A jelenleg is kompatibilitási alapként szolgáló „hagyományos" számítógép utasításkészle számos megoldásukban (pl. univerzális műveletkészlet és regiszterek, bázisregiszteres címzés, kötött mező és adathosszúságok stb.) a 15—20 évvel ezelőtti technológiai szintet tükrözik. A magasabbszintű programozási nyelvekről e kötött architektúrákra történő kódgenerálásnak az op malizálása sokáig a rendszerprogramozás és a hatékony eszközkihasználtság egyik központi pro lémája volt, amely az utóbbi időben — különösen a „software-krízis" kirobbanása után — néme háttérbe szorult.

Másrészt viszont a számítógép-architektúrák terén egyre meghatározóbbá válik két fejlő tendencia:

- a mikroprogramozás terjedésével, technológiájának fejlődésével, a mikroprogramozha eszközök olcsóbbá válásával egyre rugalmasabban alakíthatók ki az utasításkészletek;
- az osztott feldolgozás terjedésével megnő a dedikált feldolgozó egységek szerepe, ez így igényt is támaszt a speciális utasításkészletek iránt, ezen belül a magasabbszintű nyelvek feldolgozására orientáltak iránt is.

Az ilyen eszközökkel történő optimalizálás új jelentőséget nyer. A programszekvenciák mikroprogramszekvenciákkal helyettesíthetődnek, a fordítás-értelmezés határ megválasztására módunkban áll eldönteni például, hogy mire, mennyi gyors hozzáférésű lokális tárolót haszná lunk és hol, milyen funkciókat párhuzamosítunk. Ezáltal az utasításrendszer-tervezés során ho zott döntések sokkal „hatékonyság-értékenyebbek" [1].

Nyelvorientált speciális processzorok tervezésénél más megközelítést kell alkalmaznunk, a hardware-konfigurációban elfoglalt helye szempontjából is „felhasználó-közeli" (pl. termi ba vagy különálló mikrogépbe épített) feldolgozó egységről van szó, és megint mást, ha egy szá mítógép központi erőforrásai közé tartozó nyelvfeldolgozó segédprocesszorról. Az előbbi ese ben elsődleges szempont a megvalósítás egyszerűsége, a beépített hardware eszközök minimá mennyisége, a működés interaktív volta; az utóbbinál előtérbe kerül a feldolgozás hatékonysá gának kérdése.

* Az itt közölt ismertetés az OMFB által támogatott Tudományos Kutatási Munkák 2. fejezete keretében végzett tevékenység eredményei alapján készült.

A felhasználás általánossága szempontjából a tervezési módszer függ attól is, hogy az utasításrendszer „intelligenciaszintjét" általában növeljük-e a magasabbszintű programozási nyelvek felé (amire jó példa lehet a VAX–11/780, vagy a HB 66-os modell), vagy pedig csak egy meghatározott forrásnyelvhez optimális architektúra kialakítása a cél. Ez utóbbi célkitűzést indokolják az egyszerre több, mikroprogramozott virtuális architektúrát szolgáltató gépek (pl. B1700, NCR Criterion) és a különböző programozási nyelvekhez fizikailag is dedikált, különálló segédprocesszorokat tartalmazó struktúrák (IBM „Future System": [3]).

Ebben a cikkben meghatározott forrásnyelvhez dedikált, központi erőforrásként használt processzorok utasításkészletének tervezési kérdéseivel foglalkozunk.

2. A magasabbszintű nyelvek Kanonikus Értelmezési Formája

Egy-egy meghatározott forrásnyelvhez optimális áteresztőképességet biztosító utasításkészletek kialakításának általános szabályait kutatta M. J. Flynn [1] egy univerzális emulátorgépre irányuló kísérleti projekt kapcsán. A szabályok általános alkalmazhatóságán – a tetszőleges forrásnyelvhez való alkalmazhatóságon túlmenően – itt azt kell érteni, hogy azok olyan feltételezésekkel élnek, amelyeket elfogadva a tervezés első menetében, ill, fázisaiban (ld. 4. fejezet) előálló megoldás a későbbiek során a konkrét feltételek alapján pontosítható. E feltételezések a következőket jelentik:

– egy adott feladatot végrehajtó program feldolgozása tekintetében egy számítógép áteresztőképessége jellemezhető azzal, hogy a program gépi kódú tárolásához mennyi operatívtároló-kapacitás, végrehajtásához mennyi utasításlehívás és egyéb operatívtároló-hozzáférés szükséges;

– a különböző forrásnyelvi elemek (mint például műveletfajták, adattípusok) egyforma gyakorisággal fordulnak elő;

– az utasításkészletet értelmező gép belső struktúrája, ill. mikroarchitektúrája nem implikál különösebb korlátozást a gyors hozzáférésű belső tároló méretére és szervezésére, a kezelendő adatmezők hosszúságának változtathatóságára, valamint a különböző utasításfeldolgozási fázisok végrehajtásának időbeli összevonhatóságára.

Az ilyen értelemben vett optimalitás biztosítása a célja annak az „ideális" – Kanonikus Értelmezési Formának nevezett – gépi nyelvnek, amelyet egy adott forrásnyelvből az 1. táblázatban összefoglalt szabályok segítségével kaphatunk.

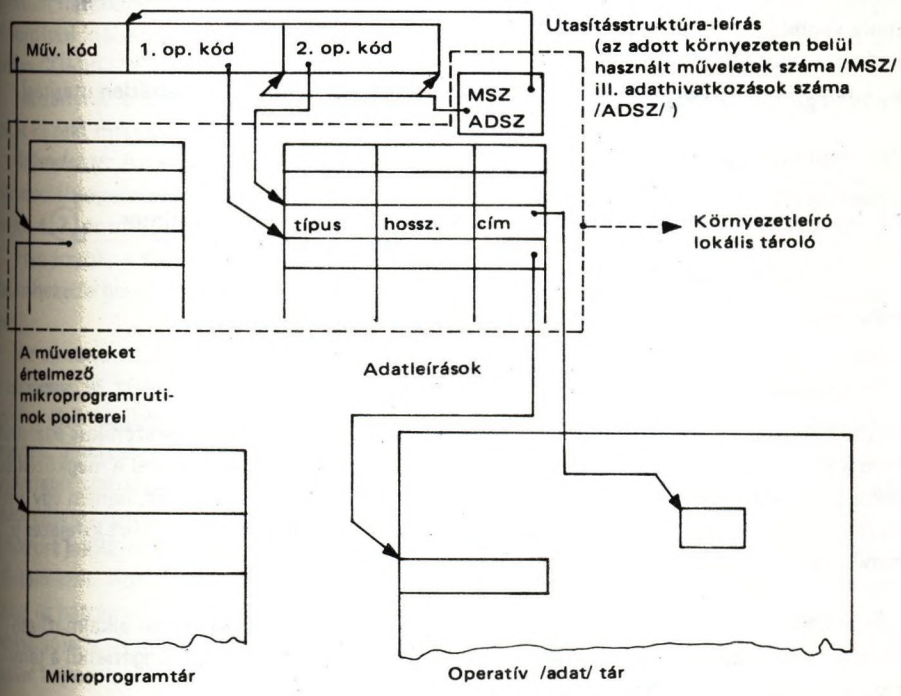
Az egyértelmű megfeleltetési szabályok érvényesülési tendenciái már megfigyelhetők egyes újabb univerzális architektúrákban is: pl. a VAX–11/780 modellben a forrásnyelvhez igazodó magasabbszintű gépi műveletek példája lehet a DO WHILE ciklus megvalósítása egyetlen gépi utasítással, az operanduskijelölések számát csökkentő megoldások példája pedig, hogy a leggyakoribb aritmetikai műveletek rendelkezésre állnak mind kétcímes, mind háromcímes utasítások formájában is.

Az operatív tárigény minimalizálását előíró szabályok (realizálásuk sémáját az 1. ábra illusztrálja) azon alapulnak, hogy minden programon belül kialakulnak viszonylag zárt „környezeek" (programozási nyelvtől függően szubrutinok, blokkok, modulok, eljárások stb.), amelyen belül nem kerül felhasználásra a gépi nyelv teljes műveletkészlete, hanem annak csak egy része, és viszonylag kevés adatelemnévre történik hivatkozás, tehát nem szükséges az operanduskód-mezőknek egy nagyobb összefüggő címtartományt átfogniuk.

1. sz. táblázat. A Kanonikus Értelmezési Forma kialakításának szabályai

1. Az egyértelmű megfeleltetés szabályai	2. A környezethez minimalizálás szabályai	3. A tárhozzáférések minimalizálásának szabályai
a) A forrásnyelven megjelölhető minden — nem hozzárendelést jelentő — művelet gépi nyelv szinten történő végrehajtásához elegendő gépi utasítás	a) Egy adott programkörnyezeten belül használt műveletek száma határozza meg a gépi utasítások műveleti kód mezőjének hosszúságát	a) A vezérlésátadást (procedurális műveletet) előíró gépi utasítások végrehajtásához nem szükséges tár hozzáférést végrehajtani (a következő utasítás lehívásán túlmenően)
b) Egy forrásnyelvi utasításban szereplő adatnevek gépi nyelvi szinten történő kezeléséhez elegendő az egymástól eltérő adatelemnevekhez tartozó operandusok kijelölése (azaz a különböző paraméterként, többször is szereplő adatelemnevek számára egyetlen operandus van fenntartva).	b) Egy adott programkörnyezeten belül használt különböző adatelemnevek száma határozza meg a gépi utasítások operandus kód mezőjének hosszúságát	b) A funkcionális műveleteket előíró gépi utasítások végrehajtásához csak az operandus mezők által kijelölt adatok kiolvasása, ill. beírása szükséges mint tárhozzáférés
		c) A program végrehajtása során a környezetek váltása egy-egy tárbeírást és tárkiolvasást követelhet meg
Tárigényben és végrehajtási időben	Tárigényben	Végrehajtási időben

eredményeznek megtakarítást e szabályok



1. ábra. Műveletkód- és operanduskód-feldolgozás a Kanonikus Értelmezési Formában

3. Egy COBOL-gép tervezése a Kanonikus Értelmezési Forma szabályai alapján

M. J. Flynn az általa javasolt szabályok alkalmazhatóságát egy FORTRAN-gép kialakításával próbálta ki [1] és azonos beépített eszközmennyiség mellett ötszörös áteresztőképességjavulást mért szabványos FORTRAN Benchmark programok futtatása során az IBM 370 utasításkészlethez képest. Ennek alapján azt vizsgáltuk, hogy mennyiben alkalmazható ez a megközelítés egy COBOL-gép tervezéséhez. COBOL esetén a Kanonikus Értelmezési Forma alkalmazásával kisebb hatékonyságjavulás volt várható, mivel

– a COBOL adatfeldolgozó jellegénél fogva a programok adat/kód helyfoglalási aránya az operatív tárolóban nagyobb az átlagosnál, a Kanonikus Értelmezési Forma pedig csak a kód hatékonyságának növelésére irányul;

– a COBOL viszonylag kevés műveletfajtát tartalmaz, így nyilván kisebb a jelentősége annak, hogy programkörnyezeten belül hány művelet fordul elő.

Az általunk tervezett COBOL-orientált utasításkészletre fordított COBOL etalon mérőprogram mérete harmada lett a hagyományos gépi utasításokra fordított programénál. A dinamikus jellemzők és a megvalósítás egyéb feltételeinek vizsgálatához jelenleg kifejlesztés alatt áll egy szimulátor erre az utasításkészletre.

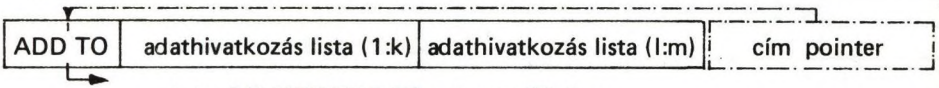
Az utasításkészlet meghatározásánál két alternatív megközelítésből indultunk ki:

a) *A forrásnyelv szintaxisát maximálisan közelítő utasításkészletet* úgy képzelhetjük el a legkönnyebben, hogy minden forrásnyelvű utasításhoz hozzárendelünk egy mikroprogram szubrutint, amely végrehajtja az utasítás funkcióját.

Forrásutasítás példa:

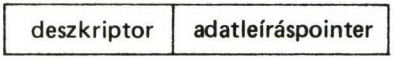
ADD $név_1$ [,$név_2$] ... [,$név_k$] TO $név_l$, ... [,$név_m$] [ON SIZE ERROR feltétlen utasítás]

A megfelelő gépi utasítás:*



1, ha ON SIZE ERROR volt specifikálva

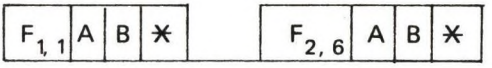
Egy adatnév mező (az adathivatkozáslista egy eleme):



A deszkriptor határozza meg, hogy globális vagy lokális adatról van-e szó; hierarchikus adatstruktúrák esetén a szintet; paraméterlista esetén az utolsó elemet mutatja. Sőt, ennél a megközelítésnél a deszkriptor azt is jelentheti, hogy a mező másik része nem adatleírás-pointert, hanem egy aritmetikai vagy logikai operátort jelöl, ezáltal egy utasításnak lengyel jelölésben felírt kifejezés is lehet paramétere.

b) *Az általános formátumokra orientált utasításkészlet* olyan általánosan alkalmazható formátumokon alapul, amelyek az utasításoknak csak a „vázát" adják meg, függetlenül a ténylegesen végrehajtandó műveletektől.

Példák általános utasításformátumokra:



, ahol

* – művelet, A, B – adatnevek

$F_{i,j}$ – i-paraméterű műveletekhez a j-ik formátum típuskódja például $F_{II,1}$ jelentése: X – tetszőleges monadikus művelet (pl. kerekítás, vizsgálatok: $= \emptyset$, $> \emptyset$, $\geqslant$, $\emptyset$ stb.); input paraméter: A; output paraméter: B.

$F_{2,6}$ jelentése: * – tetszőleges diadikus művelet (pl. +, x, –, >, =, stb.); a két input paraméter: A, B; output paraméter: a gép kifejezésvermének teteje.

Egy adatnév mező (pl. A, B) itt szintén deszkriptorból és adatleírás pointerből áll, azonban a deszkriptor itt csak az adat globális, ill, lokális voltát határozza meg.

A fenti két megközelítés értékelése az alábbiak szerint összegezhető:

Először: A forrásnyelv szintaxisát maximálisan közelítő utasításkészlet előnyei: minimális utasításszám programvégrehajtáskor; egyszerűbb fordítás; a tömörebb reprezentáció több „mező-

* Jelölések: – opcionális mező, ——→ opció feltétele

szétválasztás"-jellegű tevékenységet igényel, ezáltal az átlapolt utasításfeldolgozási lehetőségek hatékonyabban kihasználhatók; a COBOL COMPUTE utasításban megadott aritmetikai kifejezések reprezentálhatók lengyel jelöléssel, amely a Kanonikus Értelmezési Formán túlmenően is biztosíthat hatékonyságjavulást; a bonyolultabb aritmetikai utasítások (vagy akár utasítás sorozatok) fordítási időben átalakíthatók lengyel jelölésben felírt kifejezést paraméterként tartalmazó egyetlen COMPUTE utasításra.

Hátrányok: bonyolultabb formátum-értelmezés; nehéz megvalósíthatóság fix adathosszúságokat kezelő mikroarchitektúrákon.

Másodszor: Az általános formátum-orientált utasításkészlet előnyei: egyszerű értelmezés; a frekvenciált operanduskódok könnyen „bevihetők" a műveleti kódba (gyakoriság alapú tömörítés: ld. 4.(7) pont).

Hátrányok: a többparaméteres forrásműveletek több gépi utasításra fordulnak: a fordítás során nehezebb gondoskodni a feltételek és a megfelelő procedurális utasítások összekapcsolásáról.

Az elmondottakból érzékelhető, hogy a Kanonikus Értelmezési Forma által meghatározott tárigény, ill. tárhozzáférések száma ideális esetnek felelnek meg: az operandusmezők deszkriptora az első megközelítésnél, a formátummező a másodiknál olyan „többlet eszközráfordítást" jelentenek, amelyek nem küszöbölhetők ki; hasonlóképpen nehezen tartható be az egyértelmű megfeleltetés szabálya (egy operanduskód-mező az azonos adatnevek számára) akkor, ha egy sokparaméteres forrásművelet több paraméterét ugyanaz az adatnév jelöli (ehhez külön iterációs faktort jelölő mezőkre is szükség lenne) stb. A fentiekben vázolt két utasításkészlet a Kanonikus Értelmezési Forma gyakorlati megközelítéseinek foghatók fel sok tekintetben ellentétes előnyökkel és hátrányokkal. Elképzelhető optimális közbenső megoldásként a kettő valamely kombinációja is. A megfelelő megoldás kialakításánál azt kell figyelembe venni, hogy az „ideális" formával kapcsolatos feltételezések mennyiben nem helytállóak az adott esetben.

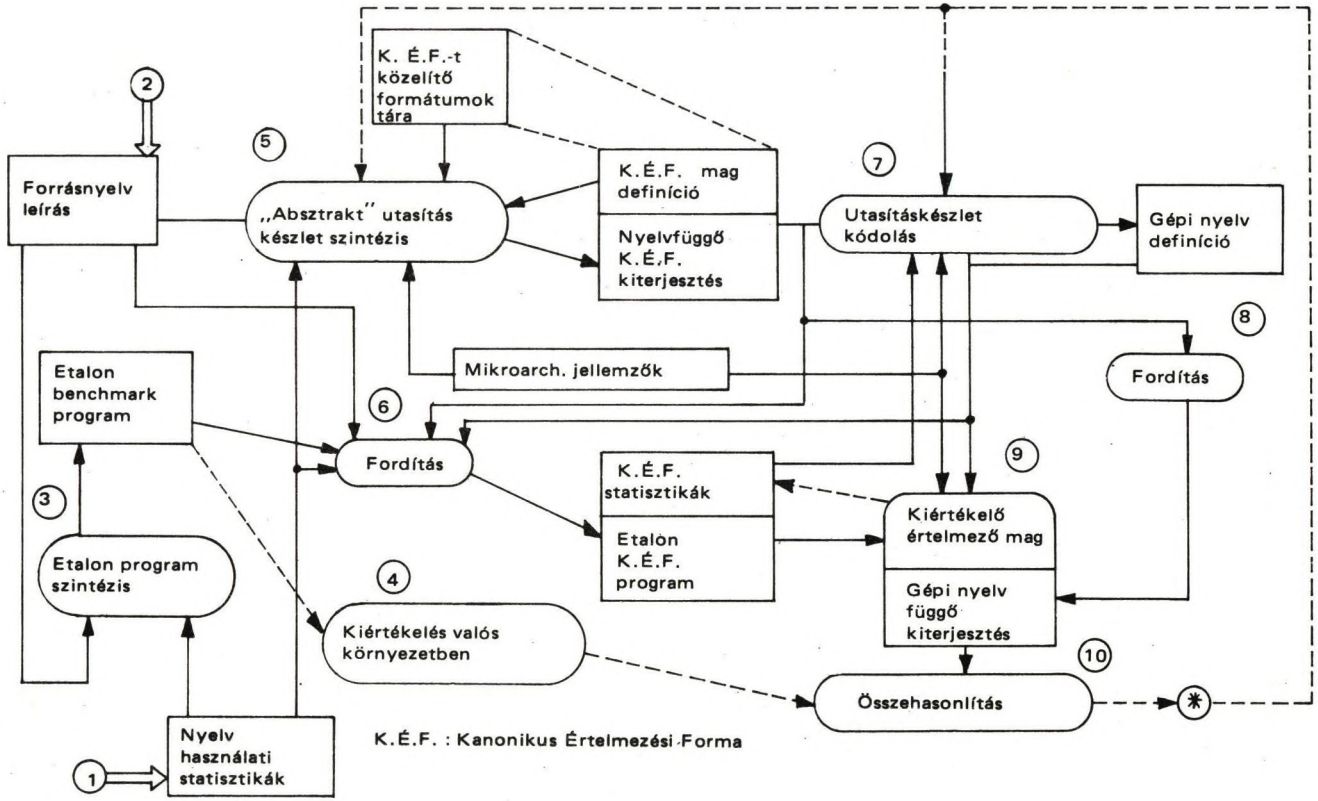
4. A nyelvorientált utasításkészletek tervezésének általános folyamata

Az eddigiekből is látható volt, hogy a Kanonikus Értelmezési Forma szabályok az utasításkészlet tervezésének csak egy részét képezi. Az erre alapozott tervezés teljes folyamatát [6] a 2. ábrán illusztráljuk, az egyes fázisok főbb jellemzőit a 2. táblázatban összesítettük az alábbi megjegyzésekkel:

ad(1.) A legalapvetőbb és általában vizsgált adatok (pl. az egyes műveletek, adattípusok használatának gyakorisága) mellett itt érdekes a Kanonikus Értelmezési Forma alkalmazhatóságát befolyásoló több speciális jellemző is (COBOL esetében ilyenek voltak: az egy paragrafuson belül hivatkozott adatnevek száma; a globális és lokális nevek aránya; a paragrafusok hossza; a paragrafusok száma; stb.).

ad(2.) Itt a forrásnyelvnek olyan részletességű formális leírásáról van szó, ahogy azt az (5.) és (6.) fázist támogató programeszközök indokolják. Az általunk feltételezett utasításkészlet szintézis céljaira általában elegendő a közelítendő forrásnyelv absztrakt szintaxisának definíciója akkor, ha a nyelv objektumainak és műveleteinek bizonyos – a Kanonikus Értelmezési Forma szempontjából mérvadó – tulajdonságait rögzített kulcsszavak azonosítják a leírásban [6].

ad(4.) Az utasításkészlet-függő statikus és dinamikus jellemzőkre példa lehet a Flynn által is használt programméret, utasításlehívásszám, tárhozzáférési szám [1] és a [7]-ben leírt projektben használt jellemzők: az operatív tároló és a feldolgozó processzor között, valamint a regiszter tárolók között mozgatott információk mennyisége.



2. ábra. A nyelvorientált utasításkészlet tervezés általános folyamatának blokk-vázlata

2. Táblázat A nyelvorientált utasításkészlet tervezés fázisai

Tervezési fázis		Eredmény	Lehetséges gépi segédlet	Hivatkozás (példa)
1.	A forrásnyelv használatát jellemző statisztikák gyűjtése	Statikus és dinamikus statisztikák	Szövegkezelők, nyomkövetők	[5]
2.	Formális leírás készítése a forrásnyelvről	Gépi eszközzel feldolgozható nyelvleírás	Nyelv definíciós célra kénelmes nyelvek (pl.: CDL, PROLOG)	[6] (illusztráció)
3.	Etalon benchmark program szintézise	Átlagos terhelést jelentő program architekrúra jellemzők méréséhez	Szintetizáló program (rezolúciós mechanizmus előnyös)	[6] (kísérlet PROLOG nyelven) -teljesítményhangolások teljes konfigurációkra
4.	Kiértékelés valós környezetben	Az összehasonlítási alapul szolgáló meglevő architektúra jellemzői	Kódanalizátorok, nyomkövetők	[1], [2]
5.	Utasítás készlet szintézis	A Kanonikus Értelmezési Források megfelelő „absztrakt" utasításkészlet	Szintetizáló program (rezolúciós mechanizmus előnyös)	[6] (illusztráció)
6.	Etalon program fordítás	Etalon program a Kanonikus Értelmezési Formának megfelelő utasítás készleten	Fordítóprogram	
7.	Utasításkészlet kódolás	Tömör reprecentációt biztosító utasításkészlet	Szintetizáló program (rezolúciós mechanizmus előnyös)	[2] (2—3-szoros méretcsökkenést értek el csak ezáltal)
8.	Az architektúra leírás lefordítása, általánosan értelmezhető kódra	A vizsgált utasításkészlethez szimulátos	Fordítóprogram	[7]: ISP architektúra leíró nyelv használata
9.	A vizsgált utasításkészlet verifikálása, és kiértékelése	Statikus és dinamikus architektúra jellemzők (ld. 4.)	Kiértékelő szimulátor (automatikusan generálható)	[7]: ISP értelmező — kísérlet PROLOG nyelven (folyamatban)
10.	A vizsgált utasításkészlet jellemzőinek összehasonlítása a meglevőével v. az előző változatáéval	Döntés: az új utasításkészlet alkalmazhatóságára és a tervezés folytatására vonatkozóan		

ad (5.) A (2.) szerint funkcionálisan teljes és az (1.) szerint oprimális – a Kanonikus Értelmezési Formát legjobban közelítő – utasításkészlet „absztrakt" leírásának szintézise. Ebben a fázisban kerülnek kiválasztásra a megfelelő utasításformátum-alternatívák (különböző megoldások pl. „n"-paraméteres műveletek, indexelt értékek, egy utasításon belüli több művelet, bizonyos paraméterek iteratív használatának stb. kifejezésére: ld. 3. fejezet), ill, azokból megfelelő kombinációk képződhetnek. Egyes mikroarchitektúra tulajdonságok (a rendelkezésre álló lokális tároló mennyisége és szervezése, átlapolt, ill. pipe-line feldolgozás lehetősége, adatszélességek flexibilitása stb.) befolyásolja azt, hogy milyen irányban térünk el az „ideális" formától.

ad (6.) Az (1.) fázisban meghatározott statikus statisztikák ennek megfelelően átkonvertálásra kerülnek.

ad (7.) Ebben a fázisban a Kanonikus Értelmezési Forma alkalmazásával nyert hatékonyság javuláson túlmenően további javítás érhető el. Az (5.) fázisnál „absztrakció" alatt azt értettük, hogy csak az egyes utasításmezőknek a száma és a rendeltetése, valamint az azok kombinálási lehetőségeire vontatkozó korlátozások adottak; ezen túlmenően az egyes mezők sorrendje, hosszúsága és kódolása tetszőleges lehet. Ebben a fázisban az utasításkészlet reprezentációjának összes részlete meghatározásra kerül. A programok kódjának tömörsége ezáltal biztosítható, hogy gyakorisági alapon változó hosszúságú mezőket használunk a formátumok, a műveletek, az operandusok azonosítására; a gyakori operanduskódokat „bevisszük" a műveleti kódba; a gyakori műveleti kódokat a formátumkódba; a gyakori paraméterértékek kiválasztásával csökkentjük az operandusok számát (pl. kétparaméteres műveletek helyett egyparaméteresek használata 0, 1, 2, space stb. értékek esetén).

ad (10) Az összehasonlítás eredményétől függően módosulhatnak az (5.) és a (7.) fázisban alkalmazott stratégiák, és azok megismérlésével sor kerülhet a tervezés következő menetére.

5. Következtetések

A nyelvorientált utasításkészletek meghatározásához felállíthatók olyan általános érvényű szabályok, amelyek alapján a tervezés jelentős részben rutinszerűvé válik és gépi eszközökkel támogatható. Az itt vázolt tervezési módszerrel fél nagyságrend körüli hatékonyságjavulás érhető el a ma elterjedt utasításkészlethez képest. A nyelvorientált processzorok előterjesztését az alábbi körülmények még inkább valószínűvé teszik:

– a tároló és processzor elemek olcsóbbá válásával az eszközkihasználás hatékonyságának kérdése továbbra sem veszít aktualitásából, mivel az ár/teljesítmény viszony továbbra is motivált piaci szerepet fog játszani;

– a jelenleg elterjedt magasabbszintű programozási nyelvek feldolgozása kompatibilitási okok miatt még sokáig feladat lesz, a korszerű programozási ill. feladatmegoldási módszereket képviselő nyelveket illetően pedig egyrészt az itt vizsgált tervezési módszer van annyira általános, hogy azokhoz is alkalmazható legyen, másrészt ezeknek az új módszereknek az alkalmazásához éppen a megfelelően illeszkedő architektúrák nem állnak rendelkezésre;

– a Neumann-géptől merőben eltérő architektúrák kutatásának gazdaságossági aktualitás szempontjából előfeltétele az, hogy a hagyományos programozáshoz illeszkedő – közvetlenebbül megtérülő – optimális eszközök kidolgozása meg legyen oldva.

[1] Hoevel L. W., Flynn M. J.: „The Structure pf DEL: A New Theory of Interpretive System Design", Stanford University, March 1977.
[2] Tannenbaum A. S.: „Implications of Structured Programming for Machine Architecture", CACM, March 1978. p. 237–246.
[3] Lecht Ch.: „The Waves of Change. Chapter VIII.", Computer World, July 4. 1977.
[4] Nyelvorientált mikroprogramozott számítógépek utasításkészlet tervezésének egy megközelítése. SZKI tanulmány, 1978. december.
[5] Chevance R. J., Heidet T.: „Static Profile and Dynamic Behavior of COBOL Programs", SIGPLAN Notices, March 1978. p. 44–57.
[6] Egy architektúra tervezési környezet előzetes specifikációja. SZKI tanulmány, 1978. december.
[7] Barbacci, Burr, Fuller, Sieworek: „Evaluation of Alternative Computer Architectures", Carnegie–Mellon University, 1977.

OPERÁCIÓS RENDSZEREK HORDOZHATÓSÁGÁRÓL

Somogyi József
SZÁMKI

Az operációs rendszerek hordozhatóságának egyik legnagyobb akadálya a mai számítógépek input-output rendszerének rendkívüli változatossága. A karakterenkénti átviteltől a bonyolult csatornaprogramok végrehajtására képes input-output processzorokig a legkülönfélébb berendezések találhatók meg egymás mellett. Az operációs rendszer hierarchikus felépítése azonban lehetővé teszi az egymástól jelentősen eltérő gépek közötti hordozhatóságot is.

1. Bevezetés

Az operációs rendszerek hordozhatóságának kiterjedt irodalma ellenére ma még kevés olyan operációs rendszer van, melyeket nem-triviálisan különböző gépre ténylegesen átvittek. [1, 2, 3, 4]. Az egyetlen széles körben elterjedt operációs rendszer, amelyik ilyen hordozáson átesett, a PDP–11 UNIX operációs rendszere [5].

Az intézetünkben készült ANSWER operációs rendszer tervezése és megvalósítása során így több olyan probléma került felszínre, melyek híján voltak minden előzménynek.

Így az input-output rendszerek sokfélesége, mint hordozási probléma, az eddigi irodalomban nem szerepel súlyának megfelelően. Ennek oka az lehet, hogy a ténylegesen elvégzett munkák során az átvitel olyan gépek között történt, melyek input-output rendszere nem különbözött egymástól olyan mértékben, hogy a probléma teljes életnagyságában megmutatkozzon.

Történt ugyan kísérlet az UNIX rendszernek IBM gépre való átvitelére, ahol a PDP és IBM gépek input-output rendszere már elég jelentősen eltér egymástól ahhoz, hogy általánosabb következtetéseket lehessen levonni belőle. Ez a munka azonban megmaradt kísérleti stádiumban, és az eredmények sincsenek publikálva.

A jelen dolgozat speciálisan az input-output rendszerek problémáival foglalkozik.

A kiindulási feladat az ANSWER operációs rendszer elkészítése volt, mely

– a programfejlesztést támogatja
– interaktív
– hordozható.

A három feltétel egyenlő súllyal szerepel, mint meghatározó tényező, mi azonban csak a hordozhatósági feltétel következményeivel foglalkozunk.

A hordozhatóság minimális feltételeként az egész operációs rendszer egyetlen magasszintű nyelven készült. A nyelv kiválasztásának problémája messze meghaladja a dolgozat kereteit, így csak megemlítjük, hogy a választott nyelv a CDL2 (C.H.A. Koster, 1975), mely

– nyílt nyelv, ezáltal lehetővé teszi a hardwarehez való közvetlen hozzáférést,
– struktúrált nyelv, aminek, mint látni fogjuk, az operációs rendszerek hordozhatósága szempontjából döntő szerepe van,
– jó minőségű fordítóprogram áll mögötte, mely feleslegessé teszi a hatékonyság miatti aggodalmat.

Operációs rendszerek esetében azonban a hordozhatóság nem pusztán implementálási kérdés, hanem legalább ugyanannyira tervezési kérdés is. Attól, hogy az operációs rendszer magasszintű nyelven van megírva, még nem lesz hordozható, a hordozhatóságot az operációs rendszerbe bele kell tervezni.

A továbbiakban az ANSWER rendszer tervezése és elkészítése során szerzett tapasztalatokból levonható általános következtetésekkel foglalkozunk.

2. Az input-output rendszer

2.1 Általános megjegyzések

A ma létező számológépek input-output rendszere első pillantásra rendkívül változatosnak látszik. A tüzetesebb vizsgálat azonban azt mutatja, hogy ha eltekintünk néhány extrém esettől, pl. a még manapság is gyártott megszakításrendszer nélküli gépektől, akkor a fennmaradó nagy többség három kategóriába sorolható.

2.1.1 Input-output processzorokkal rendelkező gépek

Az input-output processzorok (csatornák) leglényegesebb tulajdonsága, hogy programozhatók. Utasításkészletük általában elég szerény, és input-output műveletekre van kihegyezve: read, write, control, stb. A csatornautasításokból álló csatornaprogramot az input-output processzorok önállóan hajtják végre. Az input-output műveletekben a központi egység feladata csak annyi, hogy a csatornákat felszólítsa a csatornaprogram végrehajtására, majd a művelet végén felfogja a csatorna jelzését a művelet befejezéséről.

2.1.2 Blokkosított átvitelt végző gépek

Ezek leglényegesebb tulajdonsága, hogy mindegyik perifériához van néhány regiszter (memória cím, byte számláló, status, stb. regiszter), melyek írásával lehet a perifériákat irányítani. A perifériák a központi egység egyszeri felszólítására egy műveletet végeznek el, ami pl. egy csatornautasításnak felel meg.

2.1.3 Karakteres átvitelt végző gépek

A perifériák a központi egység egyszeri felszólítására csak egy karakter átvitelét végzik el. Gyors perifériák esetén (mágneslemez, mágnesszalag) azonban ez az átviteli mód a gép operációs rendszerét megoldhatatlan feladatok elé állítaná. Ezért a valóságos konfigurációkban a karakteres átvitelű perifériák a blokkosított átvitelű perifériákkal együtt jelennek meg.

2.2 Hordozhatósági problémák

A csatornával rendelkező és nem rendelkező gépek közötti átmenetkor a fő probléma az, hogy a csatornaprogramból központi egységnek szóló programot (vagy fordítva) kell csinálni. Ez azonban már nem közönséges hordozási probléma, aminek hardware és software okai egyaránt vannak.

2.2.1 Hardware okok

A csatornákkal rendelkező gépeken futó programok több, egymással párhuzamosan végrehajtható programrészre esnek szét. Egy értelmes program legalább három ilyen részből áll: az input programból, a feldolgozó programból, és az output programból. Az input és output programot a csatornák, a feldolgozó programot a központi egység hajtja végre. Ezek a programrészek azáltal állnak össze egyetlen programmá, hogy a részek egymásnak információkat adnak át. (Pl. a központi egység által végrehajtott program mondja meg az input-output terület címét és hosszát.)

Az ilyen, időben párhuzamosan végrehajtott programokkal — mai terminológiával processzekkel — kapcsolatban két feladatot kell megoldani: a programok szinkronizálását, és az egymás közötti kommunikációt.

A jelenlegi hardwarek a szinkronizációt kielégítően oldják meg. A csatornákat a központi egység input-output utasításai segítségével lehet aktivizálni, a csatorna pedig megszakítással jelzi a program végét.

A kommunikációval azonban már súlyos problémák vannak, elsősorban a csatornautasítások címzési módjának hiányossága miatt. A csatornautasítás két operandusa közül a memóriacím csak abszolut cím lehet, a byte számláló pedig csak direkt operandus. Ezért a csatornaprogram bemenő paramétereit vagy statikusan kell a csatornaprogramba bekódolni, vagy a csatornaprogramot dinamikusan felül kell írni.

Ilyen körülmények között illuzórikus azt elvárni, hogy a csatornaprogramokat egy multiprocesszoros gép központi egységére át lehessen vinni.

2.2.2 Software okok

Míg a hardware okok a csatornaprogramok hordozását teszik lehetetlenné, addig a software okok ezen csatornaprogramok írását.

Ennek főleg történeti okai vannak. A magasszintű nyelven programozó előtt a programok központi egység és csatornaprogramokra való szétválasztása rejtve marad. De még a csatornákkal nem rendelkező gépeken is az input-output műveletek végrehajtása az operációs rendszer meghívásával, vagyis a programon kívül történik. Emiatt a magasszintű nyelvek, nyelveket nem operációs rendszerek írására találtak ki, az input-output műveletekkel eléggé mostohán bánnak.

Így jelenleg a magasszintű nyelveken még formális lehetőség sincs arra, hogy egy csatornaprogramot le lehessen írni. Ez azonban még csak a kisebbik probléma. A programban pedig is azt is kifejezésre kell juttatni, hogy a (valahogyan megírt) csatornaprogram a „főprogrammal" párhuzamosan futó programrész. Vagyis, a csatornaprogramok írásához konkurrens nyelvre van szükség.

2.3 Az ANSWER rendszer input-output hierarchiája

Az általános megjegyzésekben felsorolt géptípusoknak megfelelően definiáljuk a következő input-output szinteket.

2.3.1 Program szint

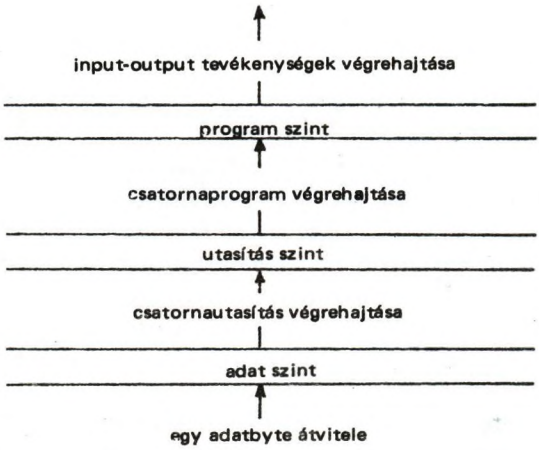
Ezen a szinten az operációs rendszer feltételezi, hogy vannak csatornaprocesszorok, amelyek képesek több utasításból álló csatornaprogramok végrehajtására. Feltételezi továbbá, hogy a központi egységnek van olyan utasítása, mellyel a csatornát el tudja indítani, és hogy a csatornaprogram végét megszakítás jelzi.

2.3.2 Utasítás szint

Ezen a szinten az operációs rendszer feltételezi, hogy a perifériák egy utasítás (read, write, stb.) végrehajtására képesek. Feltételezi továbbá, hogy vannak cím, számláló, stb. regiszterek, melyek írásával a perifériák vezérelhetők.

2.3.3 Adat szint

Ezen a szinten az operációs rendszer feltételezi, hogy a perifériák képesek egy adat átvitelére. Az egyes szintek által nyújtott és elvárt szolgáltatásokat a következő ábrán foglaljuk össze:



Az így definiált szintek rendelkeznek azzal a tulajdonsággal, hogy mindegyik alá oda lehet tenni egy létező hardwaret. A különböző szintű gépek közötti átmenet azonban nem közönséges értelemben vett hordozás. Magasabb szintű gépre való áttérés ugyanis az alacsonyabb szinteknek az operációs rendszerből való elhagyásával, alacsonyabb szintű gépre való áttérés pedig az alacsonyabb szinteknek az operációs rendszerhez való hozzáadásával történik.

Az azonos szinten lévő gépek viszont már csak formailag térnek el egymástól, funkcionálisan nem. Így az azonos szinten levő gépek között az átmenet már közönséges értelemben vett hordozással megvalósítható.

Szükséges megemlíteni, hogy bár a csatornával nem rendelkező gépeken az operációs rendszer interpretálja a csatornaprogramokat, ennek az interpretációnak a hatékonysághoz nincs semmi köze. Az input-output műveletek végrehajtása minden operációs rendszerben úgy történik, hogy a programok valamilyen formában (pl. kontrol blokk) leírják input-output igényüket, és az operációs rendszer ezt a formát interpretálja. A csatornaprogramok interpretálásakor pontosan ez történik, csatornautasítás alakú kontrol blokkokkal.

2.4 Input-output programozás az ANSWER rendszerben

A csatornákkal rendelkező gépeken futó programok párhuzamos természetét az operációs rendszerek általában elrejtik a programozó elől. Az operációs rendszerből nézve azonban a párhuzamosság nyilvánvaló, ezért valamiféle processzkezelő mechanizmusra mindenképpen szükség van.

Az ANSWER rendszerben futó processzek dinamikusan újabb processzeket hozhatnak létre. Ezekben az új processzekben a létrehozó processz által megnevezett programok kerülnek végrehajtásra. Ez körülbelül a legegyszerűbb mechanizmus, ami tetszőleges (nem konkurrens) magasszintű nyelven is könnyen megvalósítható, és csatornaprogramokra is kiterjeszthető.

Mind minden operációs rendszerben, így az ANSWER rendszerben is, a gyakran előforduló feladatokra kész utility programok állnak rendelkezésre. Az ANSWER rendszerben azonban a központi egység által végrehajtható „közönséges" programok mellett előregyártott csatornaprogramok is vannak, és a rendszer processzkezelő mechanizmusa ezekre a programokra is kiterjed. Egy input-output művelet elindítása így egy csatornán futó processz létrehozását jelenti.

Ezzel egyrészt elkerülhető, hogy a magasszintű nyelven írt programokba a csatornaprogramokat bele kelljen erőltetni, másrészt nyilvánvalóvá válik a programok párhuzamos szerkezete.

3. Következtetések

Bár elvileg nem látszik lehetetlennek, hogy egy operációs rendszert változtatás nélkül lehessen vinni egyik gépről a másikra, a jelenlegi hardware-software körülmények között ez még a legismertebb géptípusok között is megoldhatatlan. Ez így nem túlságosan meglepő, hiszen abszolut hordozhatóságot még operációs rendszernél magasabb szintű és szabványosított nyelven írt programok esetében sem lehet elérni. Míg azonban ez utóbbinak csak technikai okai vannak, az operációs rendszerek hordozhatóságának a mai hardwarek elvi határt szabnak.

Viszont lényeges megállapításokat tudunk tenni arra nézve, hogy egy reálisabb hordozhatósági követelmény milyen feltételeket ró ki magára az operációs rendszerre. Az input-output rendszerek vizsgálata két olyan feltételre vezet, melyek az operációs rendszer szerkezetét alapvetően befolyásolják.

Az egyik ilyen feltétel a szigorúan hierarchikus szerkezet. Ez a más programoknál is erénynek számító tulajdonság hordozható operációs rendszerek esetében létszükséglet.

A másik fontos feltétel a konkurrens programozás. Ez a követelmény más, ebben a dolgozatban nem tárgyalt okokból is fennáll, alkalmazását azonban ki kell terjeszteni a csatornák programozására is.

Irodalomjegyzék

1. J.E. Stoy and C. Strachey: OS6 — An experimental operating system for a small computer. Part 1: General principles and structure. Comp. J., 15 (May, 1972)
2. J.E. Stoy and C. Strachey: OS6 — An experimental operating system for a small computer. Part 2: Input-output and filing system. Comp. J., 15 (August, 1972)
3. D. Morris, G.R. Frank and C.J. Theaker: Machine independent operating systems. In Information Processing 77, North-Holland (1977)
4. D. Thalmann and B. Levrat: SPIP, a way of writing portable operating systems. Proc. ACM Computing Symposium (1977)
5. S.C. Johnson and D.M. Ritchie: Portability of C programs and the UNIX system. Bell System Technical J., 57 (July-August, 1978).

AUTOMATIZÁLT ADATHORDOZÓ ELŐKÉSZÍTŐ RENDSZER ÉS KÉT NAGYGÉP KAPCSOLATÁNAK SZIMULÁLÁSA SOFTWARE ESZKÖZÖKKEL

Szigetvári Miklós
KSH, SZIG

...ezetés

Előadásunkban ismertetjük a KSH Számítástechnikai Igazgatóságán 1978–79-ben kifej-...ett automatizált adathordozó előkészítő rendszert és két IBM nagyszámítógép hardware ...csolatának szimulálását. Tisztában vagyunk azzal, hogy a KSH SZIG helyzete több szem-...ből sajátos:

- két nagy IBM számítógép van
- virtuális operációs rendszert használunk
- speciálisak a feldolgozási igények,

...t előadásunkban kiemeljük azokat a tanulságokat, tapasztalatokat, amelyek véleményünk ...int általánosan hasznosíthatók.

...rdware és a software konfiguráció vázlata

A jelenlegi konfiguráció alapja egy IBM 370/155 számítógép 1Mbyte memóriakapacitással, ...ális tárkezelésre alkalmas hardwarrel és kb 600Mbyte mágneslemezkapacitással. A géphez ...programozói terminál (Remote Job Entry) és 18 display terminál van kapcsolva.

Ez az alapkonfiguráció 1978 végén egy 370/138-as számítógéppel bővült, amely 512 Kbyte ...acitású és 4 db 100 Mbyte-mágneslemezzel rendelkezik.

A fennálló közismert korlátozások következtében a két gépet nem lehet hardware-el össze-...csolni, üvegfalakkal elkülönített géptermekben vannak elhelyezve.

Mindkét gépen SVS (Single Virtual Storage) operációs rendszer használunk. Az SVS rend-...az MTV operációs rendszer virtuális tárkezeléssel kiegészített változata, számunkra a legnagyobb ...ye az, hogy elfogadható nagysága mellett (kb. 160–180 Kbyte valós memória) támogatja ...nteraktív TSO (Time Sharing Option) lehetőséget. Az ismertetendő feladatok szempontjá-...lényeges, hogy az SVS-t a HASP (Houstone Automatic Spooling and Priority System) ...szerrel együtt használjuk.

A KSH SZIG-en nagytömegű statisztikai adatfeldolgozás folyik, a 370/155 gépen naponta ...350–400 job fut. A felhasználók körülbelül 10 000 darab mágnesszalagból álló készletet ...elnak. A mágnesszalagkészlet nyilvántartása és kezelése automatizált, az operációs rendszer ...mánykatalógusán keresztül történik. A felhasználók csak állománynevekre hivatkoznak ...megfelelő szalagszámok kikeresését az operációs rendszer végzi egy nagyméretű (10 000–...000 bejegyzés) katalógusból.

Az ismertetendő rendszert két lépésben terveztük meg. Az első lépés célja az volt, hogy el-...tsünk egy olyan adathordozó előkészítő rendszert. amely a jobok lefuttatásához szükséges ...mennyi információt képes kezelni. A második lépésben azt szerettük volna elérni, hogy a két ...tien számítógép a felhasználók és a gépkezelők számára egységes rendszert alkosson.

Az automatikus adathordozó előkészítés (SETUP)

Közismert gyakorlat, hogy a felhasználók job-jaikhoz különböző formájú, kézzel kitöltött munkalapokat mellékelnek. Ezeknek a lapoknak a kezelése, nyilvántartása, áttekintése rendkívül nehézkes. A mágnesszalagok, mágneslemezek előkészítése pontatlan, nagyon gyakori, hogy a job futása közben kell a mágnesszalagért a raktárba szaladni (nálunk műszakonként 400–500 mágnesszalagot kell előkészíteni).

További hátránya a munkalapoknak az, hogy a programozói terminálról és a TSO–ból küldhető job-ok körét korlátozni kell olyanokra, amelyek nem igényelnek adathordozó előkészítést.

Egy olyan rendszer előkészítését határoztuk el, ahol:

- minden információt, amely a job lefutásához szükséges a felhasználók adnak meg a job-ban elhelyezett vezérkártyákon;
- felhasználók által megadott információkat a gépkezelő lekérdezheti;
- nincs szükség munkalapra, TSO-ból programozói terminálról tetszőleges job-ok küldhetők.

A SETUP rendszer működése

Az SVS/HASP rendszerben a job-ok beolvasását, ütemezését az elkészült outputok kinyomtatását a HASP végzi. A job lefutásának minden fázisa a HASP-on keresztül irányítható, a HASP tartalmaz információkat az egyes várakozási sorok állapotáról. Mivel a felhasználó által megadott információknak a job „egész élete" folyamán rendelkezésre kell állniok, ezért a SETUP rendszert úgy alakítottuk ki, hogy együttműködve a HASP-al végigkísérje a feldolgozás egész menetét.

A jobok beolvadása:

A beolvasást a HASP végzi lokális kártyaolvasóról vagy programozói terminálról, vagy az ún. nevezett internal reader-ről. (Az internal reader-rel lehet jobokat átküldeni TSO terminálokról batch futásra, vagy job-ból job-ot elindítani). Ha job jelenik meg valamelyik olvasón, akkor a HASP átadja a SETUP számára a szükséges információkat (jobnév, jobsorszám, jobosztály, prioritás, a becsült CPU idő) és a SETUP program felkészül a job fogadására.

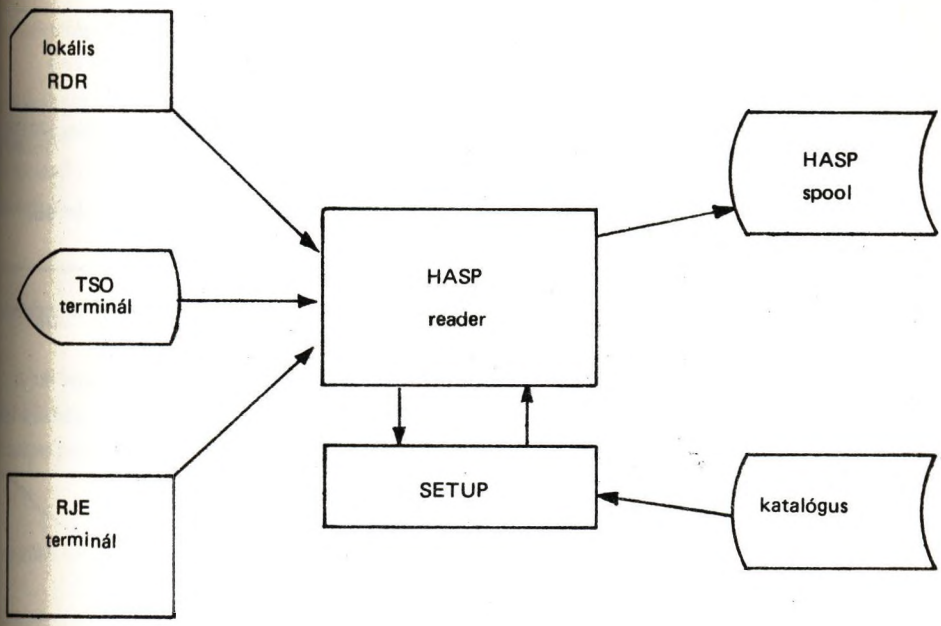
Ha a beolvasás alatt álló job-ban a SETUP rendszer vezérkártyája jelenik meg, akkor ezt a kártyát a SETUP elemzi.

Ha a kártya hibás, akkor hibajelzéseit a kártya 73–80 oszlopára helyezi, ha a kártya hibátlan azaz az állománynevekhez megtalálhatók a katalógusbejegyzések, nincs formális hiba, akkor az információkat saját táblázatába írja.

A job-ok indítása

Amikor egy job beolvasása befejeződött, a SETUP program dönt a job további sorsáról:

- ha SETUP hiba fordul elő, akkor átkerül a job egy olyan osztályba, ahonnan gyors jobcontrol ellenőrzése után törlődik;



- ha a job-ban nem volt adathordozó előkészítési igény, akkor beáll az input várakozó sorba;
- ha adathordozó előkészítési igény volt a job-ban, akkor felfüggesztett (hold) állapotba kerül. Ezt az állapotot a gépkezelőnek kell feloldania, ha az előkészítés megtörtént.

A job-ok befejeződése

A HASP számára akkor fejeződik be egy job, ha valamennyi outputja elkészül. Ekkor a HASP és a SETUP töröl valamennyi, a jobra vonatkozó információt.

A felhasználók

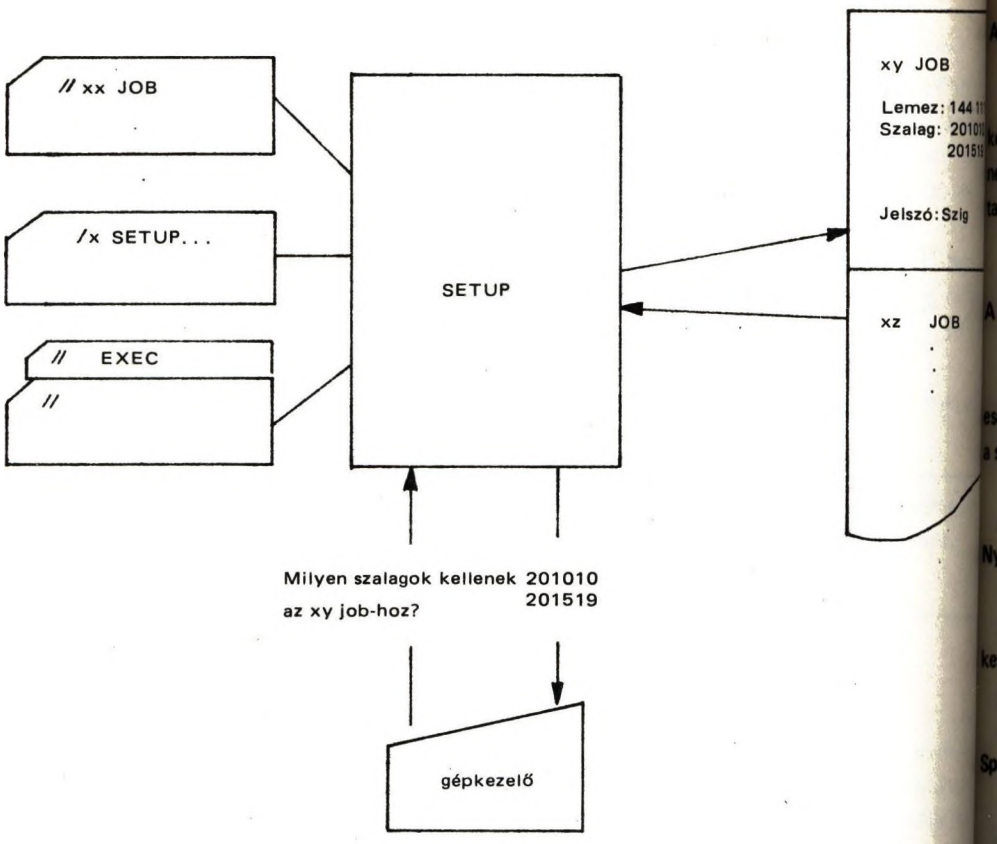
A felhasználók igényeiket vezérkártyákon adják meg. Ezek a kártyák a SETUP karaktersorozattal kezdődnek, és az OS job vezérlő nyelv (JCL) szabályaihoz hasonlóan kulcsszavas paramétereket és paraméterlistákat tartalmaznak.

A felhasználók által megadható információkat a következő csoportokra lehet bontani:

Ütemezési információk

Multiprogramozású operációs rendszerben rendkívül fontos az, hogy pontosan ismert legyen a lefutásra váró jobok erőforrásigénye. A felhasznált erőforrások:

- Gépidő



A job várható futási ideje (start stop idő) előre nehezen becsülhető, ugyanakkor lényeges eleme az ütemezésnek. A rendszerben megadható a becsült idő, de ha a felhasználó ezt nem adja meg, akkor a SETUP rendszer a tervezett CPU idő tízszeresét tekinti becsült futási időnek. (A start/stop és a CPU idő közötti 10:1 arány több év adatai alapján jó átlag a jelenlegi feladatainkra.)

– Memórianagyság.

A virtuális operációs rendszerekben a memórianagyság jelentősége csökkent. Mi a jobosztályokat használjuk a memóriaigény jelzésére.

– Egyidőben használt mágnesszalagegység, illetve mágneslemezek száma.

A felhasználóknak a SETUP vezérkártyákon meg kell adnia, hogy hány szalag, illetve lemezegységet köt le egyidőben a job-ja.

Az ütemezés szempontjából lényeges még a jobosztály és a prioritás. Ezeket az információkat az OS jobkártya megfelelő mezőiben adják meg. A jelenleg használt jobosztályok a memóriaigény és a sürgősség alapján kategorizálják a job-okat. A jobosztályokon belüli sorrendet (prioritást) a HASP algoritmusa állapítja meg. A felhasználó által megadott értékek túllépése esetén a jobok futása megszakad.

...ételőkészítési információk

A felhasználónak fel kell sorolnia azokat a mágneslemezeket, mágnesszalagokat, amelye... a job-jában használni kíván. A hivatkozás történhet sorszámra, vagy állománynévre. Állomány...re történő hivatkozás esetén a SETUP rendszer a névhez tartozó sorszámokat kikeresi a ka...lógusból és az állománynév helyébe helyettesíti.

...lefutáshoz szükséges információk

A job lefutása során a jelszóval (password) védett állományokba való írás, illetve olvasás ...etén szükséges a gépkezelői beavatkozás. A felhasználók a SETUP vezérkártyákon megadják ...szükséges jelszavakat és a job-ok kérésére ezeket a jelszavakat a gépkezelőnek kell begépelnie.

...yomtatási információk

A nyomtatási információkban adhatják meg a felhasználók a különleges nyomtatási igénye...t (speciális papír, több példány stb.).

...peciális kérések

A felhasználók előszeretettel írnak elő speciális kéréseket job-jaikhoz. Ezek körét igye...keztünk korlátozni, de azért megadtuk a lehetőségét annak, hogy speciális kéréseket írjon elő ...elhasználó.

Noha a felhasználók számára egy kevés többletmunkát jelent a SETUP kártyák elkészítése, ...kárpótolja őket, az, hogy megszűntek a gépteremmel a félreértések, valamint az, hogy tetszőleges jobot beolvashatnak a programozói terminálon (a TSO jelenleg csak kísérleti üzemben működik).

...A gépkezelők

Az SVS/HASP rendszerben a gépkezelővel való kommunikációkat a HASP végzi. Igyekeztünk úgy kialakítani a SETUP kommunikációs részét, hogy hasonló legyen a HASP-hoz.

A SETUP operátori parancs három részből áll:

...obbkiválasztás

A parancs első részében ... definiálja a gépkezelő azokat a job-okat, amelyekről információt ...kar kérni. A job kiválasztó részben feltételek írhatók elő:

jobnévre N
jobsorszámra J
a becsült futási időre R
az egyidőben használt lemezek és szalagok számára D, T
a jobb osztályra C
a prioritásra P
a várakozó sorra Q

Például:

$UJ = 201–245, Q = H, D = Ø, R< 6 Ø

parancs azokat a jobokat választja ki, amelyek sorszáma 201 és 245 között van, felfüggesztett állapotban vannak, nem használnak cserélhető mágnes lemezt, a becsült futási idő kevesebb mint 60 perc.

Mezőkiválasztás

Itt definiálja az operátor, hogy a kiválasztott jobok mely információira kíváncsi:
általános információk
a használt mágneslemezek
a használt mágnesszalagok
a jelszavak
az output kinyomtatásával kapcsolatos információk
speciális információk.

Küldési hely, egyéb igények

Jelenleg a 370/155 számítógéphez négy konzol tartozik. Egy írógép, 2 display és egy mátrixprinter, amely a mágnesszalagtárakban van elhelyezve. A küldési hely meghatározásával irányíthatja az operátor például a szalagraktárba a listát, ahol a raktárosok azonnal elkezdhetik a mágnesszalagok előkészítését.

Lehetőség van arra is, hogy a kiválasztott jobok által igényelt összes mágnesszalag, illetve mágneslemez rendezett listáját elkészítse a gépkezelő egy konzolra vagy sornyomtatóra.

Ebben a részben kérhető a SETUP-nak az a speciális szolgáltatása, hogy mágnesszalagra írja az input várakozó sor egy kiválasztott részét.

A gépkezelők rövid idő alatt elsajátították a SETUP kezelését. Az adathordozó előkészítés gyorsabb, pontosabb.

A software megvalósítása

Az elkészítésben rendkívül nagy segítséget jelentett a HASP szerkezete. Könnyen módosítható, teljes egészében rendelkezésre áll a forrásprogram is. A SETUP program a HASP mell

fut kitüntetett (supervisor) állapotban. A két program az OS központi táblázatán (CVT) keresztül tart kapcsolatot egymással. A HASP az egyes SETUP funkciókat közvetlen vezéléssátadással hívja meg.

A SETUP program a táblázatait a virtuális memóriában tartja (kb. 100Kbyte). Ez a megoldás lényegesen gyorsít a működésen, ugyanakkor a táblázat legnagyobb része állandóan inaktív „kilapozódik". Eddigi tapasztalataink alapján a SETUP rendszer nem jelent észrevehető terhelést az operációs rendszer számára.

További fejlesztési elképzelésünk, hogy szeretnénk megvalósítani a SETUP segítségével a feltételes job vezérlést, ahol a felhasználók a jobok j között definiálhatnak kapcsolatokat, az egyes jobok lefutása a megelőző jobok eredményétől függhet.

A két számítógép kapcsolata

Annak ellenére, hogy két azonos típusú számítógéppel rendelkezünk, és ezeken a gépeken azonos operációs rendszer fut, a hardware kapcsolat hiányában komoly problémát jelentett egy egységes rendszer kialakítása. A közös állománykatalógus, forrásprogram és loadprogram-könyvtárak használata az egyes jobokat az egyes gépekhez köti, ezért az egyes feldolgozások a gépekhez vannak rendelve, vagy a 155 vagy a 138-as gépen futnak állandóan. A 155 gépen elsősorban interaktív munkák, programtesztelés, adatbázisokkal kapcsolatos feladatok futnak, míg a 138-as gépen a rendszeres feldolgozások. A munkák szétválasztása jobosztályok alapján történik, a 138-as gépen numerikus a 155-ös gépen alfabetikus karakterek a jobosztályok.

A jobok beolvasása, adatelőkészítés

Valamennyi job egységesen a 155 gépbe kerül beolvasásra. Az adatelőkészítés a SETUP rendszer segítségével történik meg, külön-külön a 155-ös jobosztályokra és a 138-as jobosztályokra, Az adatelőkészítés megszervezésénél a 138-as gép állománykatalógusának elérése jelentett problémát. Ezt úgy oldottuk meg, hogy a gépkezelők a 138 gép állománykatalógusát 2–3 óránként mágnesszalagra dumpolják és ezt a mágnesszalagot a 155 gép egy előre kijelölt mágneslemezére töltik. A SETUP program a jobosztály alapján dönti el, hogy melyik állománykatalógusban kell keresnie. A katalógusok kezelésére szolgáló IBM software rendkívül lassú (egy 10–12 000 bejegyzést tartalmazó katalógus szalagraírása 3–4 óra/, ezért a lemezdumpoló programot használjuk arra, hogy a rögzített cylindereken elhelyezkedő katalógust kiírjuk, illetve visszatöltsük.

Jobok átvitele a 138-as gépre

A 155 gép várakozó soraiban levő 138-as jobokat egy SETUP parancs segítségével a gépkezelő mágnesszalagra írhatja. A szalagraírás két lépésben történik:

1. A SETUP a HASP internalreader-re ír egy job-ot, amely adatkártyaként tartalmazza az átküldendő jobok valamennyi kártyáját.
2. Az elkészített job lefutásának az eredménye az a mágnesszalag, amelyről a job-ok a másik gépen betölthatők.

A mágnesszalag az előkészített adathordozókkal együtt a 138-as gépre kerül. Mivel a 138-as gépen nem fut a SETUP rendszer, ezért a job-ok kiírásával egyidőben elkészül egy olyan SETUP lista is, amely a job-ok futtatási forgatókönyvének tekinthető. A szalagra kiírt job-ok törlődnek a 155 gép várakozó sorából.

Programok átvitele a 138-as gépre

Mivel a felhasználók programteszteléseiket a 155 gépen végzik, gyakori, hogy az elkészült programokat át kell vinni a 138-as gépre futtatásra. Kidolgoztunk ezért egy rendszer, amelynek a felhasználó a SETUP-hoz hasonló vezérkártyákon definiálja, hogy a 155-gép mely könyvtáraiból milyen programokat akar átvinni a 138-as gép könyvtáraiba. A felhasználók ezeket a vezérkártyákat a 138-as gépre küldött job-jaikban helyezik el.

Az a program, amely az átküldött job-okat szalagra írja, megkeresi ezeket a vezérkártyákat és összegyűjti a 155-gép közös könyvtáraira leadott igényeket. Az elkészült igénylistát a küldési hely (138-as könyvtárak) alapján rendezi és az átküldött job-okat tartalmazó mágnesszalagra írja az átküldeni kívánt programokat is.

Az elkészült mágnesszalag betöltése is két lépésben történik:

- az átküldött programok betöltése a megfelelő könyvtárakba,
- az átküldött job-ok betöltése a 138 gép várakozó sorába.

A rendszer kezelése

A rendszer működésének alapvető feltétele az, hogy olyan vezető gépkezelő irányítsa a feldolgozást, aki átlátja mindkét gép működését, és helyes sorrendben kezdeményezi az egyes feladatokat:

- katalógusdump a 138 gépen
- katalógusdump visszatöltése a 155 gépen
- adathordozó előkészítés a 138 gépre
- job-ok és programok szalagraírása
- a készült szalag betöltése a 138 gépen.

Ennek az irányítási feladatnak a támogatására a két gépterem közötti helységben egy display konzolt helyeztünk el a vezető gépkezelő számára. Ez a konzol a 155 géphez van kapcsolva, erről végzi a vezető gépkezelő a kétgépes rendszer irányítását.

A jelenlegi kétgépes rendszer használata a felhasználók számára egyszerű, a gépterem részéről azonban körültekintést igényel. Továbbfejlesztésének alapfeltétele azonban valamilyen hardware-kapcsolat (osztott mágneslemez, csatorna adapter) megteremtése.

A MEZŐGAZDASÁGI VÁLLALATOK AUTOMATIZÁLT TERVEZÉSI RENDSZERE ÉS ALKALMAZÁSÁNAK TAPASZTALATAI

Dr. Tóth József
AGRÁRTUDOMÁNYI EGYETEM, DEBRECEN

A mezőgazdasági vállalat, még ha el is tekintenénk sokirányú külső (partneri, területi és népgazdasági) kapcsolataitól, bonyolult rendszer, amelynek alrendszerei és elemei térben és időben sokirányú kölcsönhatásban vannak egymással.

Jellemzőjük, hogy általában sokirányú termelési (gyakran feldolgozási és szolgáltatási) tevékenységet folytatnak, s a vállalaton belül a különböző termelési ágak egymással sokirányú kapcsolatban és kölcsönhatásban vannak, mind a termékek termelése és felhasználása, mind a termelési erőforrások hasznosítása, mind pedig a biológiai tényezők vonatkozásában.

Adott termék egyik ágazatban késztermékként jelentkezik, a másik ágazatban (sőt esetenként ugyanabban az ágazatban pl. vetőmag) alapanyag, vagy termelési eszköz (pl. feldolgozó üzemek, takarmánytermesztés és állattenyésztés), s a különböző ágazatok adott termelési erőforrást egyidejűleg, vagy eltérő időben hasznosítanak (pl. a traktorokat és talajművelő gépeket különböző ágazatok egyidejűleg, vagy eltérő időben hasznosítják, a kombájnt a kalászosok a betakarítási időszakban, a napraforgó és a kukorica különböző adapterekkel felszerelve az őszi időszakban igénylik.) A biológiai kapcsolatok tágabb, vagy szűkebb határok között jelentős befolyással vannak a termelési arányokra, (pl. állati termék termelés és takarmány-termelés, évelő növények és növénytársítások arányai, vetésváltás stb.)

Ugyanakkor a mezőgazdasági termelés földhöz kötött és az időjárásnak kitett, s a termőföld fizikai, kémiai és biológiai tulajdonságai és domborzata, az időjárási tényezők igen nagy befolyást gyakorolnak a termelésre, a termelési technológiákra, az erőforrás szükségletre és az erőforrások hasznosítására.

Jellemző a mezőgazdaságra, a termelés viszonylag alacsonyabb szintű koncentrációja, sok esetben 2–5 ezer há-s vállalat léte. E vállalatok gyakran igen eltérő természeti adottságok (termőföld, lombozat, időjárási viszonyok különbözősége és eltérő közgazdasági viszonyok mellett eltérő piaci lehetőségek, piachoz való távolság, pénzügyi és vagyoni helyzet stb.) folytatják tevékenységüket és tulajdonképpen a vállalati feltételek és lehetőségek oly sokfélék, ahogy vállalat, s azonos két vállalat nem is található.

A fentiekben röviden vázolt, valamint a terjedelmesség elkerülése végett nem említett mezőgazdasági vállalatok tervezésével kapcsolatban a következőket jegyezzük meg:

1. A mezőgazdasági vállalat hatékony gazdálkodásához nem nélkülözheti a magas színvonalú, megalapozott vállalati (fejlesztési és éves) tervet. Minél bonyolultabb egy rendszer, annál inkább függ működése és hatékonysága a tervezés színvonalától.
2. A mezőgazdasági vállalatok egyedi, a természeti , közgazdasági és üzemi feltételeket sokoldalúan figyelembe-vevő, részletesen kimunkált tervet igényelnek, s még az alapparaméterek nagyrészét is egyedileg a vállalat konkrét feltételeit figyelembevéve kell kimunkálni, nem törekedhetünk tehát univerzális megoldásokra.
3. A mezőgazdasági vállalati tervezés nem lehet egyszeri aktus, hanem permanens folyamat kell, hogy legyen, állandóan követve a természeti, a közgazdasági és vállalati feltételek változását, beleértve a tudományos-technikai fejlődés adta lehetőségeket is.

Ilyen körülmények között nem kis feladat a tervezés automatizált rendszerének kimunkálására vállalkozni, ugyanakkor a mezőgazdasági vállalati tervezés bonyolultságából adódóan igen munkaigényes. Valószínűleg a korszerű matematikai tervezési módszerek is elsősorban munka-

igényességük miatt nem terjedhettek el széles körben a gyakorlatban, hiszen a módszerek alkalmazása részletesebb és szélesebb körű előkészítő és befejező munkát tett szükségessé, s ennek gépesítése nem volt megoldva.

Többéves munkával egy olyan tervezési rendszert sikerült kidolgozni a mezőgazdasági vállalatok számára, amelyben a nagytömegű és időigényes manuális munkák — beleértve a gépírási feladatokat is — lényegében teljesen gépesítve vannak, s a szakember kizárólag az érdemi adatokat végzi, ugyanakkor a vállalat egyedi tervezése valósítható meg, figyelembe véve annak konkrét feltételeit.

Az automatizált tervezési rendszer kidolgozása mindenekelőtt a vállalatnak, mint komplex rendszernek az elemzését, a vállalati alrendszerek felderítését, a vállalat külső kapcsolatának áttekintését és rendszerezését, elemzését, s vállalati modellek kidolgozását kívánta meg. E kapcsolatok elemzése vezetett el az alapvető döntési feladatok rendszerének és kölcsönhatásának vizsgálatához, az információszükséglet feltárásához, rendszerezéséhez, majd a tervezési folyamat fázisainak áttekintésével a komplex tervezési rendszer kialakításához.

A rendszer kidolgozásánál alapvető elvként tartottuk szem előtt, hogy az egyrészt általános valamennyi mezőgazdasági vállalatnál alkalmazható legyen, másrészt, hogy tegye lehetővé a vállalat konkrét viszonyait messzemenően kifejezve egyedi, az adott vállalatnál gyakorlatban jól megvalósítható és hatékony gazdálkodást biztosító terv kimunkálását.

Kiinduló munkafázisa a konkrét feltételek tanulmányozása. Ezt ma még nem tudjuk automatizálni, de ha a vállalati könyvelés és nyilvántartás számítógépes rendszere megvalósul, ez a munkafolyamat is nagyobbrészt automatizálható lesz.

A konkrét feltételek ismeretére alapozva koncepciókat alakítunk ki a vállalat lehetőségeiről, a termelhető termékekről, az elérhető tényleges hozamokról, technológiai és termelési erőforrás lehetőségeire, illetve azok változtatásának lehetőségeire. Ez a folyamat sem ma, sem jövőben nem automatizálható, mindig a tervezők, a vállalati vezetés és a szakemberek feladata marad, de nincs is terhelve nagymennyiségű manuális munkával.

A következő feladat a termelési technológiai alternatívák, vagy a tervezéshez szükséges paraméterek kidolgozása. Ez az igen munka- és időigényes, nagymennyiségű manuális munkát tartalmazó munkafázis rendszerünkben automatizált. E részprogramrendszerünk működési vázlatát az az 1/a. és 1/b. sz. ábrákon mutatjuk be.

Az ábrákból kitűnik, hogy a technológiai tervezés törzsadattárára alapoz. A törzsadattárak tartalmazzák a tervezéshez szükséges mindazon paramétereket, amelyek az ország valamely vállalatára, (vagy legalábbis egy nagyobb táj, vagy terület vállalatára) adottak. Ezen adattárak és tartalmuk a következő:

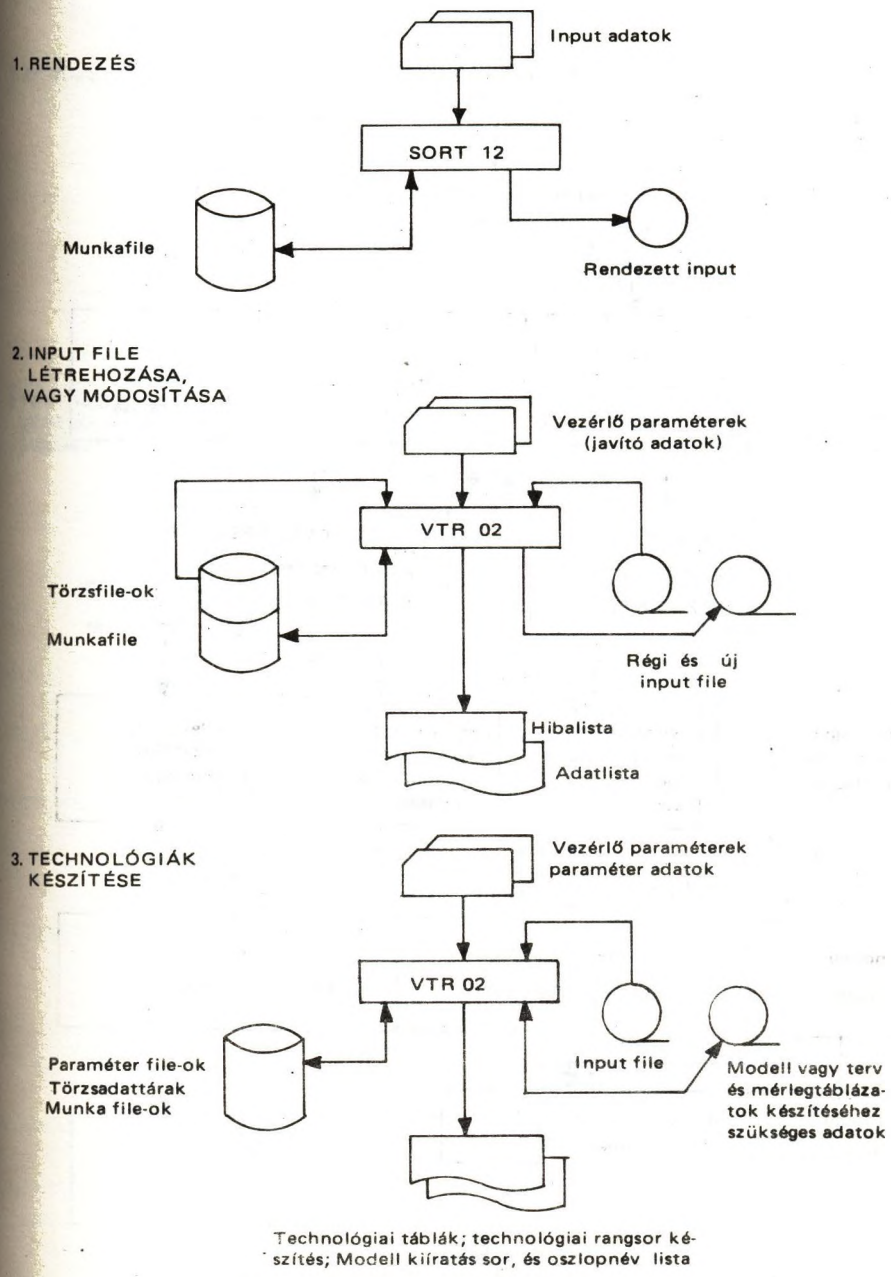
a) *Gépek és eszközök adattára*
   Kiterjed a mezőgazdaságban használatos valamennyi gépre és eszközre tartalmazva azok kódszámát, megnevezését, árát és költségadatait (amortizáció, javítási és egyéb költségek, illetve költségkulcsok).

b) *Anyagok adattára*
   Felöleli a mezőgazdaságban használatos anyagféleségeket, tartalmazva ezek kódszámát, megnevezéseket, egységárat, hatóanyagtartalmát és mennyiségi egységét, természetes csomagolási típusok szerint, mivel az árak ettől függően is eltérőek.

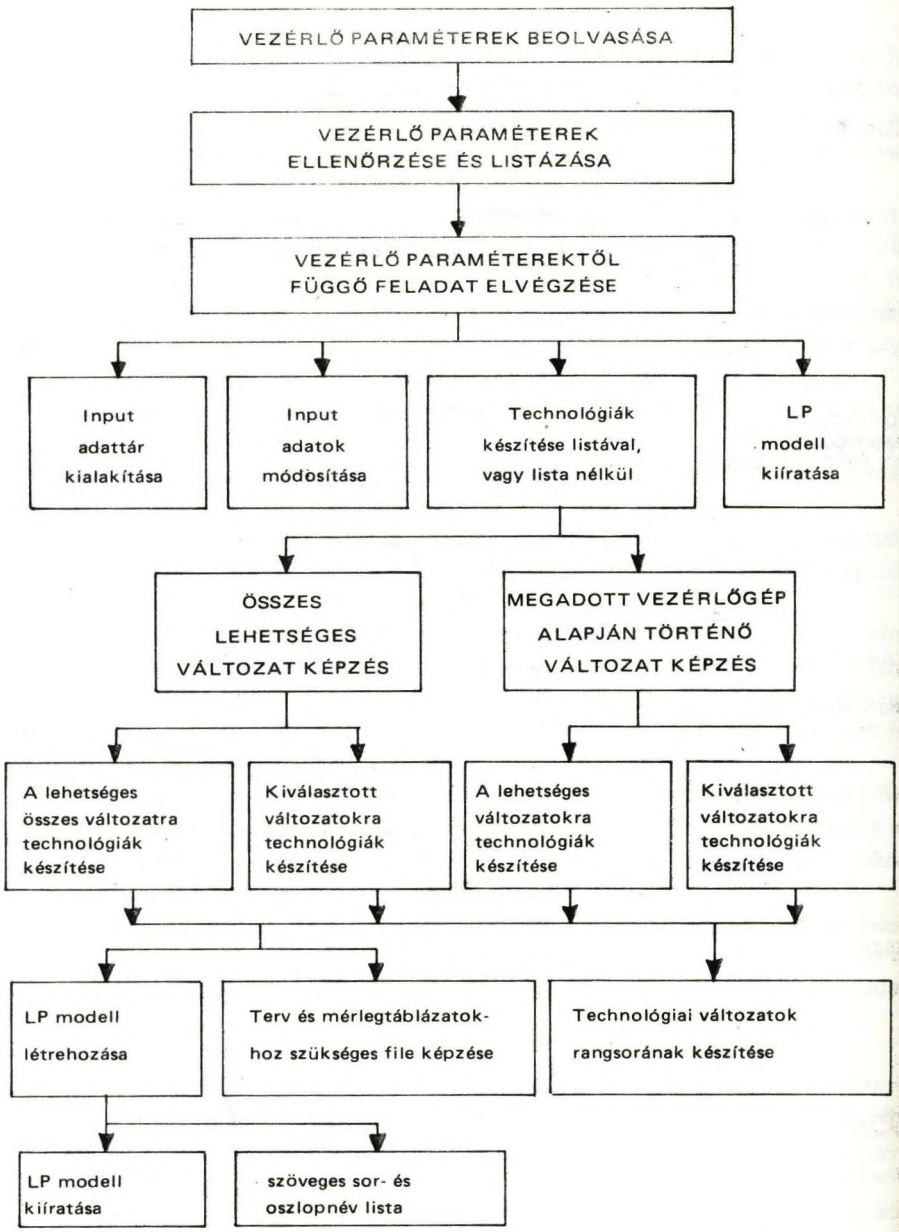
c) *Műveleti adattár*
   A mezőgazdaságban előforduló műveletek kódszámát és megnevezését tartalmazza.

d) *Ágazatmegnevezési adattár*
   A mezőgazdaságban előforduló termelési ágazatok kódszámát és megnevezését foglalja magában.

A törzsadattárkezelő program működési vázlatát a 2. ábra mutatja.



1. a ábra
TECHNOLÓGIÁK KÉSZÍTÉSÉNEK FOLYAMATA

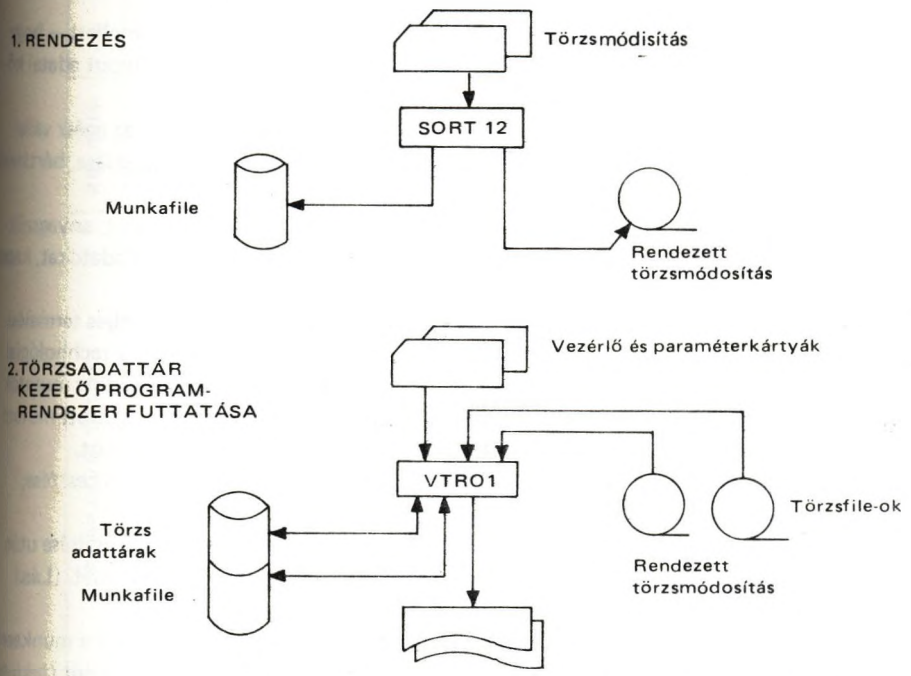


1. b. ábra

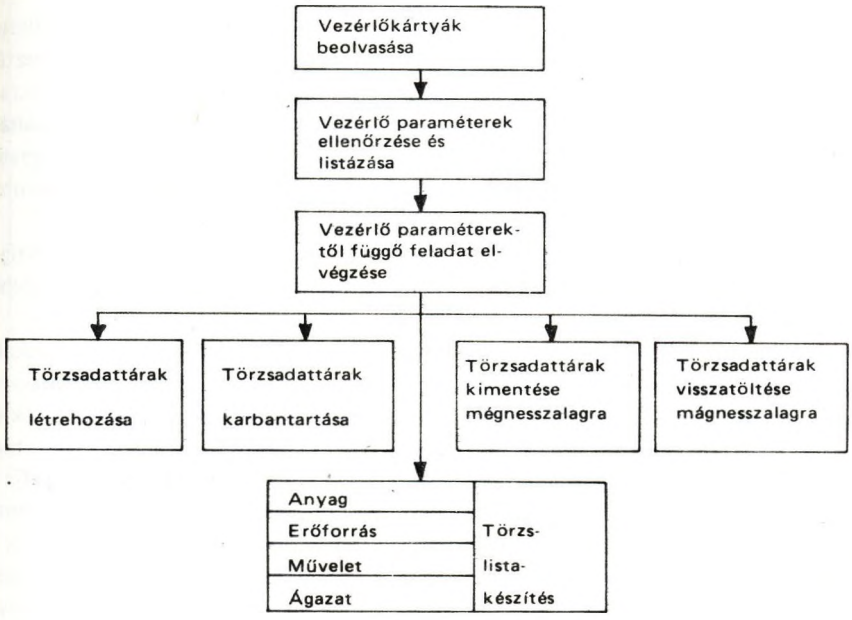
A TECHNOLÓGIÁKAT KÉSZÍTŐ PROGRAMRENDSZER (VTRO2) MŰKÖDÉSI VÁZLATA VEZÉRLŐ PARAMÉTEREKTŐL FÜGGŐEN

TÖRZSADATTÁRKEZELŐ RENDSZER MŰKÖDÉSE

Folyamatábra



A TÖRZSADATTÁRKEZELŐ PROGRAMRENDSZER (VTR01) MUKÖDÉSE



2. ábra

Megjegyezhető, hogy a törzsadattárak folyamatosan a tervezési munkák során állandóan tovább építhetők (a program hibajelzést ír ki, ha a törzsadattárban nem szereplő eszközre, anyagra stb. hivatkozunk) és állandóan karbantarthatók (ár vagy hatóanyag változás stb.) másrészt, hogy a törzsadattárak önmagukban is hasznosak, mert jól rendszerezett nyilvántartást jelentenek és különböző elemzésekre adnak lehetőséget.

A technológia szerkesztő program másik jellemzője, hogy törzsadattárak mellett – hogy a technológiákat az adott vállalat konkrét feltételeire készítsük el — a rendszer input adatai közzött kell adni a vállalatra, illetve az ágazatra vonatkozó információkat.

A vállalati információk azokat a jellemzőket foglalják magukba, amelyek az egész vállalatra vonatkoznak, pl. a vállalat neve, székhelye, terület-nagysága, s ennek megoszlása, bértételek és köztehermunkák stb.

Az ágazatra vonatkozó információk adják meg az adott ágazat megnevezését, anyagszükségletét, fő és melléktermékekre vonatkozó naturális és értékadatokat, műveleti adatokat, kapcsolatokat és teljesítmény adatokat stb.

Ezen információk alapján a számítógép összeállítja a vállalat ágazataira a teljes termelési folyamatot naturális és értékparamétereivel együtt. Egy-egy ágazatra igen sokféle technológiai változat készíthető el. Ezek közül kívánság szerint a számítógép elkészíti és megadott mutatók alapján sorba rendezi az összes lehetséges változatot, kiválasztja és elkészíti a megadott mutató szerint legkedvezőbb változatot, vagy elkészít egy általunk meghatározott változatot.

A technológiák kidolgozása után következhet a matematikai modell megszerkesztése, mely folyamat tervezési rendszerünkben szintén automatizált.

A modell felépítésének vázlatát a 3. ábrán mutatjuk be. A technológiák elkészítése után a vezérlő paraméter hatására a VTRO2-es program végzi el a modell megszerkesztését. (Lásd 1.a. és 1.b. sz. ábra.)

A programcsomag automatikusan képezi a területmérleget (vagy mérlegeket) a munkaerő, a gép és eszközmérlegeket, munkaerő csoportok, illetve géptípusok szerint, havonként (tekintve a munka idényszerűségét), s automatikusan képezi az erőforrásváltozókat, azok paramétereit, valamint a célfüggvényt, illetve a célfüggvényeket.

Jelenleg még a modellszerkesztés nem tökéletes, tekintve, hogy egyrészt az állattenyésztési változókra nem rendelkezünk technológia szerkesztő programmal, így az állattenyésztés modellbe kerülő adatait kell megadnunk az input adatok között, másrészt a termelési korlátokra vonatkozó feltételek modellbe építését általában az LP modell elkészített adatbázisának módosításával történhet a LIPROS programcsomag adatkezelő részének használatával. A modell adatbázisának kialakítását, módosítását és megoldását a 4. ábra mutatja. (Ezt a lineáris programozási programcsomagot a SZÁMKI Operációkutatási Főosztályán készítették.)

A modell megoldása és a variáns számítások elvégzése (ami, mint ismeretes eddig is automatizált volt) után kerül sor a döntésre, ami sem ma, sem a jövőben nem automatizálható, a vállalatvezetés feladata.

A tervezés utolsó fázisa a döntés után a tervmérlegek és tervtáblázatok kimunkálása. Ezt a feladatot a számítógép a döntésről informálva — automatikusan oldja meg. (A program működési vázlatát az 5. ábra mutatja.) Kidolgozza a termelési szerkezetet, átlaghozamok és összetermék táblázatot, takarmánymérlegeket, munkaerő, gép és eszközmérlegeket ágazatonként, havonkénti, munkaerő csoport, illetve gép- és eszköztípusonként megbontva, megadja az anyagfelhasználási tervet, kimunkálja a termelési érték, árbevétel, költség és jövedelemtervet, a költségeket költségcsoportonként részletezve, az általános költségeket kulcsszámok alapján, a gépek és eszközök éves fix költségeit pedig csak igénybevétele alapján terhelve az ágazatokra.

Csupán jelezném, gogy a feltételek megváltozása esetén, rendszerünkben viszonylag gyors és nagyobbrészt automatizáltan oldható meg a terv felülvizsgálata, szükség esetén átdolgozása.

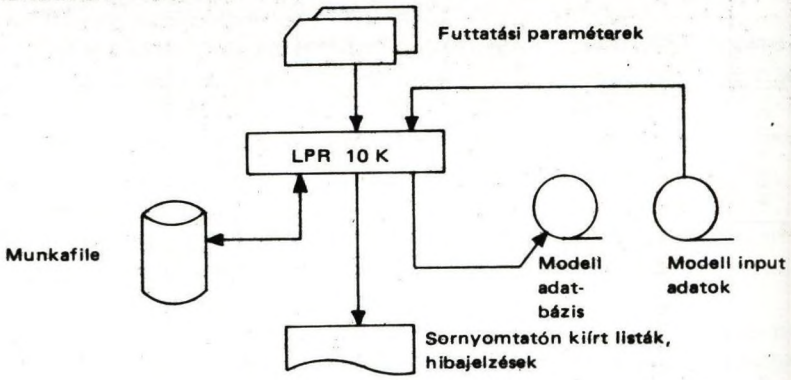
TEVÉKENYSÉG VÁLTOZÓK						FORRÁSVÁLTOZÓK			RELÁCIÓ	b
NÖVÉNYTERMESZTÉS VÁLTOZÓK			ÁLLATTENYÉSZTÉS VÁLTOZÓK			GÉP VÁLTOZÓK		MUNKAERŐ VÁLTOZÓK		
Őszi-búza	kuko-rica	--- ---	Szarvas-marha	ser-tés	--- ---	¦ ¦ ¦ ¦ ¦ ¦	--- --- ---	¦ ¦ ¦ ¦ ¦ ¦		
Fajlagos munkaerő és gépszükséglet havonta (I.—XII. hónap)						Munkaerő és gép fajlagos kapacitása havonta (I.—XII. hónap)			$\leq$	$\underline{0}$
Terület, takarmány, anyagmérlegek és egyéb feltételek									$\leq$	$\underline{b}$
CÉLFÜGGVÉNY									———	EXTRÉM

3. ábra

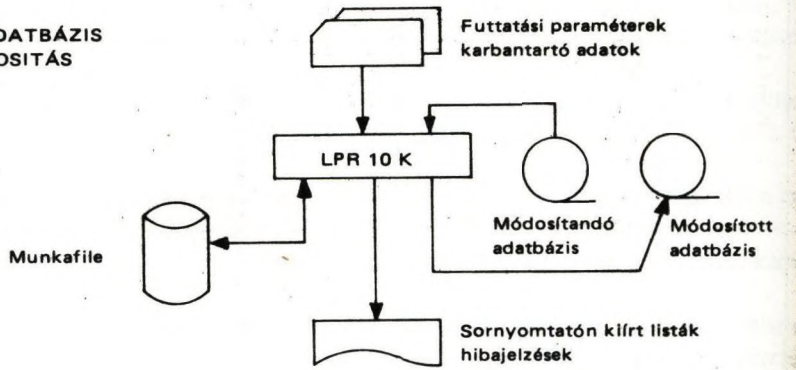
A MODELL FELÉPÍTÉSE

A LIPROS ADATBÁZISKEZELŐ RENDSZER HASZNÁLATA

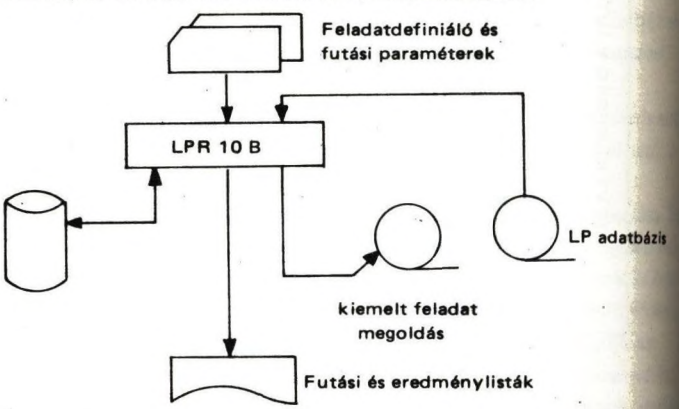
1. LP ADATBÁZIS LÉTREHOZÁSA



2.LP ADATBÁZIS MÓDOSITÁS

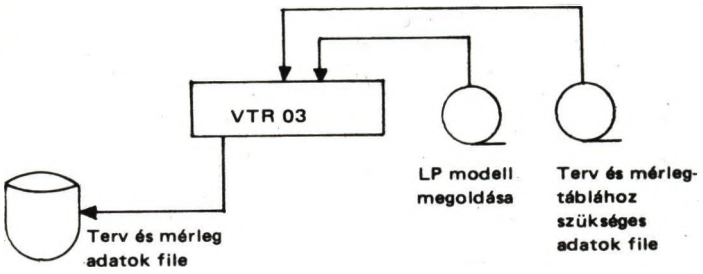


A LIPROS OPTIMALIZÁLÓ RENDSZER HASZNÁLATA

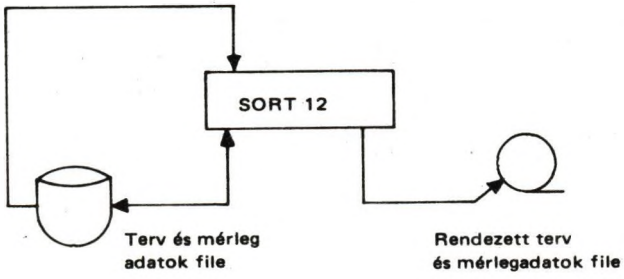


4. ábra

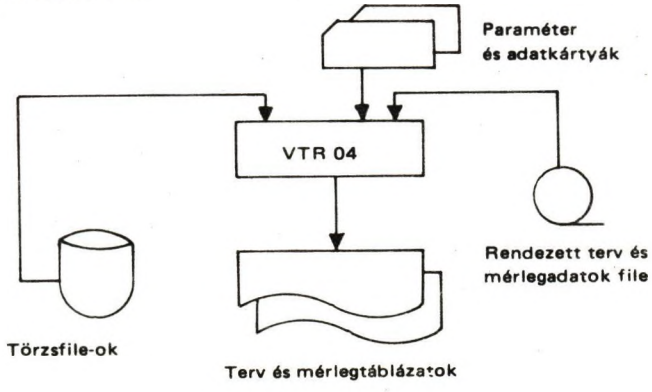
1. FELSZORZÁS
(Megoldás x
alapmodell)



2. RENDEZÉS



3. TERV ÉS MÉRLEG-
TÁBLÁZATOK KÉSZITÉSE



5. ábra

A TERV ÉS MÉRLEGTÁBLÁZATOK KÉSZÍTÉSÉNEK FOLYAMATA

Az R—10-es számítógépünkre kidolgozott tervezési rendszerünkkel megvalósítható feladatokat összefoglalóan az 1. táblázat mutatja.

Tervezési rendszerünket már több termelőszövetkezetben sikerrel alkalmaztuk a gyakorlatban, s terv végrehajtása során jelentősen növekedett a szövetkezet jövedelme. Hangsúlyoznunk kell, hogy e jelentős jövedelememelkedés döntően és elsősorban a szövetkezeti tagság jó munkájának és a jó szövetkezeti vezetésnek az eredménye, de segítette azt a jó tervezés is. Erre vonatkozóan csupán egy — eddigi munkánk során igaz legnagyobb eredményt felmutató termelőszövetkezet példáját ragadjuk ki.

A termelőszövetkezet nem egészen 2000 ha területen, igen jó természeti feltételek között gazdálkodik, tagsága nagyon szorgalmas és munkaszerető s igen kitűnő vezetéssel és szakemberekkel rendelkezik, s tervezésünk előtt is magas színvonalon, jövedelmezően gazdálkodott.

1975-ben kezdtük el a szövetkezet fejlesztési tervének kimunkálását, s a terv sarokszámainak kialakulása után a termelőszövetkezet — a részletek kimunkálását nem is várva meg —, elkezdte a terv megvalósítását. A szövetkezet három év alatt a tervet lényegében, — bár bizonyos rugalmassággal, ami a jó vezetés sajátja — megvalósította, illetve túl is teljesítette.

Három év alatt a magasszintről induló termelőszövetkezet, a mezőgazdaság számára nem éppen kedvező közgazdasági körülmények ellenére termelési értékét és a szövetkezeti bruttó jövedelmét megduplázta, pedig kizárólag mezőgazdasági tevékenységet folytat, tehát ipari tevékenységgel nem foglalkozik, s mindhárom évben elnyerte a Kiváló Termelőszövetkezet címet. Ilyen fejlődésre aligha lehet sok példát találni.

Sorszám	A rendszer fázisai	Programév	Feladatai
1.	Törzsadattár kezelés	VTRO1	— Paraméterek ellenőrzése — Törzsadattárak létrehozása mágneslemezre — Törzsadattárak mágnesszalagra másolása — Módosítás — Mágnesszalagról mágneslemezre másolás — Törzslisták készítése — Hibalisták készítése
2.	Növénytermesztési technológiák készítése	VTRO2	— Paraméterek ellenőrzése — Technológiai adatbázis létrehozás — Technológiai adatbázis módosítás — Technológia készítés vezérgépkiválasztással — Technológiák tervezése egy előző futás alkalmával meghatározott ágazatok változatai közül kiválasztott ágazat — változat sorozatokra — Technológia tervek ágazatonkénti változatainak rangsorolása kívánt célfüggvényegyüttható alapján — Terv — mérleg táblázatokhoz szükséges adattár kialakítása — Modell szerkesztés — Modell kiíratása mátrix formában sor- és oszlopnevekkel
3.	Modell esetleges módisítása és megoldása	LIPROS program-csomag	— LP modell adatbázis létrehozása, módosítása a modell megoldása
4.	Terv és mérleg-táblázatok készítése	VTRO3	— Modell megoldással történő felszorzása után a terv és mérlegtáblázatok elkészítése — Szántóföldi növénytermesztés szerkezete — Növénytermesztés termékfelosztási terve — Takarmánymérleg — Műszakszükséglet (szakmunka, segédmunka) — Műszakszükséglet (gépekre) — Műtrágya felhasználási terv — Anyagfelhasználási terv — Termelési érték, költségek és jövedelmek ágazati szinten

ABSZTRAKT GÉPRE ALAPOZOTT RENDSZERPROGRAMOZÁSI KÖRNYEZET LÉTREHOZÁSA AZ ÁSZSZ SZÁMÍTÓGÉPEIN

Dr. Treer Róbert—Wlassics Péter
ÁSZSZ

1. Bevezetés

Az ÁSZSZ számítógépein elvégzendő programozási feladatok nagy horizontális és vertikális kiterjedtsége a lépcsőzetes programfejlesztést gépfüggetlen módon támogató rendszerprogramozási környezet megteremtését teszi szükségessé. Horizontális kiterjedtség alatt a sok különböző típusú (Honeywell 66-os sorozat, Datanet, MTS 7500, INTERSCAN, PDP 11) egymással és más gépekkel kapcsolatban álló számítógépet, vertikális kiterjedtség alatt a legkülönbözőbb szinteken jelentkező, gyakran egymásra épülő, jellegükben egymástól sokszor élesen eltérő programozási feladatot értjük.

Ez az igény indított arra, hogy a gépi kódhoz közeli hatékonyságot megcélozva megkíséreljük egy ilyen környezet felépítését. Előadásunkban a környezet létrehozásának első – tervezési és alapeszköz-megvalósítási – fázisáról számolunk be.

2. A rendszerprogramozási környezet sajátosságai

Rendszerprogramozási környezet alatt egymásra épülő, egymással szoros kölcsönhatásba levő programfejlesztési eszközök olyan csoportját értjük, amely a felhasználói körében felmerülő problémák „kényelmes" leírására és megoldására vagy eleve kész, vagy könnyen létrehozható eszközöket (továbbiakban: nyelveket) tartalmaz. Ezek közül egyesek a környezet horizontális és vertikális fejlesztésére, mások típusfeladatok célorientált megoldására alkalmasak. Fontos, hogy a környezet nyelvein megfogalmazott algoritmusok a környezet számítógépein hatékonyan — ha szükséges, gépfüggetlen módon — legyenek megvalósíthatóak.

Az ÁSZSZ számítógépein fejlesztés alatt álló (részleteiben már elkészült) programozási környezet főbb specifikumait az A—E pontokban foglaljuk össze.

A) A környezet a *lépcsőzetes programfejlesztés* módszerét támogatja, alapelemeiből (AM absztrakt gép, AM/MA rendszerprogramozási nyelv) kiindulva — ismert vagy új rendszerprogramozási nyelvek implementációinak létrehozásával — saját maga is lépcsőzetesen fejleszthető. Ennek módszere lényegében ugyanaz, mint az alkalmazási feladatok megoldására a B, C, D pontokban részletezett eljárás.

B) A lépcsőzetes programfejlesztés történhet *kiterjesztéssel:* kiválasztjuk a környezet azon kiterjeszthető nyelv-elemét, amely a megoldandó feladat/típus/nak leginkább megfelel. Ezután nyelvünk fogalmaira támaszkodva a feladat kategóriáinak jobban megfelelő absztrakt adat-, illetve vezérlési struktúrákat, műveleteket, azaz nyelvi fogalmakat definiálunk. A probléma leírás egy ilyen lépcsőfokának a kiválasztott nyelv egy kiterjesztése felel meg, melyet kezdetben a „saját területértője" a környezet „gazdájának" segítségével definiálhat. A feladat és megoldásának absztrakciós szintjei egymásra épülő nyelvek hierarchiájában valósulnak meg. A hierarchia egy adott szintjén eltekinthetünk az alacsonyabb szintek megvalósításának módjától. A kiterjesztés további lehetősége bármely szinten megszüntethető. A nyelv ilyen „lezárása" fordító-programjának bizonyos mértékű egyszerűsítését eredményezi és mivel a fordító-program az AM gépfüggetlen assemblerben van írva, a kapott speciális célú nyelven írt programok a környezet számítógépein futtathatóak.

C) A környezet rendszerprogramozási nyelvei közül több (kezdetben az AM/MA) alkalmas *...axis és szemantika definiálására*. A szintaxis megadása lényegében meta-nyelven (a BNF ...ltozatában) történhet, de ennek során olyan programtulajdonságokat is definiálhatunk, ...yeket általában csak a szemantika megadásával vesznek figyelembe. A „kibővített szintaxis" ...sával (1. 4. 1. fejezet) magasabb szintű kompatibilitás érthető el különböző implementá... között, több kritérium alapján lesz eldönthető egy program formális helyessége, egységes ...osztikai rendszer létesíthető. Az ily módon megadott *nyelv-definiálás implementáció-orien-*..., nem igényli a formális nyelvek elméletének mélyebb ismeretét, így a némi rendszersoft-...es beütéssel rendelkező szakember is használhatja a szakterületének leginkább megfelelő ...definiálására.

D) Az implementáció-orientált nyelvdefiníció egyben *fordítóprogram definíció*, és a defi... nyelv *transzlátorának automatikus generálását* is biztosítja.

E) Az eddig felsorolt jellemzők nyelvek, nyelv-változatok keletkezésének elősegítésével ...szólag csak tovább növelik a bábeli nyelvzavart. Azért látszólag, mert a létrejövő nyelvek ugyan... a *portábilis alapelemekre* (az AM absztrakt gép, az AM/MA kiterjeszthető nyelv) épül... ...nek és így az „anyanyelvük" közös. Az „új" nyelv fordítóprogramja az AM absztrakt gépet ...valósító többi számítógépen is futtatható.

F) Egy programozási környezet használata alapvetően befolyásolhatja a programozási ...ust, a készített programok hatékonyságát, programozási rendszerek létrehozásának általános ...dszereit. Ezért nagyon fontos az alapvető módszertani elvek helyes kiválasztása. Esetünkben ...k az elvek a *strukturált programozás,* modularitás, feladatok strukturális dekompozíciójá... egymással rokon elvei. Programfejlesztési eszközeinket is ezen elvek figyelembevételével ...tuk létre.

...A rendszerprogramozási környezet számítógépe

...az AM absztrakt gép

A rendszerprogramozási környezet alapvető számítógépének az általunk definiált AM-gépet ...intjük, programainkat kezdetben erre a gépre írjuk, a később létrehozandó software termékek ...része ezen a gépen végrehajtandó utasításokba fordítódik. A továbbiakban ezen absztrakt ... tervezését és létrehozását ismertetjük.

...1. Portabilitás és absztrakt gép

Az ASSZ körülményeiből adódóan nyilvánvaló, hogy tervezési kritériumaink között fő... ...yen kell szerepelnie a portabilitás (hordozhatóság) követelményének mind a környezet prog... ...eszközeinek, mind a velük létrehozott programok vonatkozásában. A gépfüggetlenségnek a ...nyezet fejlesztésének kiinduló szintjén való biztosítása azért fontos, mert – bár a készített ...ogramok, de fejlesztési eszközök között is lehetnek gépfüggőek – a szükséges esetekben a ...rtabilitás utólagosan nagyon nehezen biztosítható. Ez annál is inkább igaz, mivel nem kívánunk ... a nyelvszabványosítás „gúzsbakötő" gyakorlatával, hanem programozónak „önkifejezésé... maximális szabadságot és a környezet eszközein keresztül egységes módszert adva, őt fel... ...orientált nyelvezet használatára ösztönözzük.

Mindezek alapján a portábilis software termelésének *absztrakt gépre* alapozott technikája ...ellett döntöttünk. [1] Az absztrakt (fiktív) gép az egy feladatcsoport megoldásához leginkább ...megfelelő műveletek és adattípusok együttese által definiált számítógép. Ezen a gépen tehát bi... ...onyos feladatok megoldására könnyen írhatóak hatékony programok, ezek végrehajtásához ...zonban az absztrakt gépnek létező számítógépen realizálva (implementálva) kell lennie.

Az A), B), C) pontokban az AM tervezésének főbb területeit foglaljuk össze.

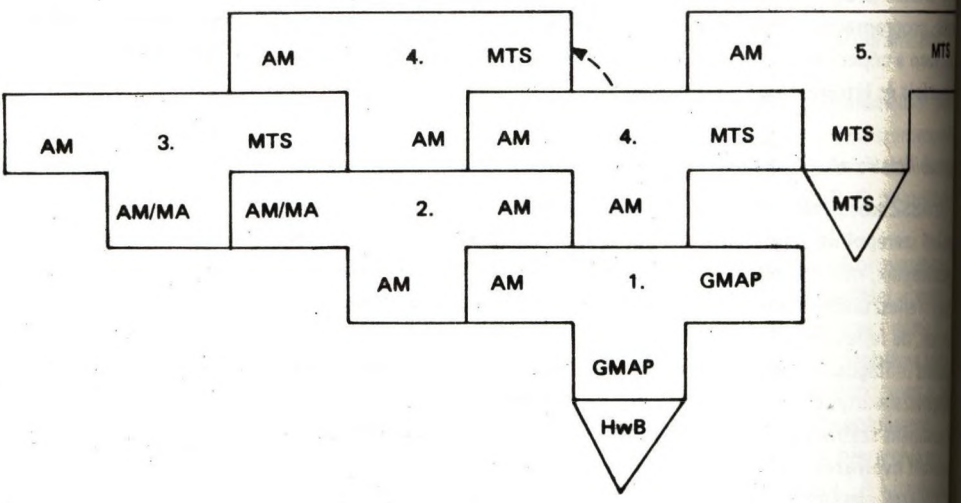
A) Mivel absztrakt gépünk g felhasználási területe és a felhasználás módjai igen szerteágazóak, a hatékonyság csak a konkrét számítógépi lehetőségek teljes megőrzésével biztosítható. Ezért a tervezett AM absztrakt gép lényegében egy *magasszintű általános assembler* programozási nyelv utasításait tudja végrehajtani. Az általánosságot szükségessé teszi a *felhasználás módjainak* változatossága: az AM nyelvet programozáson kívül fordítóprogramok által generált közbülső (intermediate) kódként [2], valamint a szemantika leírásban használt primitív (ismert jelentésűnek feltételezett) nyelvként egyaránt felhasználjuk.

B) Az absztrakt gép és az ASZSZ számítógépeinek *strukturális megfeleltetése* jelentése a legnagyobb gondot. Utóbbiak körében a kiszámítógépek viszonylag homogénebb (16 bit szóhossz, byte-címzés a tipikus) csoportján kívül kiemelt jelentősége van a 36 bit szóhosszúságú, szócímzésű Honeywell 66-os sorozatú (HwB 66) számítógépeknek, annál is inkább, mivel a kisgépekre készülő fejlesztések egy részét is célszerű ezen a kényelmesebb gépen végezni.

A *szóhosszúság* és az *alapvető adattípusok* mellett meg kellett terveznünk a *AM architektúráját* és *utasításait.* Stack-számítógép elképzeléseket elemezve úgy találtuk, hogy jobban járunk, ha egy akkumulátort és indexregisztereket tervezünk, amelyeket konkrét gépi regiszterekkel tudunk reprezentálni.

Problémát jelentett a számítógépek utasításkészlete („tudása") közötti nagy szintbeli eltérés is. Ennek áthidalására – egy absztrakt gép helyett – *absztrakt gépek* hierarchiájának tervezését határoztuk el.

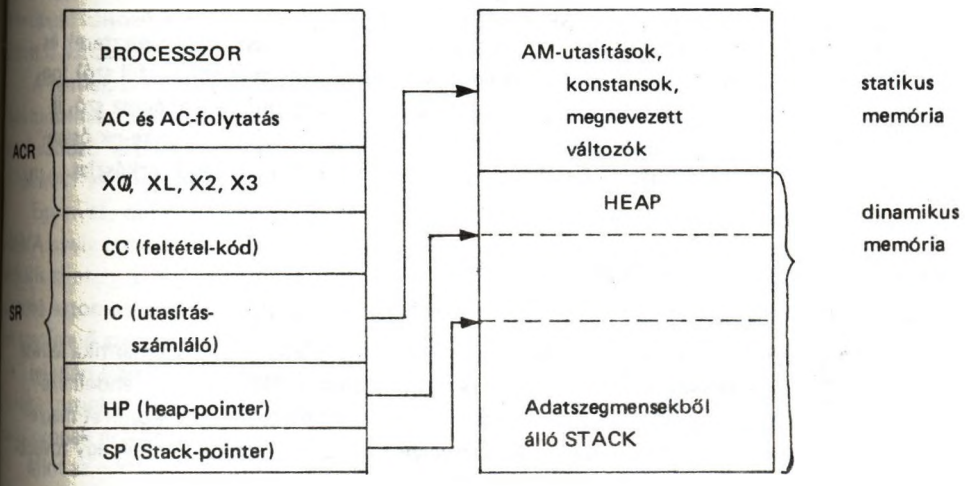
C) A tervezés harmadik területét az AM *implementálási eszközének* meghatározása jelentette. A számbajöhető alternatívák közül a *makróprocesszort* választottuk. A HwB 66 számítógép GMAP makró-assemblere önmagában (run-time rutinok nélkül) alkalmas volt AM realizálására. Kisszámítógépeink általában makróprocesszorral nem rendelkeznek, de az AM leképezése az illető gépre előállító transzlátort egy AM/MA nyelvű programmal (1. ábra 3. elem) az úgynevezett „half bootstrap" módszer [3] alkalmazásával könnyen generálhatunk.



1. ábra

3.2. *Az AM absztrakt gép felépítése*

Az AM processzorból, általános célú regiszterekből (ACR), speciális regiszterekből (SR), statikus (program) és dinamikus memóriából (M) áll. *Architektúrája* a 2. ábrán látható.



2. ábra. *Az AM absztrakt gép architektúrája*

Az *AM utasítások* közül az adatmozgató, aritmetikai, logikai és összehasonlító jellegűek ACR és M, a string-műveletek M és M között érvényesek. Egyéb utasításcsoportok: vezérlésátadó (egyszerű és paraméterezhető eljáráshívások), I/O (logikai-rekordokra), adatgeneráló, területfoglaló és speciális utasítások. Az *utasítások szerkezetét* egy csoportra az 1. táblázatban mutattuk be. Az utasításokban a műveleti kód és regiszternév egybeírandó, az operandus részeit", " választja el egymástól.

opcionális címke	műveleti kód	regiszter név	típus	"," és operandus	
				1. rész	opcionális 2. és 3. rész
	LDR	A	I	azonosító	egész szám
	STR	sp	A		
azonosító			B	XØ	XØ
	ADD	Ø	R	X1	X1
	SUB	1		X2	X2
	IOR	2		X3	X3
	EOR	3			
	AND			S	*
				SL	–
	REL			SG	
				literál	

1. táblázat

Az AM adatmozgató, aritmetikai, logikai és összehasonlító utasításai

A táblázatban sp szóközt, * indirekciót jelöl. Általában a kijelölt cím tartalma az operandus, a — jel hatására azonban maga a cím. Néhány utasítás:

```
KEZD LDRØ I,INDEX     LDR  I, CIMTAB, XØ,  *  ADD I,=17
     STR  I,SL, 5
```

Az AM *absztrakt gép szóhosszúsága* 16 bit (egész szó). Ilyen hosszúak a regiszterek és ez az egész számok (I), címek és pointerek (A) hossza is. Egész szavak felső felét (fél szó) foglalja le a byte (karakter) típus (B). Lebegőpontos műveletek két egymás utáni egész szó, illetve az AC-folytatásregiszterrel kiegészített AC-t veszik figyelembe (R-típus). Byte-ok összefüggő sorozata alkotja a string-típust. A karakterkészlet a standard ASCII karakterkészlet.

4. A rendszerprogramozási környezet alapvető eszköze

— az AM/MA nyelv

Az AM/MA szintaktikus makró alapú kiterjeszthető nyelv a szintaxis és szemantika leírása mellett fordítóprogramok generálására is alkalmas rendszerprogramozási nyelv. A rendelkezésünkre álló keret csak fogalmainak vázlatos ismertetését teszi lehetővé. A nyelv konkrét megjelenési formájában hasonlít a szerző korábbi MADE nyelvére [4, 5] lényegében annak egy továbbfejlesztése.

A továbbfejlesztés eredményeként a MADE nyelv két eleme (a szintaxis leíró nyelv és a szemantikai rutinok írására szolgáló procedurális nyelv) mellett AM/MA-ban megjelennek a kiterjeszthető nyelvek konstrukciós mechanizmusait megvalósító utasítások is. Mivel a szintaxist leíró nyelv a kiterjeszthető nyelvek definíciós mechanizmusaként, a procedurális nyelv pedig magnyelveként funkcionál, AM/MA *(ön) kiterjeszthető* nyelv. Az új szintaktikai konstrukciók létrehozása LL(1) kontextusban történhet, leírásukra a közismert BNF (Backus Normal Form) egy variánsa szolgál. A szemantika leírására (és egyben a fordítóprogram megfelelő részének generálására) — utasításaiban a szintaktikai fogalmakra, azok szemantikai attributumaira hivatkozva — a magnyelvet használjuk.

Összefoglalva: az AM/MA kísérlet a nyelv (szemantika) definíció, a nyelvkiterjesztés és a fordítóprogram generálás feladatának egy közös eszközben történő megvalósítására. Ez az eszköz egy általános makroprocesszor és ebből adódóan sokoldalúan használható egyéb területeken is (editálás, programparametrizálás stb.).

4.1. Szintaktikus makró alapú nyelv és fordítóprogramdefiníció

Az AM/MA nyelv (és családja: a belőle kiterjesztéssel nyerhető hasonló nyelvek) alkalmasak egy programozási nyelv szintaxisának és szemantikájának megadására, valamint e compiler-orientált definíció révén leírt nyelv fordítóprogramjának generálására. Mint más szemantika-definíciók, ez is a nyelv programjainak strukturális dekompozíciójára épül: a *szintaxist* (azaz a programok struktúráját) leíró metanyelv egy szabálya egy szintaktikai fogalmat, annak lehetséges alternatíváit írja le, ahol minden egyes alternatíva alacsonyabb szintű fogalmakból alkotott struktúra.

Egy fogalom *szemantikájának* leírása a szintaxis alternatívái szerint szétágazó programrészletekkel történik. Minden egyes programrészlet az alternatíva alacsonyabb szintű fogalmainak konkrét előfordulásaiból, azok bizonyos attributumaiból kiindulva definiálja azt az algoritmust, amelynek végrehajtása a „program környezetében" a szemantikának megfelelő változásokat hozza létre.

Programkörnyezet alatt ebben az összefüggésben az aktuális fordítóprogramot (nyelvkiterjesztés esete) vagy egy moduláris fordítóprogram-vázat (nyelvdefiníció esete) értünk. Az AM/MA alapnyelv fordítóprogramja az ún. standard környezetet alkotja.

Az AM/MA rendszer használata esetén tetszés szerint használhatjuk fel a standard környezet moduljait, illetve helyettesíthetjük őket új AM/MA-ban írt rutinokkal. Erre az ad lehetőséget, hogy a standard környezet minden moduljának megfelel az AM/MA egy „standard" eljárás-típusú változója, melynek kezdeti értéke az illető modul, de bármilyen más eljárás-típusú értéket is adhatunk neki.

A fentiek szemléltetésére tekintsük a lexikális analizátor (scanner) végzi egy nyelv elemi objektumainak (atomjainak) felismerését. AM/MA-ban a scanner modult a SCAN nevű eljárás-típusú változó értéke adja meg. SCAN standard értéke az AM/MA rendszer kezdeti állapotában az AM/MA nyelv atomjait felismerő AM nyelvű eljárás.

Egy nyelv definiálásakor a lexikális analizátor leírását meg is takaríthatjuk amennyiben az AM/MA nyelv atomjainak struktúráját megfelelőnek találjuk (SCAN:=standard;). Ha azonban a lexikális szinten is „új" a definiálandó nyelv, atomjainak leírására célszerű egy kis (pl. végesállapotú automata-szerű) nyelvtant írni az AM/MA nyelven, a szabályokhoz megadva a szabályok megfelelő szemantikai rutinokat. (A szemantikai rutin mindig a szabály jobboldalára, a „:=" jel után írandó.) Az egész <atomdefiníciós rész> szemantikája az egyes szabályok szemantikai rutinjaiból fog összeadódni. Ezt a „egyesített" szemantikai rutint kell a SCAN eljárásváltozónak értékül adni.

Mint az eddigiekből kiderült, nyelvünk atomjainak vannak bizonyos tulajdonságai, rendelkeznek bizonyos *attributumokkal*, mint pl. valamely atom-osztályhoz való tartozás, érték (változó esetén), szemantikai rutin (szintaktikai fogalom esetén). A felsoroltak egyben AM/MA atomjainak standard attributumai. Mivel nyelvünkben a szemantikai rutin-attributum neve PROC, és a birtokviszony jelzésére a "." szimbólumot használjuk, a fenti példában az új lexikális analizátort a

SCAN:= <atomdefiníciós rész>. PROC; utasítással jelölhetjük ki.

A standard attributumok mellett az AM/MA programban definiálhatunk további attribumokat is. Erre látunk példát a 3. ábrán, ahol változók részére a „type" attributumot definiálunk. (Az attributum típusának definíciója szerint mindössze négy lehetséges értéket vehet fel.) A NUMTYP néven generált eljárás az éppen analizálás alatt álló atom type attributumát kérdezi le és int vagy real típus esetén ad igaz értéket.

```
....
....
REAL    A, B, C;
NEWTYPE TYPETIPUS: int, real, bool, text;
ATTRIBUTE TYPETIPUS type TO <variable> ;
NEWPROC NUMTYP;
NUMTYP:=THIS:type=int \/ THIS. type=real;
DEFINE      <statement> CLASS
"initialise" <variable | NUMRYP>
        [(",," | „and") <variable | NUMTYP>.  NEXT]
"to" <numexpr>  [(",,"  |  „and") <numexpr> . NEXT]   ";"
=
INTEGER n, i;
:=<variable>. NUMBER;
:=1;
:=<numexpr.>. NUMBER /=n THEN
```

```
    ERROR („kiegyensúlyozatlan inicializáció")
else WHILE n<=i DO
    BEGIN
    GENER (Ø, SYMB (<variable>.i),    ":=",
              SYMB (<numexpr>.i), ";" ;
    i:=i+1;
    END;
DEFEND;
.....
x:=2.343;
initialise A, B and C to 1, x, x↑2+1;
.....
```

3. ábra

Egy programrészlet AM/MA nyelven

Az AM/MA fordítóprogram egyik alapeleme egy standard *szimbólumtábla* (ST), melynek kezelését (és ezzel egyben szerkezetének definiálását is) szintén egy standard eljárás-változó értéke végzi. A szimbólumtáblát kiinduló állapotában az AM/MA alapszimbólumai alkotják. Minden egyes eleme tartalmazza az illető szimbólum nevét (mnemonic), valamint attributumait. Egy AM/MA program fordítása során a definiált új fogalmaknak, eljárásoknak stb. megfelelő atomok kerülnek ST-be.

Az AM/MA programokban lényegében egy kibővített *metanyelvvel* definiálhatjuk egy nyelvi fogalom struktúráját. A 3. ábra DEFINE utasításának hatására a <statement> osztály egy új utasítással bővül, egy az „initialise" kulcsszóval kezdődő olyan új utasítással, amelyet minimálisan egy változó, a „to" szöveg és egy pontosvesszővel lezárt numerikus kifejezés követhet. A változó csak numerikus típusú lehet. A szintaktikai kategóriába ágyazott feltétel mező (a" | " jel után) atomelőfordulások szemantikai attributumait (mint esetünkben a type attributum) tartalmazó kifejezés, ez indokolja a „kibővített szintaxis" fogalom használatát.

A meta-nyelvi leírásban az opcionálisan (egy vagy többször) szereplő részeket [és] jelek közé, az alternatív eseteket egymástól | jellel elválasztva írjuk. Mivel ily módon az AM/MA program írásakor egy szintaktikai fogalom konkrét előfordulásainak száma meghatározatlan, a szintaktikus fogalmaknak van egy NUMBER nevű attributumuk, amely ezt az értéket felveszi.

A definiált nyelvi fogalom struktúrája kezdődhet valamilyen konkrét atommal, esetleg kulcsszóval (példánkban ez az eset), de elláthatjuk külön névvel is. Ez a név, ill. kezdő atom bekerül ST-be, PROC attributumaként a definíció jobboldalára (a szemantikai rutinra) történő hivatkozással.

Nyelvi fogalmainkból fogalmi hierarchiákat (ST-láncokat) is felépíthetünk. Bár ez a módszer szépen áttekinthető nyelvdefiníciót ad, hatékonyabb fordítóprogramot kaphatunk, ha nem visszük túlzásba a kategóriák használatát és a szemantikai tevékenységek egy részét is az elemi objektumok eljárásaira bízzuk.

Végezetül a szimbólum-táblába kerülnek a kibővített szintaxisnak megfelelő konkrét atom előfordulások, és így ST végeredményben a konkrét (külső) elemi objektumok és az absztrakt elemi objektumok (szintaktikai fogalmak) egymásnak való megfeleltetését is vezérli.

A szemantika megmaradó részét a DEFINE utasítás jobboldalán szereplő szemantikai rutin tartalmazza, amely a már ismertetett fordítóprogram-elemekben, valamint az ún. szegmensekben változásokat generál. A struktúra konkrét, a kibővített szintaxisnak megfelelő előfordulása a szemantikai rutin explicit meghívását idézi elő. A szegmens-ek tartalma az AM/MA nyelv szövegegyráló utasításaiból (pl. GENER) adódik. Ez felel meg a szokásos makro-kifejtéskor generálódó szövegnek.

A kiterjeszthető nyelvek témaköre mintegy tizenhárom éves múltra tekinthet vissza. Ennek meglehetősen sok vitát kavart területnek az áttekintése [6] többszörösét töltené ki a rendelkezésünkre álló kereteknek.

Véleményünk szerint a nyelvkiterjesztés és a nyelv/fordítóprogram definiálás problémaköre sok közös elemet tartalmaz. Ezért néhány speciális vezérlő utasítás bevezetésével AM/MA-ban egyesítettük a két lehetőséget: az előző fejezetben vázolt nyelvdefiníciós módszer általános nyelvkiterjesztési mechanizmusként is működik. Az egyetlen lényegi különbség, hogy nyelvkiterjesztésnél az egyetlen „működő" compiler időnként *önmagát* írja – részlegesen – felül.

5. Az AM absztrakt gép és az AM/MA nyelv implementálása

Az AM reprezentációját a HwB 66, valamint az MTS 7500 számítógépeken hoztuk létre. A HwB implementáció a GMAP makróassemblerben történő kódolása mintegy 1,5 emberhónapnyi munkát igényelt. A HwB reprezentáció maximális jellegű (valamivel több mint 2000 GMAP utasítás) – valószínű, hogy AM egyszerűbb formában lesz szabványosítható. Eddigi tapasztalataink szerint az AM nyelvű program által generált GMAP kód kb. 30%-kal hosszabb egy átlagos GMAP programozó által írt kódnál. A futási időben nyert veszteség nem éri el ezt a szintet.

Jelenleg absztrakt gépünket még nem tekintjük véglegesnek, a kísérlet még nem zárult le. A teljes gépfüggetlenség megvalósíthatatlanságának és a relatív portabilitásért fizetendő árnak az ismeretében is úgy érezzük, hogy érdemes ezen az úton tovább haladni.

Az első nagyobb AM program az AM/MA kézzel írt fordítóprogramja. (Jelenleg determinisztikus top-down feldolgozást tesz lehetővé.) Erre vonatkozólag nem tudunk hasonló adatokat adni, mert az AM/MA compiler írása nem vált külön a nyelv tervezésétől.

Az AM/MA eszközzel folyó programgenerálási és kiterjesztési kísérletek 1979. májusában (e cikk írásának idején) kezdődtek el. A következő néhány hónapban az előző fejezetekben említett alkalmazásokon kívül egy speciális célú nyelv kerül definiálásra, amely üzemszerű futtatásra kerülő job-sorozatok adat- és program-elemei eseményfüggő dinamikus összefüggéseinek leírására lesz alkalmas. A speciális célú nyelv fordítóprogramja egy már létrehozott programfuttató rendszer által igényelt speciális formátumra fordítja az említett összefüggéseket.

Kiinduló rendszerprogramozási környezetünk használatának, az alapeszközök használhatóságának terén szerzett első tapasztalatainkról az előadásban szeretnénk beszámolni.

Irodalom

[1] Newey, M., Poole, P., Waite, W: Abstract machine modelling to produce portable software, Software—Practice and Experience, 2, 1972

[2] Coleman, S.: JANUS: a universal intermediate language (Ph. D. thesis), Univ. of. Colorado, 1979

[3] Lecarme, O., Peyrolle—Thomas, M: Self-compiling compilers, Software—Practice and Experience, 8, No.2, 1978

[4] Treer, R.: A syntax-macro definition language, SzTAKI Tanulmányok, 50, 1976

[5] Treer, R.: MADE, egy új szintaktikus makró definíciós nyelv, Programozási Rendszerek '75, Szeged, 1975

[6] Solntseff, N., Yezerski, A.: A survey of extensible programming languages, Ann. Review in Aut. Progr. Vol 7, No 5, 1974

A FELHASZNÁLÓK SZŰKSÉGLETEIVEL ÖSSZEHANGOLT SZÁMÍTÓKÖZPONT FEJLESZTÉSE

Tóth Béla
ÉLGAV

1. Röviden a múltról

Az Élelmiszeripari Ügyvitelszervezési és Gépi Adatfeldolgozó Vállalat 1964 óta végez adatfeldolgozási szolgáltatást az élelmiszeripari vállalatok részére. A hagyományos lyukkártya-redszerű feldolgozásról 1968-ban térünk át az elektronikus számítógépek alkalmazására, kezdetben egy, majd rövidesen három azonos konfigurációjú BULL GE 115-ös kisszámítógéppel. A három azonos gép igen kedvező feldolgozási lehetőséget biztosított, az erőforrások azonossága lehetőséget biztosított a szezonálisan jelentkező munkacsúcsok elvégzésére és az alkalmazási programrendszerek kifejlesztésére is. E gépek közül kettő még 1980 végéig üzemelni fog.

A fejlődésünk következő állomását az adatrögzítés korszerűsítése jelentette, 1975-ben a lyukkártyás inputról áttértünk a mágnesszalagos inputra, amelyet az elsők között bevezetett csoportos adatrögzítő rendszer, az MDS 2400-as KEY to DISC SYSTEM üzembeállítása tette lehetővé.

Feldolgozó kapacitásunkat rövidesen kinőttük, mert a kezdeti nehézségeken túljutva felhasználóink számítástechnikai „étvágya" is megnőtt, így döntenünk kellett a Bull gépek kiváltásáról. Vállalatunk ESZR gép mellett döntött, és elsők között helyezte üzembe az R 20-as számítógépet. A BULL GE 115-ös gépről az ESZR rendszerre való áttérés nem ment zökkenőmentesen, a közel egy évig tartó üzemeltetési és alkalmazási ill. áttérési problémákon túljutva 1977-től már három műszakban üzemeltettük az R—20-ast. Beigazolódott a döntésünk helyessée, így 1977-ben az R 20-ast egy R 22-es, majd 1978 végén egy megnövelt kapacitású és 29 MB-os nagylemezekkel felszerelt további R 22-es, a harmadik ESZR gép követte.

2. Az alkalmazási terület jellemzése

Vállalatunk fő profilja az élelmiszeripari vállalatok részére végzett komplex számítástechnikai szolgáltatás, az alkalmazási rendszerek szervezésétől a programozáson keresztül a rendszeres napi három műszakos adatfeldolgozásig. Profilunkhoz tartozik az élelmiszeripari vállalatok részére végzett rendszerfejlesztési tevékenység ellátása is, amely rövidesen a számítástechnikai bázisintézményi funkciókkal is bővül.

Az élelmiszeripar területén a számítógépeket két fő területen alkalmazzák:

- vállalat irányítási célokra, amely nagyrészt adatfeldolgozási feladatokat jelent, és némely területen kiegészül műszaki tudományos számítások elvégzésével is.
- ipari folyamatokban, amely vezérlési ill. mérési-adatgyűjtési feladatokat jelent.

Az élelmiszeripar 13 iparágra tagozódik, szakmai, technológiai, ill. feldolgozandó nyersanyag (termény, élőállat stb.) szerint. Az élelmiszeripar számítógépes adatfeldolgozási igényeinek mintegy 80—85%-át az ÉLGAV számítóközpontja bérmunkában végzi. Az iparágak közül saját számítóközponttal csak a tejipar és a növényolajipar rendelkezik. A bázisintézményi funkciónak megfelelően most létesítünk a söripar részére egy kihelyezett számítóközpontot, amelyet a beindulás és begyakorlás után adunk át üzemeltetésre a söriparnak.

Néhány iparág részére a SZÜV számítóközpontjai is dolgoznak (hús, bor, ill. konzervipar), néhány feldolgozási rendszerben vertikális kooperáció is megvalósult az ÉLGAV és a SZÜV regionális számítóközpontjai között. További kooperációnak gátja a távadat-feldolgozó rendszerek hiánya.

Az élelmiszeripar számítástechnikai fejlesztései az V. ötéves terv időszakában és azt megelőzően is zömében az ÉLGAV-nál koncentrálódtak. A VI. ötéves terv időszakára feladatunk – az élelmiszeripari bázisintézményi funkciónkból adódóan – az ágazat számítógépes alkalmazási rendszereinek ki, ill, továbbfejlesztése, azaz a komplex számítástechnikai szolgáltatás végzésén belül többek között biztosítanunk kell

- a felhasználó vállalataink felkészültségének és számítástechnikai berendezésekkel való ellátottságának koordinált folyamatos fejlesztését, ezzel együtt a gyorsabb és operatívabb termelést segítő hatékony alkalmazási rendszerek bevezetését;
- a rövid és hosszútávú élelmiszeripari tervezéshez szükséges adatbázisok létrehozását és az adatvédelemmel kapcsolatos kutatást;
- a vállalati és ágazati AIR körébe tartozó tervezések és számítógépes programok kidolgozásának irányítását és bevezetését, a bevezetéshez szükséges centralizált számítógépes kapacitás megvalósítását és üzemeltetését;
- az iparági számítóközpontok a decentralizált adatgyűjtő és lekérdező rendszerek létesítésével kapcsolatos rendszerszervezési és irányítási feladatok ellátását, összhangban az élelmiszeripari sajátosságokat figyelembe vevő alkalmazási mintarendszerek kidolgozásával és bevezetésével;
- a működtetett távadatfeldolgozási rendszereink üzembiztonságának növelésén túl, ki kell dolgoznunk jelenleg alkalmazott off-line berendezések üzemeltetési tapasztalatainak felhasználásával az élelmiszeripari távadatfeldolgozási rendszerét és annak folyamatos továbbfejlesztését egy közmű-szerű élelmiszeripari számítógép hálózattá.

3. Azt élelmiszeripar sajátosságai adatfeldolgozási szempontból

Mint ismeretes az élelmiszeripar a mezőgazdasággal szoros kapcsolatban területileg decentralizált Magyarország egész területére kiterjedő állandó, ill. a profiltól függően időszakos (szezonális) felvásárló, feldolgozó és értékesítő szervezetekből áll, zömmel tröszti szervezetben centralizált irányítással. Az egyes iparágak a termelési-feldolgozási profiljukat tekintve különböző technológiával és a technológia által megkövetelt szervezeti rendszerben működnek.

Adatfeldolgozási szempontból a területi széttagoltság és a központi irányítás azt jelenti, hogy egy-egy adatfeldolgozó rendszernek a feldolgozandó elsődleges adatokat azok keletkezési helyén (esetenként többszáz helyen, az ország területén szétszórtan) kell rögzíteni és eljuttatni jelenleg Budapesten működő központi számítógépbe és a feldolgozás eredményeit az irányításhoz, elszámoláshoz szükséges tablókat, átutalási megbízásokat stb. pedig a tröszti irányításon túl a gyáregységekhez, a megyei központokhoz, ill, a kirendeltségekhez kell visszajuttatni.

Az élelmiszeripari trösztök általános vezetési és irányítási rendszerére az jellemző, hogy egy tröszton belül általában három vezetési (döntési) és ehhez kapcsolódó információ szint van, mégpedig

a) tröszti vezetés,
b) vállalati, gyári vezetés,
c) üzem, v, kirendeltségi vezetés.

A vezetés információ rendszerének szerepéből következik, hogy feladatát csak akkor oldja meg jól, ha az adatfeldolgozó rendszer illeszkedik az információs folyamatokhoz, (időben

és térben) ezért a vezetői szinteknek megfelelően az irányításhoz szükséges információ feldolgozását szükségképpen *decentralizáltan* lehet csak megoldani.

Ez egyben azt is jelenti, hogy az adatrögzítéssel, adatgyűjtéssel homogén, az adattovábbító rendszerekkel a vezetői szintekre kellőképpen gyűjtött (tömörített) hierarchikus feldolgozási redszereket kell üzemeltetni, amelyben a decentralizált és a centralizált számítóközpontokban futó rendszerek (alrendszerek) között kellő összhang van, és a feldolgozó rendszerek egy egységes szerkezetű adatbázist képeznek a feldolgozandó és a feldolgozott adatokból.

Példaként talán a Gabona Tröszt jellemzi legjobban a helyzetet. A tröszt 19 megyei vállalata megyénként mintegy 10–12 kirendeltséggel, gyáregységgel, magtárral rendelkezik, így közel 200 helyről kerül a feldolgozandó adattömeg az ÉLGAV számítóközpontjába. Az adatgyűjtéshez jelenleg lyukszalagos technikára kidolgozott TAP–3 adatvégállomásokat használunk, amelyeket a 19 megyei vállalathoz telepítettünk. A gabona felvásárlási rendszer naponta végzi – a felvásárlási bizonylatokból készült lyukszalagos input – a végállomásokról az adatok gyűjtését az ezévben üzembehelyezett MDS vonalkoncentrátorok segítségével, még az adatok visszajuttatása postai úton történik a kirendeltségekig. A banki átutalási megbízások a megye-székhelyekre jutnak vissza.

Az adatfeldolgozás teljes átfutása iparáganként eltérő, egyre inkább a rövid válaszidejű feldolgozási rendszerek felé fejlődik, ami az élelmiszeripar közegészségügyi sajátosságaiból, ill. a termékek romlandóságával, vagy minőségének gyors változásával magyarázható. Példaként a Hűtőipart és a Húsipart említhetjük, ahol a friss áru feldolgozása a félkész-, ill. késztermékek kiszállítása nem maradhat el. Ezen területek adatfeldolgozását végző napi rész-rendszereink számítógépes folyamatai 6–12 órás késedelemnél többet nem bírnak ki, a feldolgozások általában az éjszakai órákban történnek, ahhoz, hogy az áruk kiszállítása már a hajnali órákban megtörténhessen. Szintén naponta készülő feldolgozás a szeszipar élesztő termelésének szétosztását végző programrendszer is, amely a programozott elosztási tervnek megfelelően készíti a kiszállítási ütemtervet, a postai csomagfeladó szelvényeket, valamint az elszámolási tablókat. A kiszállítási tablók késedelme a kenyérgyárak munkáját is akadályozhatja, és emiatt esetleg az ország egyrészében nem tudnak kenyeret sütni.

Az eltérő területi sajátosságok miatt más-más az egyes iparágak, vállalatok számítástechnikai fogadókészsége. Az adatok előkészítése a legkülönfélébb berendezéseken történik, a manuális könyveléstől a kisszámítógépes rendszerekig. Ez a heterogénitás nagyfokú rugalmasságot követel a feldolgozási rendszerektől, hiszen egyik napról a másikra még egy-egy iparágon sem oldható meg az összes adatelőkészítő berendezés lecserélése, hiszen ha p. ismét a gabonaipart tekintjük, akkor egy kisszámítógépes intelligens terminál rendszerre való áttérés a gabonaiparnál egyidejűleg 200 db installálását jelentené, ami hozzávetőlegesen 4–500 millió forint beruházással lenne megvalósítható.

Hasonló a helyzet a többi iparágnál is, így azt hiszem érthető az a koncepció, hogy fokozatosan modulárisan bővíthető rendszereket szabad csak bevezetni és alkalmaznunk.

A bemutatást még tovább lehetne folytatni, azonban ezekből is világosan kitűnik az ÉLGA sajátos szerepe és helyzete a számítógépes bérmunka vállalatok között.

Összefoglalva megállapíthatjuk, hogy jelenleg és a közeljövőben számítóközpontunknak az alábbi követelményeket kielégítő rendszert kell kifejlesztenie:

- a határidőre érzékeny napi feldolgozásokhoz megfelelő megbízhatósági paraméterekkel rendelkező többgépes konfiguráció kialakítása szükséges, amely biztosítja a 6–24 órás átfutási idejű feldolgozások teljesítését,
- a számviteli rendből következően biztosítani kell a különböző iparágak adattárainak védelmét, a reprodukálhatóságot és a hosszúidejű tárolást, miközben a számítógéprendszereink generáció váltása folyik,

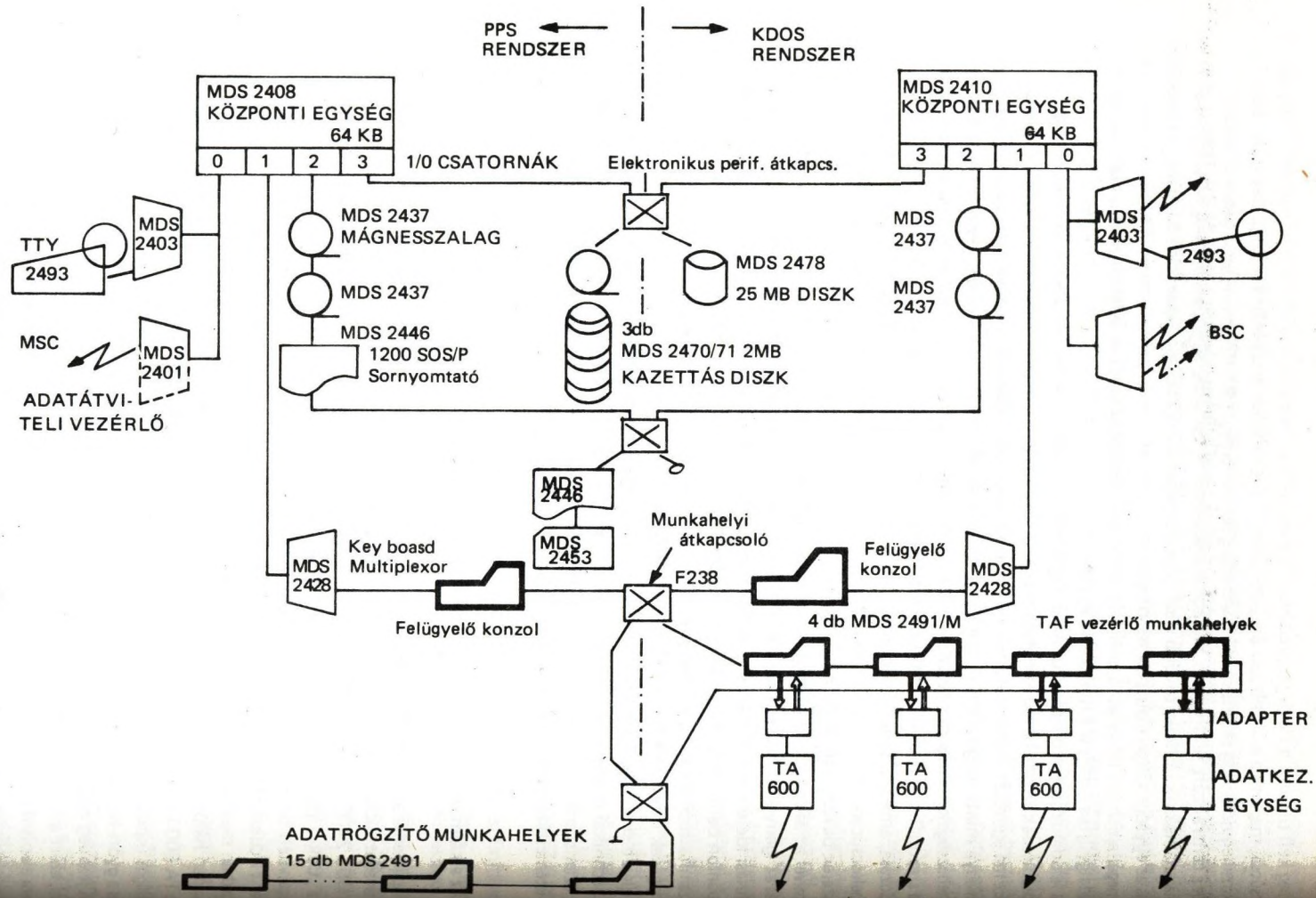
- meg kell oldani a havi, negyedéves és éves zárlati munkák egyidejűleg jelentkező munkacsúcsai mellett a különböző árváltozások, szezonális rendkívüli feldolgozások szintén közel egyidejűleg jelentkező munkacsúcsainak elvégzését is úgy, hogy a decentralizált és a centralizált feldolgozó kapacitásokban rejlő tartalék lehetőségeket optimálisan kihasználhassuk. Ehhez tovább kell fejlesztenünk a működtető operációs rendszereinket és az egységes input/output rendszert;
- megállapítható hogy az újabb területek bekapcsolódásával az adatrögzítés volumene az adatgyűjtés területi kiterjedése egyre nő;
- a feldolgozások komplexitásának (és a részrendszerek integrálásából adódó kapacitás problémák miatt is) növekedtével egyre sürgetőbb az on-line rendszerekre való áttérés. Ez ma egyrészt távadatátviteli hálózati problémák, másrészt a nagykapacitású tömegtárak hiánya miatt még technikailag megoldhatatlan. Ezért úgy gondoljuk, hogy a technikai nehézségek többgépes (több kisebb) számítógéprendszerek kialakításával hidalható át;
- a szezonális és időjárási hatások miatt jelentős a feldolgozásra kerülő tételszámok ingadozása, amely a teljes feldolgozás során jelentős terhelés-ingadozásokat jelent. A terheléscsúcsok emberi okok miatt is csak a területileg decentralizált adatfeldolgozó rendszerek bevezetésével oldható meg;
- mivel az alkalmazási programrendszerek kifejlesztését és bevezetését is vállalatunk végzi és az igények növekedtével nő a programozóink terhelése, ezért a gépreviteli munkák átfutási idejének csökkentése érdekében biztosítanunk kell a programozóink számítóközponton belüli interaktív géphasználatát is.

A számítástechnikai fejlesztési program a VI. ötéves terv időszakára a számítógép alkalmazás ...konyságának növelését tűzte ki központi feladatul.

A fejlesztési erőforrások korlátozott volta viszont a beruházások gondosabb előkészítésére, ...számítástechnikai rendszerek installálás előtti gondosabb tervezésére, a bevezetés előtti minta... rendszerek üzemszerű körülmények közötti tesztelésére és a hatékonyabb erőforrás használatot ...tosító üzemeltetési módszerek kifejlesztésére ösztönöz.

4. Az alkalmazott adatrögzítő és adatgyűjtő rendszerünk

A napijelentések elkészítése és visszajuttatása a felhasználó szervezetekhez ma már elképzelhetetlen távadatfeldolgozás alkalmazása nélkül. A számítóközponton belüli adatrögzítést már ...975 óta MDS 2400-as Key to Disc rendszerrel végeztük. A cukoriparral együttműködve kialakítottuk a „Cukoripari hálózat"-ot, amely a Telefongyár által gyártott TA 600-as lyukszalag... lyukszalagátvitelt biztosító hibavédelmi egységgel felszerelt végberendezésekkel valósult meg. ...rendszer 11 cukorgyárat és az ÉLGAV központját köti össze. A napi feldolgozás a felvásárlási időszakban meghaladja a 2–2 és fél órát. Hasonló berendezésekkel indult meg ez évben a „gabonaipari hálózat", amely a 19 megyeszékhellyel köti össze központunkat. Ebben a rendszerben a TA 600 korszerűsített változatát a TAP–3B alkalmazzuk. A kihelyezett területeken a lyukszalagos inputot lyukszalaglyukasztóval felszerelt könyvelőgépek állítják elő. A gyulai, ...sápai és a győri húsipari vállalat is ugyanilyen rendszerben szolgáltatja a napi feldolgozások ...datait. A jelenleg működő 33 végberendezés a közeljövőben mintegy 50 adatvégállomásra fog ...vekedni, ezért a 15 munkahelyes MDS adatrögzítő rendszerünket egy további MDS 2410-es ...rocessorra alapított vonalkoncentrátori funkciókat ellátó és 25 MB-os lemezegységgel felszerelt ...endezéssel bővítettük. Ennek üzembehelyezése 1979. I. félévében megtörtént, így már az ...ési gabonafelvásárlási kampány során a négy bejövő telefonvonal egyidejű lekezelésére alkalmas ...dszert használtuk, kiküszöbölve ezzel az adatok lyukszalagon történő fogadását.



1. sz. ábra. ÉLGÁV MDS ADATRÖGZÍTŐ ÉS ADATÁTVITELI VONALKONCENTRÁTOR

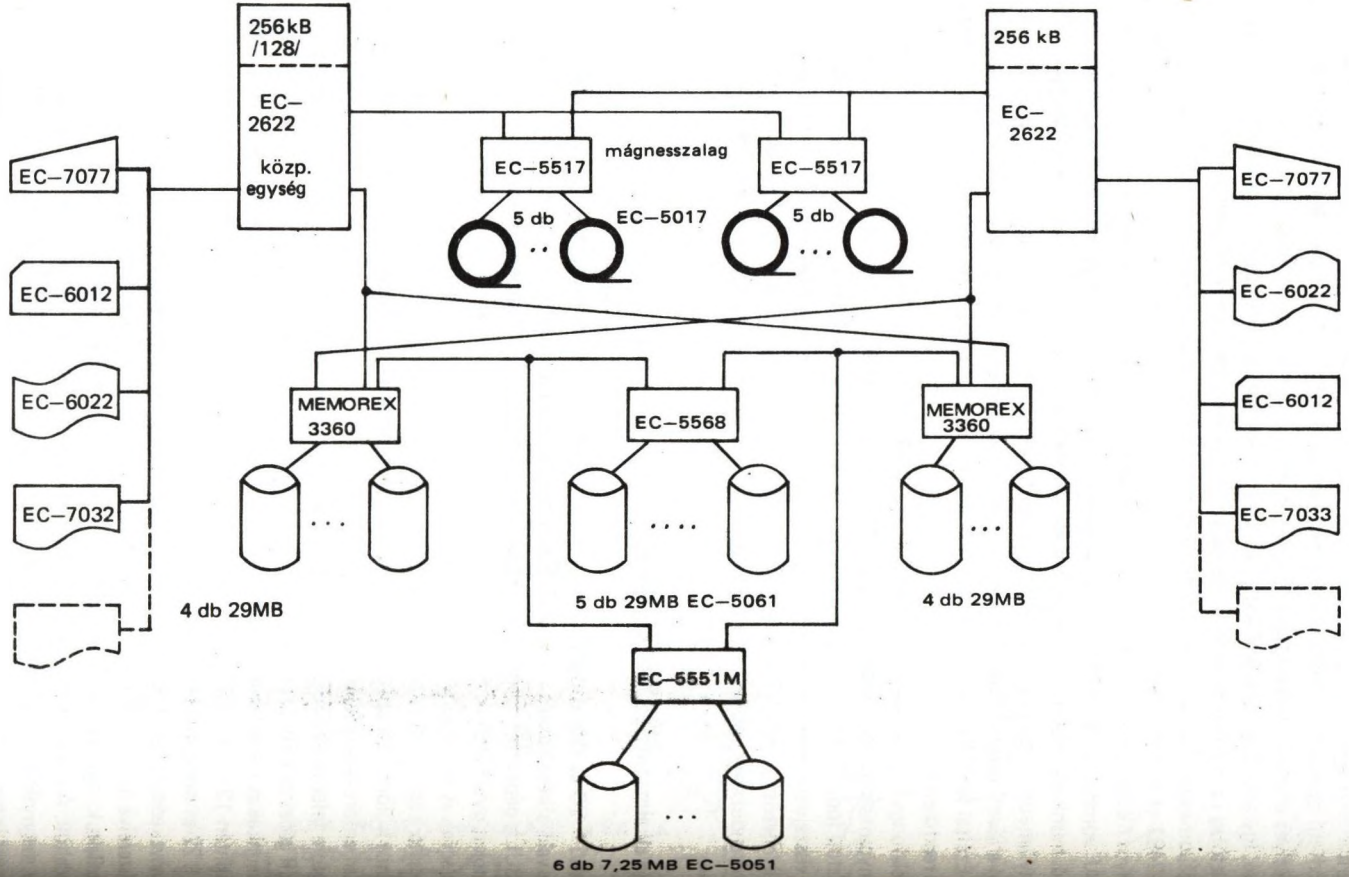
A teljes rendszer felépítése az 1. sz. ábrán látható.

A rendszert úgy építettük fel, hogy a két MDS központi egység köre a perifériákat elektro-ikus és periféria átkapcsoló segítségével lehet telepíteni és evvel a rendszereket átkonfigurálni y a működtető operációs rendszerek cseréjével biztosítható, hogy a szűk keresztmetszetet jelentő munkahelyes adatrögzítőrendszer és a TAF alrendszer üzeme minden körülmények között iztosítható. Így a karbantartások vagy rendszer meghibásodások esetén csak az off-line kon-rtálási illetőleg az output printelési munkákat kell időszakosan felfüggeszteni. A távadatátvi-li munkaállomások lyukszalagos periféria oldalát illesztettük — az MDS céggel együttműködve - az MDS 2491-es display-rendszerű adatrögzítő munkaállomáshoz, ezzel megtartottuk a TA 00-k előnyös hibavédelmi tulajdonságát ugyanakkor ugyanazon fizikai rekordszerkezetre képez-ük le a TAF vonalakon érkező adatokat, mintahogyan az a KDOS rendszerben a billentyűzött datrögzítői munkahelyekről történik. Ez által létrehoztuk egy olyan rugalmas adatgyűjtő rend-zert, amely lehetővé teszi mind a bizonylaton érkező, számítóközponton belüli adatok rögzí-tését, mind a távoli terminálokon keresztül érkező ugyanazon szerkezetű inputok egyöntetű ezelését. Ezáltal lehetővé vált, hogy a távfeldolgozásra felkészült felhasználóink fokozatosan be-lépjenek a rendszerbe.

A Telefongyár időközben kifejlesztette a TAP—3D típusú Videoton displayel felszerelt is mátrix nyomtatóval is kiegészíthető végállomását,, amely így nemcsak az adatgyűjtésre, ha-nem az adatok fogadására is alkalmassá vált. Az MDS rendszerünk az illesztés révén alkalmas az adatok visszajuttatására is. További terveink között szerepel az MDS 2400-as rendszer szinkron adatátviteli lehetőségeinek kihasználásával (BSC) on-line kapcsolat megteremtése egyrészt intel-igens végberendezések, másrészt a feldolgozó ESZR nagy számítógépek között.

5. Az alkalmazott ESZR konfiguráció

Az első R 20-as számítógép üzemeltetési tapasztalatai azt mutatták, hogy a feladatok adott határidőre való elkészítésére — a jelentősen nagyobb teljesítmény ellenére — önmagában nem nyújtja a BULL gépeknél megszokott megbízhatóságot. A Bull gépek üzemeltetése során kedvező tapasztalatokat gyűjtöttünk össze az azonos konfigurációjú — egy géptermen belüli — gépek üzemeltetésével. A több azonos konfigurációjú, viszonylag kisebb teljesítményű géppel igen ked-vező átbocsátó képességű rendszert lehet kialakítani, mert ezek nemcsak gépmeghibásodások ese-tére, hanem a különböző software problémák megoldására az inkompatibilitási kérdések tiszta-tására a karbantartások idejének célszerű ütemezésére stb. igen rugalmas rendszert biztosítanak. Az R 20 gépre telepített programok futtatása során rövid idő alatt nyilvánvalóvá vált, hogy to-vábbi ESZR gépeket kell üzembehelyezni ahhoz, hogy a napi 3 műszakban üzemelő Bull gépe-ken futó munkákat megbízhatóan áttelepíthessük. Az 1977-ben üzembehelyezett második ESZR gép még szintén 128 kB-os memóriával és 7,25 MB-os kislemezekkel felszerelve került leszállí-tásra. Az R 22-es nagyobb sebességéből a perifériák lassúsága miatt nem sikerült számottevő kapacitásnövekedést elérni, ezért a harmadik R 22-es rendszerünket már 29 MB-os nagyleme-zekkel felszerelve állítottuk üzembe 1978 végén. Ezt a harmadik rendszert már 256 kB-os memó-riával installáltuk, egyidejűleg az első R 22-es kibővítését és nagylemezekkel való felszerelését is megrendeltük. Az iker R 22 számítógéprendszer kialakításának, amelyet a 2. sz. ábrán mu-tatunk be, megnyílt a lehetősége és ehhez a rendszerhez egy nagylemezes DOS—POWER operá-ciós rendszert is generáltunk. A kialakított iker rendszer további fejlesztési lehetőségeket is biz-tosít, nevezetesen azt, hogy a közös használatú lemeztárak egyidejű elérésének megoldásával modelezhetjük a VI. ötéves terv során kialakítandó rendszerünket.


256kB
/128/
EC–
2622
közp.
egység
256 kB
EC–
2622
EC–7077
EC–6012
EC–6022
EC–7032
EC–5517
mágnesszalag
EC–5517
5 db
EC–5017
5 db
EC–7077
EC–6022
EC–6012
EC–7033
MEMOREX
3360
EC–5568
MEMOREX
3360
4 db 29MB
5 db 29MB EC–5061
4 db 29MB
EC–5551M
6 db 7,25 MB EC–5051

A két azonos gép (és az R 20-ashoz képest megbízhatóbb) üzemeltetése nagylemezes rend-rel mindenképpen biztosította az adott feladatok elvégzéséhez szükséges kapacitást így sor ülhet az első Bull gépünk leselejtezésére. Az R 20-as számítógépünket (a miskolci SZÜV szá-központban kidolgozott újítás átvételével) a BULL GE 115 számítógéppel átmenetileg ösz-kapcsoljuk, és ez fogja képezni a söripari számítóközpontban az induló konfigurációt, amelyet 0-ban egy virtuális terminál kezelőrendszerrel fogjuk kiegészíteni a „söripari hálózat" meg-ntéséhez. A VI. ötéves terv folyamán az ÉLGAV központ fejlesztésével egyidejűleg a 20-as et az R 22-es rendszer fogja kiváltani. Ezáltal biztosíthatóvá válik a jelenleg BULL gépen futó ipari feldolgozások folyamatos konvertálhatósága az ESZR rendszerre.

6. A fejlesztés következő lépcsőfoka

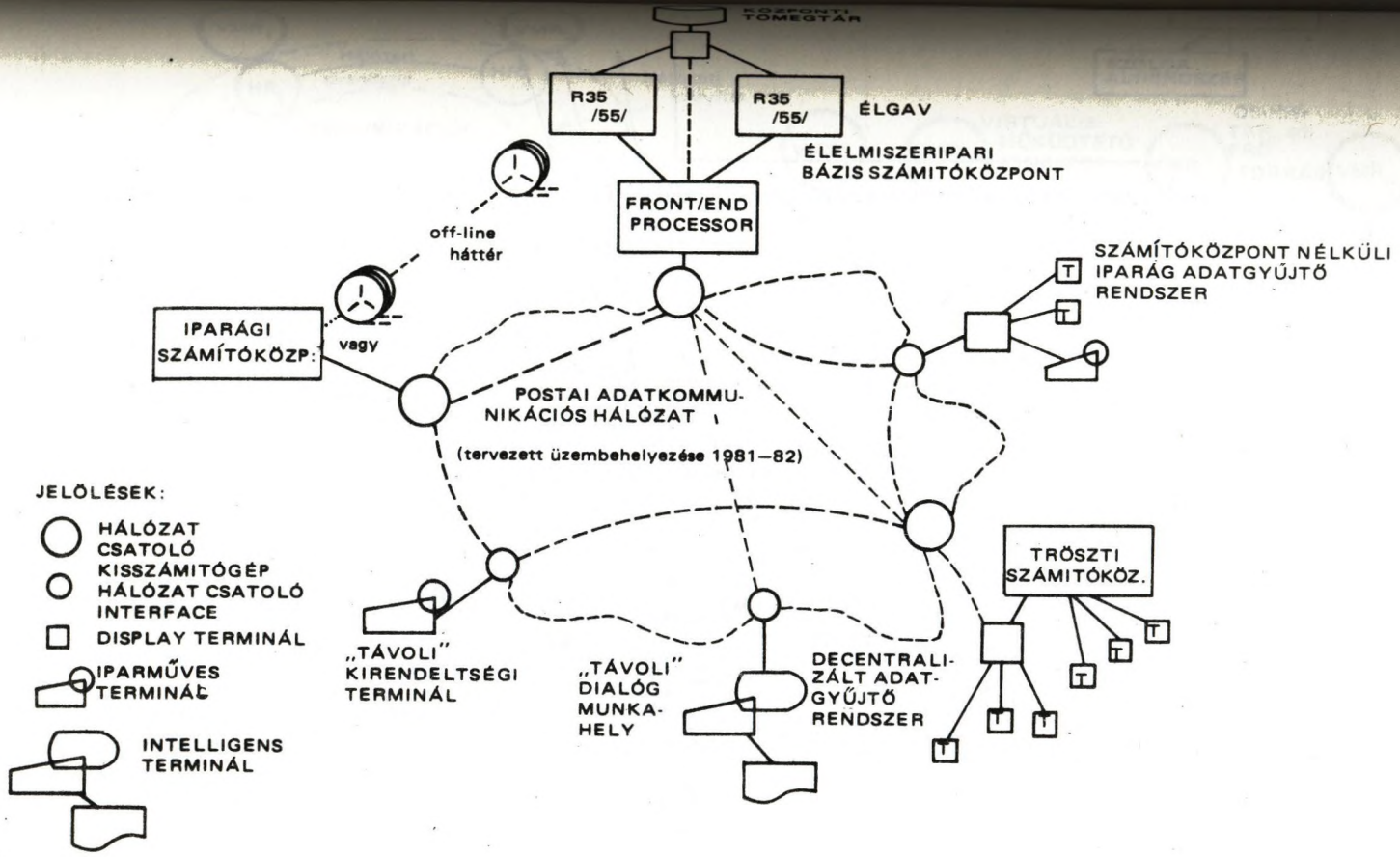
Az előzőekben megfogalmazott elvek és az élelmiszeripari sajátosságok ismeretében ki-ndható, hogy a VI. ötéves terv időszakában is az ÉLGAV központi számítástechnikai ka-citása lesz az élelmiszeriparon belül a meghatározó. Ezt a rendszert kell úgy kialakítani, hogy észt kiszolgálja a jelenleg működtetett mintegy 100 felhasználói rendszer folyamatos működ-ését, másrészt biztosítsa az új hatékonyabb on-line rendszerek kialakításának lehetőségét. A minálok számának és az adatok volumenének növekedésével a számítógépes rendszer on-line ocsátó képessége jelentősen csökken, így ebben a fejlesztési fázisban mindenképpen ükséges egy decentralizált számítógépes hálózatra áttérni. A folyamatos átmenet biztosítása ekében egy tervezett rendszer indulásakor néhány ponton többlet kapacitás beruházása válik ükségessé. Ez elsősorban a távfeldolgozó rendszerben jelent előnyt, hiszen egy programozható ront-end kisszámítógép mint kommunikációs processor a hardware megoldásokkal szemben ételen előnyös tulajdonságokkal rendelkezik.

A 3. sz. ábrán bemutatom a hierarchikus decentralizált rendszer-komponensek kapcsolatát, g a 4. sz. ábra a VI. ötéves terv végéig kialakítandó élelmiszeripari bázis számítóközpont fel-ését tünteti fel. A rendszer kiépítése mai árszinten mintegy 200 millió forintból valósít-ó meg, ami három lépcsős megvalósítás esetén összhangban van a rendelkezésünkre álló lesztési forrásokkal.

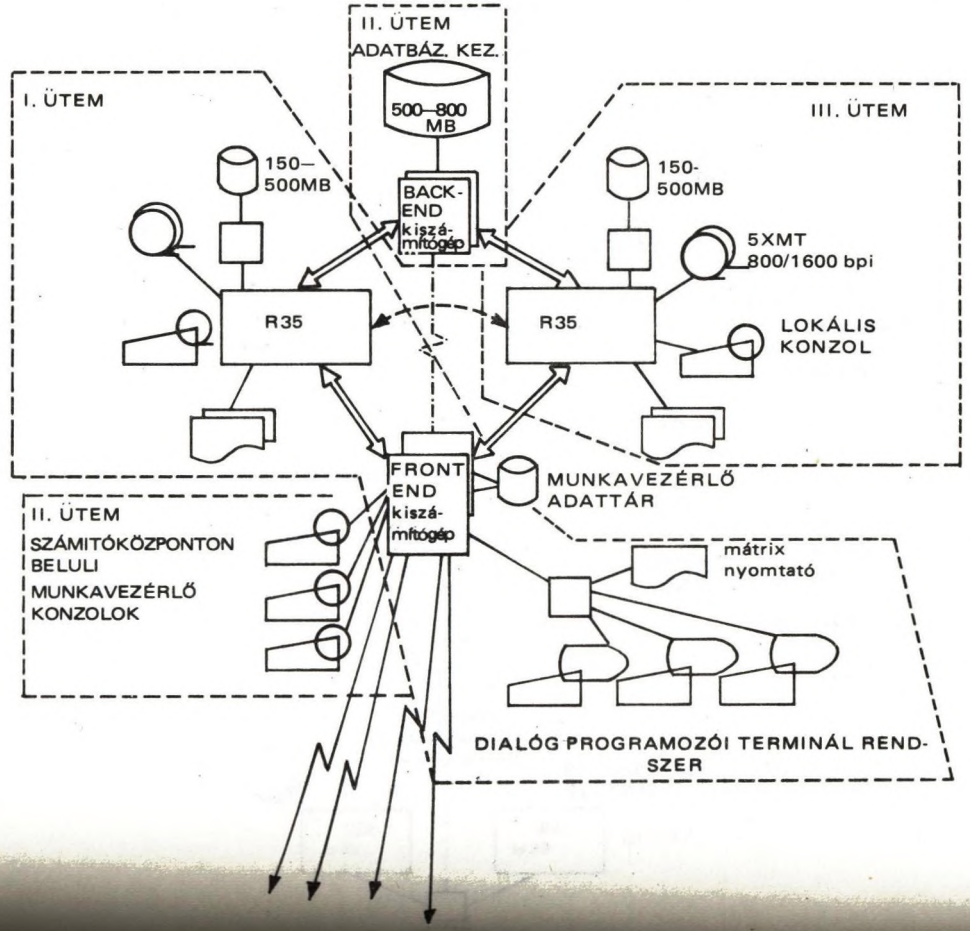
Az első ütemben 1981—82-ben egy R 35-ös és egy front-end rendszer üzembehelyezését vezzük, amely alkalmas a teljes operációs rendszer kifejlesztésére és az új munkák beindítá-a a még működő iker R 22-es számítógéprendszer mellett. A második ütemben 1982—83-ban működő rendszert kiegészítjük egy BACKEND adatbázis kezelő rendszerrel és megoldjuk a táv-datviteli vonalak csatlakoztatását. Ennek során — együttműködve a számítástechnikai fej-esztési intézetekkel — biztosíthatónak látjuk az 5. sz. ábrán részletezett koncepciónak meg-elő hálózati struktúra kialakítását, amelynek részleteiben való taglalása meghaladja ezen előadá-m kereteit. A leglényegesebb alapgondolat, hogy a hálózati alrendszer elemeit decentrali-tan valósítsuk meg, a már működő virtuális operációs rendszer alrendszerként. Ezáltal a meg-ható számítógép hálózati funkció és a korábban már futtatott kötegelt feldolgozások almasabban illeszthetők egymáshoz.

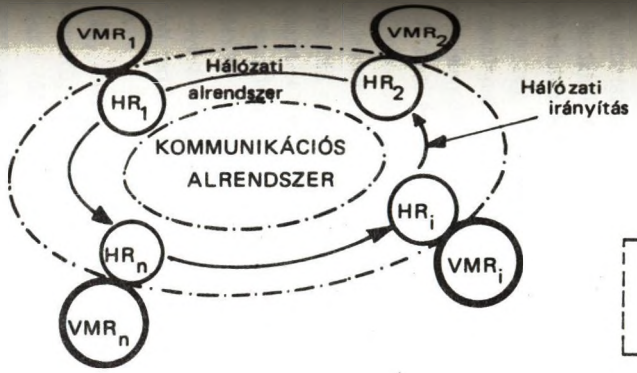
A 6. ábra a tervezett és BACKEND funkciót ellátó adatbázis kezelő rendszer fizikai és logi-vázlatát tünteti fel. Ennek előnye, hogy a harmadik ütemben üzembehelyezendő második 5-ös rendszerrel együtt lehetővé teszi a számítóközponton belüli munkák ütemezését, va-int meghibásodások esetére a részben automatizálható helyreállítást. Az irányítás-orientált dszerek sajátossága, hogy a megbízhatóság elsősorban a file-ok illetve az adatbázisok véde-re ír elő szigorú követelményeket.

Tekintettel a kisszámítógép piac (és a hazai gyártás) gazdaságos fejlődési irányára a különböző rendeltetésű kisszámítógép alapú rendszerkomponensek kifejlesztésének nincs akadálya. Ugyancsak ennek figyelembevételével tervez ünk decentralizált feldolgozó rendszert, amely a hagyományos számítóközpontot fokozatosan kollektív használatú vezető (irányító) számítóközponttá változtatja.

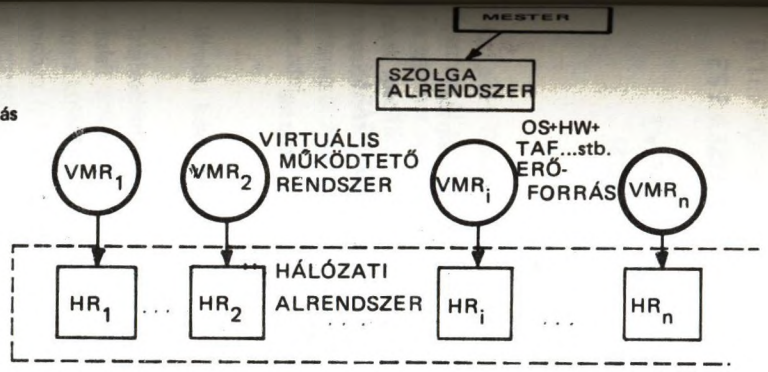


3. sz. ábra. A VI. ÖTÉVES TERV VÉGÉIG KIALAKITANDÓ, ÉLELMISZERIPARI ADATFELDOLGOZÓ RENDSZER FELÉPITÉSE

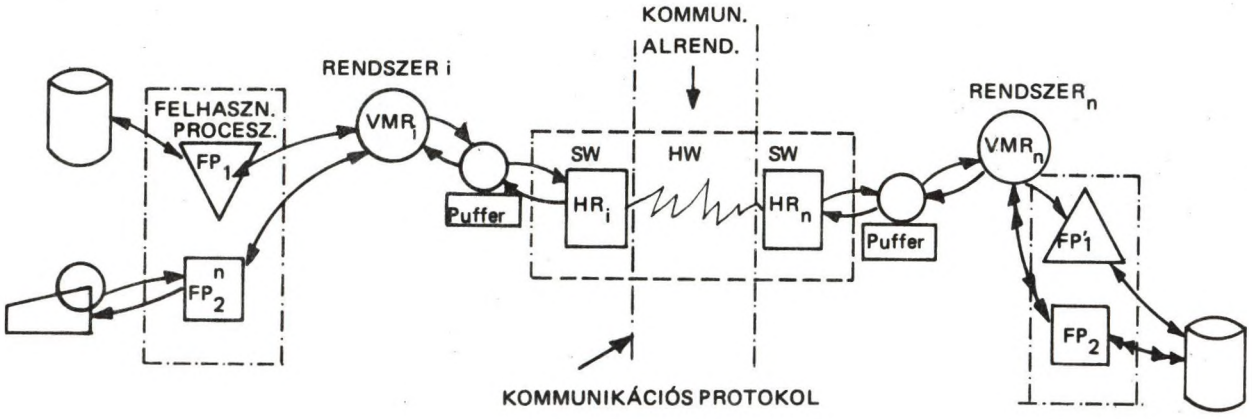

II. ÜTEM
ADATBÁZ. KEZ.
500–800 MB
I. ÜTEM
III. ÜTEM
150–500MB
BACK-END kiszá-mítógép
150-500MB
5XMT 800/1600 bpi
R35
R35
LOKÁLIS KONZOL
FRONT END kiszá-mítógép
MUNKAVEZÉRLŐ ADATTÁR
II. ÜTEM
SZÁMITÓKÖZPONTON BELULI
MUNKAVEZÉRLŐ KONZOLOK
mátrix nyomtató
DIALÓG PROGRAMOZÓI TERMINÁL RENDSZER



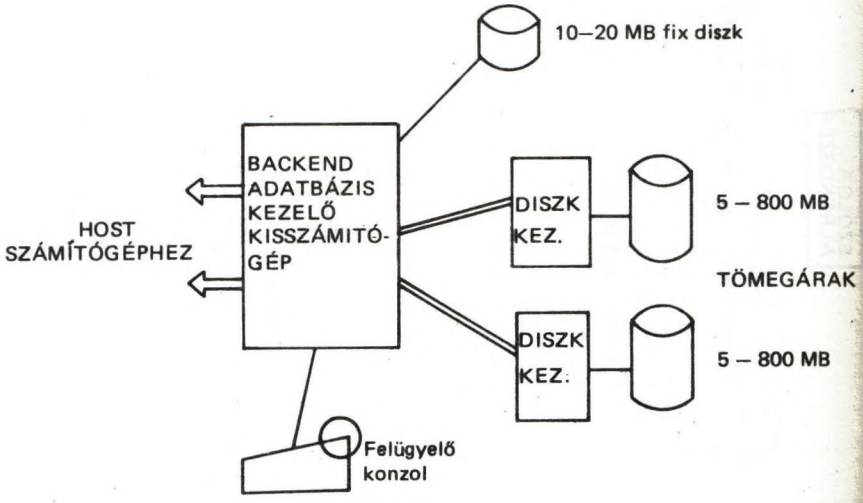
5/a. ábra. RENDSZER ELEMEK KAPCSOLATA

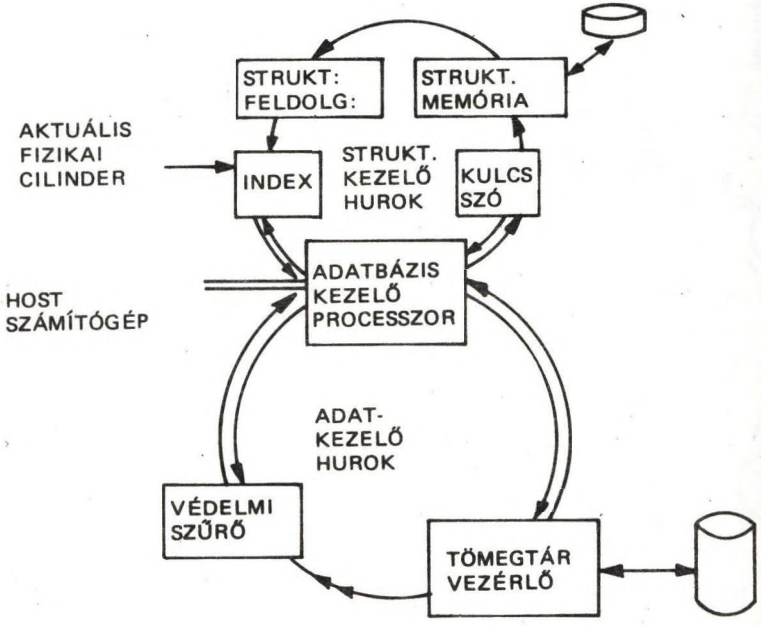


5/b. ábra. VEZÉRLÉSI KAPCSOLAT



5/c ábra. PROCESSZEK KAPCSOLATA A RENDSZER ERŐFORRÁSOK KÖZÖTTI ÚTVONALON





6. ábra. FIZIKAI ÉS LOGIKAI VÁZLAT
A BACKEND ALRENDSZERHEZ

INTERPOLÁCIÓS ELJÁRÁS, MEGBÍZHATÓSÁGI VIZSGÁLATOK ÉS RAJZOLÓ PROGRAM GEOTECHNIKAI FÜGGVÉNYEKRE

Dr. Varga Gyula — dr. Abaffy József — Reményi Péter
MTA SZTAKI MTA SZTAKI FTV

Az Építési Geotechnikai Adattárról, annak építőipari szerepköréről és technikai-technológiai fejlesztéséről vázlatos áttekintést adtunk egy másik tanulmányban. Ebben hivatkoztunk rá, hogy az egységes adatkezelő-feldolgozó rendszeren belül több olyan program került kidolgozásra, melyek specifikus építőipari mérnöki igényeknek megfelelően alapvetően eltérnek a szokványos megoldásoktól, gyári software szolgáltatásoktól. Jelen referátumban a mérnöki előkészítés és előtervezői feladatok szempontjából kiemelt jelentőségű programról kívánunk rövid tájékoztatást adni.

Mérnöki követelmények

Talajmechanikai-geotechnikai függvényeken a terepen végzett feltárások (fúrás, akna, stb.) révén észlelt építésföldtani adottságokat, illetve a felszínre hozott talajminták laborvizsgálata során nyert különféle számszerű paraméterek (talajfizikai- és állapotjellemzők) értjük a hely függvényében (pl. az adott réteg vastagsága vagy felszíne, a talajvíz terepszint alatti mélysége, a talaj viztartalma, teherbíró képessége, stb.).

Rendkívül fontos hangsúlyozni, hogy az építőiparban a beruházások telepítési döntése (térbeli elhelyezése) számos vonatkozásban eleve determinálja a megvalósítás hatékonyságát, hiszen az ország, sőt egy adott település területén is döntően eltérő beépíthetőségi feltételekkel kell számolnunk. Ez az alkalmazható műszaki megoldásokra, illetve a bekerülési költségekre egyaránt jelentős befolyással van.

A telepítési döntés után a tervezett beépítés lehatárolt területén belül is lokális eltérésekkel, különbözőségekkel kell számolni alapozás-mélyépítés-közművesítés szempontjából, hiszen a talajrétegződés, az építésföldtani adottságok vonatkozásában homogén és izotrop feltételekkel még egyetlen réteg esetében sem számolhatunk, nemhogy egy nagyobb terület teljes rétegösszetétele vonatkozásában. Így a kijelölt terület határain belül sem közömbös a kivitel műszaki megoldása és költsége szempontjából, hogy az egyes épületek milyen geometriai elrendezésben kerülnek elhelyezésre.

Nyilvánvaló, hogy az építőipari, sőt népgazdasági szempontból optimális helykijelöléshez az adott terület feltártsága, adottságainak ismeretességi foka döntő tényezőként veendő figyelembe.

Egy településen, vagy akár országosan is a feltárások és vizsgálatok területi megoszlása véletlenszerű, a kötelező talajvizsgálatok elrendelése után épült létesítményekhez kapcsolódnak. Egyrészt a földtani kialakulás, másrészt az egymáshoz mért távolság függvényében a feltártság és ismeretesség (megkutatottság) mértéke eltérően értékelhető.

Az építési feladatok egyértelműen területhez kötöttek. Így a beruházások telepítési döntéséhez, majd az építési fizikai tervezéshez is területi ábrázolás, térképi vagy helyszínrajzi részletességű megjelenítés szükséges. Tájékoztatási igény esetén a kartogrammok is kielégítőek, a döntések és tervezés azonban már pontosabb, általában paramétereket rögzítő izovonalas dokumentációt követelnek. A szintvonalas domborzati térképek számító- és rajzgépi szerkesztése-

előállítása ma már konvencionális megoldásnak tekinthető. A terület beépíthetőségének komplex megítélése azonban a geotechnikai-építésföldtani jellemzők egész sorozatának hasonló formájú előállítását teszi szükségessé. Így az izovonalas térképszerkesztést megelőző interpoláció, majd a szerkesztés és rajzolás a mérnöki előkészítés és előtervezés egyik leginkább munka- és időigényes fázisa. Automatizálása tehát nemcsak a műszaki tervezési technológia, hanem a beruházások hatékonyabb előkészítése szempontjából is kiemelt feladat.

A program vázlatos ismertetése

Egy területrészt feltártnak tekintünk egy bizonyos talajmechanikai függvényre nézve, ha ott az illető függvény szintvonalai elegendő pontossággal és megbízhatósággal rendelkezésünkre állnak. Mivel egy fúrás és a kivett minták analizálása 10 000 Ft nagyságrendű költséget jelent, ezért a fúrások számának növelése nem gazdaságos, helyette a meglévő fúrások ismert paraméterértékeiből kiegyenlítő számításokkal és interpolációval kell dolgoznunk. A számítási módszerek megválasztásánál figyelembe kell vennünk a talajmechanikai függvények kedvező és kedvezőtlen tulajdonságait. Ezek közülnéhány fontosabbat felsorolunk.

1. A függvények gradiensének normája kicsiny, a függvényt ábrázoló felületen meredek domborulatok általában nincsenek.
2. A függvények tartalmazhatnak konstans platókat, szélsőértékeket és nyeregpontokat.
3. Értelmezési tartományuk nem egyszeresen összefüggő.
4. A függvényértékek nem negatívok.

Mindezeket figyelembe véve célszerűnek látszott, hogy a fúrásokat most már egy térképszelvényen tekintve, a rendszertelenül elhelyezkedő fúráspontokban ismert függvényértékek rendszerét egy négyzetrács-rendszerrel helyettesítsük, úgy, hogy a rendszer minimális információveszteséget szenvedjen. E célból a négyzetrács élhosszát úgy választottuk meg, hogy egy résznégyzetbe, a határoló oldalakat is beleszámítva, átlagosan 4 fúráspont essék. Ezután a legkisebb négyzetek módszerével számítottuk ki a függvényértékeket az egyes résznégyzetek középpontjában. Ha a négyzet középpontja a négyzetbe eső fúráspontok által meghatározott sokszög konvex burkán belül esett és számuk legalább 4 volt, akkor egy, a térbeli pontokhoz a legkisebb négyzetek elve értelmében legközelebb fekvő sikot határoztuk meg, s a résznégyzet középpontjának koordinátáit behelyettesítve kaptuk meg a keresett interpolációs közelítő függvényértéket. Ha a konvexitás nem teljesült, vagy a pontok száma a négyzetben kevés volt, akkor a négyzetet bővítettük az őt körülvevő 8 (a széleken kevesebb) négyzettel. Ezt a bővítést kétszer ismételhettük meg. Figyelembe vettük azt az esetleges szerencsés körülményt is, ha a négyzet középpontjának elegendően kicsiny környezetében fúráspont volt, ebben az esetben a fúráspontban ismert paraméterértéket elfogadtuk függvényértékként a négyzet középpontjában. Ha a térképszelvényen volt olyan területrész, ahol a talajmechanikai függvény nem volt értelmezve, akkor azt a területrészt elegendő sok úgynevezett fiktív fúrásponttal vettük körül, s ha valamelyik négyzet középpontja az esetleges bővítések után a fiktív fúráspontok által meghatározott sokszög konvex burkán belül volt, vagy nem volt körülötte egy valódi fúráspont sem, akkor a négyzet középpontjában nem tudtunk függvényértéket kiszámítani. Annak megakadályozására, hogy a bővítés során egy értelmezési tartományba nem tartozó területrészt átinterpoláljon a módszer, alulról korlátoztuk az értelmezési tartományba nem tartozó területrészek minimális átmérőjét.

A négyzetrács résznégyzeteinek középpontjában a fenti eljárással kiszámított közelítő függvényértékek megbízhatóságának mérésére szolgáló megbízhatósági függvényt heurisztikus megoldások segítségével állítottuk elő. Abból indultunk ki, hogy további fúráspontokra támaszkodva az eljárás kevésbé megbízható függvényértéket ad a négyzetek középpontjában,

ha nem kell a négyzetet bővíteni. Ha tehát tekintjük a függvényérték kiszámításához fel-
mélt fúráspontokat és a négyzet középpontjára vonatkoztatva képezzük a fúráspontokhoz
ozó függvényértékek súlyozott átlagát a távolságok reciprok értékének négyzetével mint
okkal, akkor bővítés esetén e súlyozott átlag és a legkisebb négyzetekkel kapott függvény-
k relatív eltérése nagyobb lesz, mint ha nem kell bővíteni a négyzetet. Ezért a négyzet kö-
pontjában kiszámított függvényértéket f-fel, az előbbi súlyozott átlagot g-vel jelölve a meg-
zatósági függvényt az

$$M = \min (f,g;\ g(f) \times 100\%$$

ettel vezettük be. Ezt minden résznégyzet középpontjára kiszámítva, és a megbízhatóságra
lmas küszöbszámot megadva eldönthettük, hogy érdemes-e a rácshálózat alapján szintvona-
t rajzoltatni a továbbiakban ismertetendő rajzoló programmal, vagy a rendszer újabb fúrá-
kal kiegészítésre szorul.

Ha szükség van a rendszer kiegészítésére, akkor ki kell tűzni az új fúrás vagy fúrások
yét. Ezt, – figyelembe véve azt a körülményt, hogy egy fúrást több függvényérték kiszá-
tásához is felhasznál az eljárás, – optimálisan egy maximum-minimum feladat megoldásá-
l hajthatjuk végre. Ez a kijelölt résztartományban egy kezdeti pontból kiindulva a

$$\max_{P} (\min_{j} \overline{PPj})$$

tételnek eleget tevő P pontot adja eredményül, ahol P a kijelölt résztartományban lévő fú-
spontokat jelenti.

Az új fúráspontokban megkapott paraméter-értékek ismeretében az eljárás programját
a lefuttatjuk változatlan rácstávolság mellett, s ha a megbízhatóság megfelelő, akkor a rács-
ntokban kapott függvényértékek segítségével a talajmechanikai függvény szintvonalait kiraj-
ztathatjuk. Szintvonalat akkor kapunk, ha a megadott paraméterérték a rácshálózatban elő-
rduló legnagyobb és legkisebb függvényérték közé esik.

A szintvonalak kirajzolására általunk készített eljárás programja először megvizsgálja a
cshálózat 4 szomszédos csúcspontja által meghatározott négyzetet, halad-e át rajta szintvo-
al.Ha igen, lineáris interpolációval meghatározza a négyzet oldalain azokat a pontokat, ame-
yekből a szintvonal kiindul. Ezután felderíti az esetleges nyeregpont szituációt, majd alkal-
as görbe vonalakkal a megfelelő pontokat összeköti. Az alkalmazott görbék trigonometri-
us görbék, valamint $a^2x^c+b^2y^c = 1$ alakú görbék, ahol $C \geqslant 1$. Mivel ezek a görbék a szint-
onal közelítései, előfordulhatna, hogy különböző paraméterértékekhez tartozó görbék metsze-
ik egymást. Ez a hiba nem fordulhat elő, ha az egyes görbék végpontjai a négyzet oldalain
incsenek közelebb, mint a négyzet oldalhosszának a negyede.

A rajzoló eljárás az egymás után következő négyzetek sorrendjét optimalizálja, hogy a
ajzoló berendezés felemelt helyzetben lévő felesleges tollmozgatásait minimálisra csökkentse.

Az eddigiekben leírt eljárások programjai a Magyar Tudományos Akadémia CDC 3300-as
pén készültek USASI FORTRAN nyelven. A programok csak a gyors-memóriát használják
el és a DAISY file lehetőséget.

Ismételten kiemeljük, hogy a négyzetrács hálózat minden területbővítési szakaszában el-
ző jelzés hívja föl a tervező figyelmét a függvényérték megbízhatósági fokának csökkenésére.
záltal már szemléletileg is azonnal feltűnően minősíti a rajz a terület feltártságát, illetve a
endelkezésre álló műszaki ismeretek megbízhatóságát. Itt külön föl kell hívnunk a figyelmet
ra, hogy a feldolgozásra kerülő különböző függvényértékek megbízhatósága az adott feltárt-
g (fúráspontok térbeli eloszlása) esetében sem feltétlenül azonos. Ennek mérlegelése termé-
zetszerűleg már nem bízható a gépre, hanem a szakértő műszaki feladata és felelőssége.

Ebből szükségszerűen következik az is, hogy a gép által „kitűzött" szükséges új feltárási vizsgálati pont(ok) csak geometriai optimalizálást jelenthet, melyet a szakmai meggondolások átértékelhetnek, felülbírálhatnak.

Mindezen korlátozásokkal is leszögezhetjük, hogy a program számottevően gyorsítja és javítja a tervező tájékozottságát az adott terület megkutatottságáról. Ezáltal az új feladatnak megfelelő – esetleg szükséges – feltárási terv elkészítését és a szerződéskötést minőségileg harmonizálja és gyorsítja. Gyakorlatilag ez azt jelenti, hogy a hagyományosnál pontosabban meghatározott vizsgálatok a fölösleges kapacitás- és költséglekötést, illetve a terven felüli többletátfordításokat minimalizálhatják, a tervszerűség színvonalát javíthatják.

Gyakorlati alkalmazások

A programot egyrészt tesztelés, másrészt konkrét építőipari tervezési feladatok végrehajtásához ismételten alkalmaztuk.

Az egyik volt a Sajó völgyének morfológiai térképe. A talajmechanikai függvény szerepét itt a tengerszint feletti magasság játszotta, a fúrásokat a térképről leolvasott magasságértékek, s az értelmezési tartományból kimaradó részt a Sajó medre helyettesítette, itt-ott szigetekkel, amelyek viszont beletartoztak az értelmezési tartományba. Mivel a függvényérték kiszámítása olcsó volt, sok állt rendelkezésre. A legkisebb négyzetek módszerével kiszámított rácshálózat alapján rajzoltatott néhány szintvonal összehasonlítható volt az eredeti térkép szintvonalaival, bár ez a függvény a talajmechanikai függvények tulajdonságának egyáltalán nem tett eleget.

Budapest VII. kerületében egyre súlyosabb nehézséget jelentett, hogy a megemelkedett talajvíz a lakóházak egész sorának pincéit elöntötte. Az elöntés okának felderítése alapfeltétel volt a megszüntetés, védekezés megtervezéséhez. A több évtizedes észlelési idősorral rendelkező környező állandó talajvízszint-észlelő kutakra támaszkodva kirajzoltattuk a környék talajvízszint térképét, melynek alapján a csőrepedésből származó vízhozzáfolyást lehetett felismerni.

Budapest lakásépítése központilag kiemelt feladat, mind a még felhasználható szabad területek felhasználásával, mind az elavult belső részek rekonstrukciójával. A közlekedés- és közműépítés szint-alatti munkái is jelentősen nőnek. Ugyanakkor a talajvíz jelentős nehézségeket és többletköltséget okoz a fővárosi építkezésekben, de a meglévő építmények állagromlásában is. A Fővárosi Tanács VB megkeresése alapján folyamatba tettük a talajvízjárás törvényszerűségeinek számítógépes értékelését. A délpesti alközpont kiépítését (Pestlőrinc és Kispest rekonstrukciója) készítendő komplex építésföldtani térképezéshez ugyancsak sorozatban kerül a program alkalmazásra.

Az említett példák is igazolják, hogy a céltudatos és körültekintő programfejlesztés révén egy komplex műszaki célinformációs programrendszer egyetlen tagja is az építőipari műszaki tervezés fejlesztésének hatékony elemévé válhat.

TERMELÉSTERVEZÉSI ÉS ERŐFORRÁSSZÜKSÉGLET SZÁMÍTÓ PROGRAMRENDSZER A DBOMP ÉS AZ LPS PROGRAMCSOMAGOK FELHASZNÁLÁSÁVAL

Vasas Sándorné dr.—Sarkadi Miklós
(ÉGSZI)

Ebben az előadásban egy alkotói kollektíva* többéves fejlesztési munkájáról számolunk be. Intézetünk az Építésügyi és Városfejlesztési Minisztériummal és a Beton- és Vasbetonipari ekkel együttműködési szerződést kötött a vállalat integrált irányítási rendszerének kialakí-. E munka keretében készült el:

1. A vállalat termelésirányítási alrendszerének adatbázisa
2. A vállalat termelésirányítási alrendszerének termeléstervezési modulja, az LPS program-csomag felhasználásával
3. A vállalat termelésirányítási alrendszerének szükségletszámítási rendszere

A kidolgozott modulok R—22-es típusú gépen, DOS operációs rendszer felügyelete alatt ódnek.

állalat termelésirányítási alrendszerének adatbázisa

A vállalat-irányítás alapfeltétele a pontos, naprakész műszaki-technológiai alapadatok is-te, s hogy ezek a szükséges formában, a szükséges helyen és időben rendelkezésre áll-.

Ennek megfelelően a korszerű és új szervezési, modellezői gyakorlat célja, hogy a műszaki ági adatok egy helyen, kis redundanciával kerüljenek tárolásra:központosított formában előre meghatározott igények esetén a kívánt összehasonlításban álljanak rendelkezésre. Ezáltal biztosítható, hogy a változásokat csak egyszer kelljen átvezetni, s hogy mindig a rakész állapot szerinti információk álljanak rendelkezésre.

A Beton- és Vasbetonipari Művek termelésirányítási adatbázisát, DBOMB adatbáziskezelő-szerrel alakítottuk ki; két törzs, s két láncfileból áll. A törzsfileok rekordjainak azonosí-kulcsai) 1 Ø byte-osak; első két jegye a típuskód (TK).

TÉTELTÖRZS - *file*-ban a következő típusú rekordok találhatók:

- késztermék (TK = 1Ø)
- félkésztermék, v. fázistermék (TK = 2Ø)
- gyártóeszköz (TK = 5Ø)
- termékcsoport (TK = 11) ITJ szintek
- termékcsoport (TK = 12)
- ITJ alcsoport (TK = 13)
- ITJ csoport (TK = 14)
- főkönyvi csoport (TK = 16)
- egyéb termékcsoport (TK = 19)

programrendszer kidolgozói:
Lóczky Béla témafelelős, Buzás Sándorné, Kovács Ervin, Kőfalvy Zsolt, Novák József, Sarkadi Miklós
ogramtervező, Szigili László, Vasas Sándorné dr.

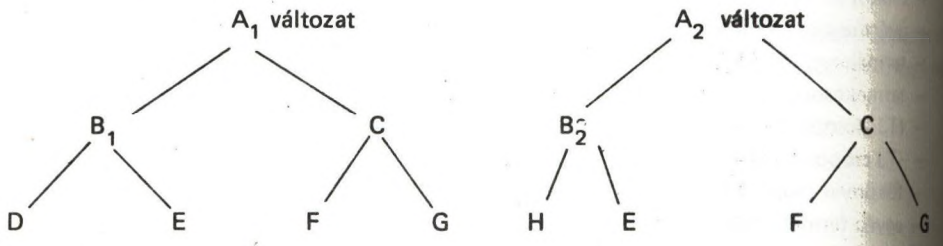
– félkésztermék csoport (TK = 21)
– anyagcsoport (TK = 31)
– gyártóeszközcsoport (TK = 51)

2. MUNKAHELY–MUNKAERŐ törzsfile (munkerő; TK = 4Ø és gyártósormunkahely; TK = 6Ø)

A rekordok tartalma: azonosító-kód megnevezés, mennyiségi egység, nettó és bruttó egységár stat. szorzó, össz. szorzó 1. csoportkód (főkönyvi kód), 2. csoportkód (ITY kód, vagy FEOR kód), stb.

3. A STRUKTÚRA láncfile a TÉTELtörzsfile rekordjainak strukturális kapcsolatait írja le, melyek egyrészt a gyártmányok és beépülő összetevői közötti összefüggéseket, másrészt a különféle csoportbatartozási információkat rögzítik.
Az egy struktúrarekordon belül megjelenő információ, hogy egy fölérendelt (pl. félkésztermék) elkészítéséhez az alárendeltből (pl. anyag) mennyi szükséges három különféle adattal (elméleti-, nettó-, bruttó-norma) kerülhet megadásra.

4. *Művelet láncfile* a TÉTEL törzsfile, s MUNKAHELY–MUNKAERŐ törzsfile közötti kapcsolatokat írja le. Egy rekordja egy fölérendelt tételtörzsbeli rekord (termék, félkésztermék) fajlagos megmunkálási igényét írja le egy adott alárendelt törzsbeli rekorddal leírt termelési keresztmetszeten.
A modellezett vállalat termelési sajátosságainak kezelése érdekében alakítottuk ki a *konvertálható termék,* s a *termékváltozat* fogalmát a késztermékek, s félkésztermékek vonatkozásában.
A BVM termelési profiljának megfelelően az ország területén levő hét gyárban termel, a nyolcadik gyár a termeléséhez szükséges sablonok előállításával foglalkozik.
Számos gyártmánycsoportot a vállalat több gyárában is gyártanak. Az ilyen gyártmányokat *konvertálható* termékeknek nevezzük és azonosítójukban a gyártókóddal különböztetjük meg, így az adatbázisban önálló termékként szerepelnek. (Ezt indokolja, hogy bizonyos adataik – pl. az ár, anyagfelhasználási jellemzői stb. – eltérő is lehetnek.)
Egy adott gyáron, üzemen belül ugyanazon terméket több fajta sablonban, vagy más típusú cementből is lehet gyártani. Ezeknek a *termékváltozatoknak* ugyanaz az azonosítójuk, neveik, áruk stb.) (tehát a törzsrekordjuk), azonban erőforrásigényük (tehát STRUKTÚRA, s MŰVELET, LÁNC kapcsolataik) részben különbözők, tehát előállítási költségeik sem feltétlenül egyformák. A változatok tárolását a STRUKTÚRA, s MŰVELET file-okban oldjuk meg, az alábbiak szerint:



A probléma általános megoldása a következő:
egy fölérendeltbe (F) beépülő alárendelteket (A) különböző tételsorszámmal látjuk el (ez lehet pl. szerelése, vagy műveleti sorrend), továbbá a láncrekordban szerepel egy fölérendelt változat (FV) és egy alárendelt változat mező (AV).

példában bemutatott két változat leírása ennek megfelelően a struktúra rekorddal:

F	FV	TS	A	AV	
A	Ø1	Ø1	B	Ø1	
A	Ø2	Ø1	B	Ø2	
A	Ø1	Ø2	C	Ø1	
A	Ø2	Ø2	C	Ø1	*
B	Ø1	Ø1	D	Ø1	
B	Ø2	Ø1	H	Ø1	
B	Ø1	Ø2	E	Ø1	
B	Ø2	Ø2	E	Ø1	*
C	Ø1	Ø1	F	Ø1	
C	Ø2	Ø2	G	Ø1	

x-el jelölt rekordokat nem kötelező megadni, mivel ha egy bizonyos tételsorszámon egy ott fölérendelt változathoz nincs rekord, akkor az azon a tételsorszámon szereplő alapvál-zatot (legkisebb sorszám) vesszük.

Ezzel a megoldással jelentős helymegtakarítás érhető el (hiszen csak az alapváltozásból ló eltéréseket kell megadni), ugyanakkor az adatbázist karbantartóktól, létrehozóktól nagyobb 53 gyelmet igényel.

A tételtörzsben minden változatnak megfeleltünk egy bitet, melynek állásából azonnal dönthető, hogy az adott változat létezik-e, vagy sem.

vállalati termelésirányítási alrendszer termeléstervezési modulja

A modul feladata az éves, negyedéves tervkészítés rendjéhez illeszkedően meghatározni vállalat gazdálkodási céljait szolgáló termelési (termékkibocsátási tervet) minden számba-tt termékcsoportra.

A vállalati éves, negyedéves termelési tervek meghatározását az alábbi tényezők együttes gyelembevételével lehet elvégezni:

Gyárthatő termékek
- leírásai,
- gyártandó mennyiségei (gyártási rendelések),
- gyártásra vonatkozó egyéb befolyásoló tényezők; pl. nyereségesség, fajlagos fedezetek.

Gyártáshoz felhasznált anyagok, sablonok, gyártási kereszmetszetek, munkakörök
- leírása,
- tervidőszaki kapacitása, ill.
- gyártásba vonásának egyéb feltételei.

Az 1., ill: 2. pontbeli tényezők közötti kapcsolatokat leíró információk
- termékek erőforrásszükségleti leírásai
- termékek műveleti normaleírásai.

Ezeknek az információknak a felhasználásával állíthatjuk össze a termelési terv meghatározá-nak tervidőszakra vonatkozó feltételeit, fogalmazhatjuk meg a célját; állíthatjuk össze az ezeket fejező lineáris egyenletrendszert.

A kialakítandó termeléstervezési modell alkalmazható a vállalat termékeinek teljes körére, gy kiemelt termékcsoportokra.

nagy része kiszűrhető, és már csak egy így „átvizsgált" feladatot engedünk át az LPS felé.

A termeléstervezési programcsomag futásának rendje
tehát a következő:

1. Tervezési segédletek előállítása (átlagtermék képzés, termékváltozatok összevont erőforrásigényének és célfüggvényegyütthatóinak meghatározása)
2. Megjegyzés: ezen programok csak szükség esetén futnak
2. Korlátok megadása (termékváltozatok és erőforrások körének meghatározása)
3. Ellentmondások vizsgálata
4. A feladat összeállítása az LPS igényeinek megfelelően
5. A feladat megoldása a kívánt célfüggvények szerint az LPS-el
6. A megoldások táblázása

Ezen utolsó fázisról kicsit részletesebben is szólunk: Mivel az LPS által készített eredménytábla számos okból nem megfelelő a vállalat számára, így az eredmény táblázására saját programokat készítettünk. Minden célfüggvény szerinti termékkibocsátási terv táblázásra kerülhet ITJ kód szerinti csoportosításban (művi és/vagy gyári szinten) és hasonlóan a főkönyvi kód szerint is. A tervtábla minden szintjén tartalmazza a terv szerinti termelés mennyiségét, értékét, nyereségességi jellemzőjét (fedezeti értéket), továbbá — ha ilyen adatok rendelkezésre állnak — összevethető különböző időszakok termelésével ill. rendelésállományával.

Ugyancsak minden célfüggvény szerint készülhet egy tábla, amely a tervezésbe vont termelési tényezőket (erőforrásokat) értékeli. (Az LPS által megadott úN. redukált ár nélkül itt többek között kiírásra.) Továbbá készül egy tábla, melyben a különböző célfüggvények szerinti tervek készülnek összehasonlításra.

A vállalat termelésirányítási alrendszer szükséglet számítási részrendszere

A termelés tervezéséhez (éves, féléves) és programozásához (dekád, havi, negyedéves) az erőforrásszükséglet kimunkálása tömegszerű és számításigényes munkát jelent. Ezt segíti elő az adatbázis használatára épült *erőforrás szükséglet számítási programcsomag,* melynek rendszeres használatával a termelésirányítás területén jelentősen növelhető a szervezettség, fokozható a naprakészség, javítható az információk pontossága és több segítség adható a döntések meghozatalához.

A tervezési vagy programozási termékmennyiségekre és a kijelölt időszakra a programcsomaggal az adatbázisban tárolt normatív adatok alapján meghatározható az anyag, gyártóeszköz, termelőkapacitás és létszámszükséglet vállalati, gyári, üzemi és telep szinten.

A programcsomag a szervezeti felépítés, a termék változatok és erőforrások különböző csoportosítási lehetőségeinek, valamint az adatbázisban tárolt adatok struktúrájának figyelembevételével 29 féle táblaválaszték gyakorlati felhasználását biztosítja, melyek közül egy-egy alkalmazó a feldolgozás folyamán az igényeinek kielégítését szolgáló táblákat kérheti.

1. *Erőforrásszükségleti táblák*

a) a végtermék, fázistermék, termékcsoportok, munkahelyek (gyártósorok) és üzemek erőforrásszükségletei erőforrástípusonként, s összesítve

2. *Erőforrácsoport-szükségleti táblák:*

Ezek szerkezete és tartalma megegyezik az előző csoportéval azzal az eltéréssel, hogy csak az erőforráscsoportok és összesített szükségleteik kerülnek kiíratásra.

Normatív termelési költségtáblák: melyek tartalmazzák előkalkulációs jelleggel a termelési adathoz szükséges normatív termelési költségeket különböző formában és csoportosításban.

A feladatot számítástechnikailag úgy oldottuk meg, hogy előállítjuk a termékváltozatok egységnyi mennyiségeinek fázisonkénti erőforrásszükségletét, melyet mágnesszalagon tárolunk. Minden egyes számítási időszakban ezek az egységnyi szükségletek „felszorzásra" kerülnek a számított mennyiségekkel és az így keletkező összeget különböző rendezettséggel táblázzuk:

Irodalomjegyzék:

Az IBM adatbankkezelő rendszerei
BME 1974.

LPS (IBM) programcsomag dokumentáció

A BVM termelésirányítási alrendszerének adatbázisa
DBOMP adatbáziskezelő rendszer segítségével
ÉGSZI tanulmány 1974.

A BVM termelésirányítási alrendszerének korszerűsítése
– rendszerterv
ÉGSZI tanulmány 1975.

ISMERETANYAG ALAPJÁN DOLGOZÓ INTERAKTIV ROBOTSZEM–RENDSZER

Vámos Tibor–Báthor Miklós–Mérő László
MTA SZTAKI

1. A rendszer általános filozófiája – intelligencia-szintek

A cél olyan rendszer létrehozása volt, amely gazdaságos módon képes különböző intelligens feladatokat megoldani, tehát olyan, amely a mini- és mikroszámítógépek eszköztárával dolgozik. A járható út hierarchikus: egy nagyon általános metodológia és egy nagy készlet különböző fajta eszköz arra, hogy a problémák széles osztálya legyen vele megoldható, továbbá egy egyszerűen kezelhető tervező rendszer, amely a meghatározott feladathoz szükséges eljárások szűk készletét összeállítja. Mindez ember-gép kapcsolatra alapozva.

A hierarchia legmagasabb szintje a kutatás-tervezésé, amely tartalmazza a szokásos rendszertervezési eszközöket és a 3. fejezetben ismertetendő interaktív grafikát.

A második szint a rendszertervezés. A cél egy meghatározott ipari munkahely kialakítása, tehát a szükséges hardware konfiguráció, a software kiválasztása és esetleg a célhoz szükséges bővítése, a rendszer szimulálása és felhasználásra alkalmas állapotban való átadása. A tervezést általában a kutatás-fejlesztési laboratóriumban lehet megkezdeni annak viszonylag bő eszköztárával.

A harmadik szint egy meghatározott munkafeladat beállítása és betanítása az adott munkahelyen, tehát a software még szűkebb részének választása, esetleg parametrizálása, kísérletezés. Ez a munkafeladat tipikusan üzemmérnöki jellegű.

Negyedik szint az operátoré. Feladata, hogy felügyelje a termelési programban meghatározott és előkészített munkafolyamatot (tehát egy szerelési feladatot, munkadarabok kivélogatását stb.).

A rendszert elsősorban szerelési és anyagmozgatási feladatokra szántuk. A későbbiekben ellenőrzési feladatokat is be kívánunk iktatni (textura – ebből felületi hibák, méretek, kontur-épség). A legfőbb korlátozások a következők:

– a tárgyak műszakiak legyenek, tehát jól meghatározott kontúrokkal vagy felületekkel rendelkezzenek;
– egy meghatározott feladatban szereplő tárgyak száma korlátozott (10–15);
– emberi beavatkozás mindig lehetséges legyen.

2. Képfeldolgozás

A képfeldolgozó berendezés hardware-jét a Budapesti Műszaki Egyetem Folyamatszabályozási Tanszékének munkatársai készítették el, ez egy 16 szürkeségi szintre bontó digitalizálót tartalmaz, amelynek szintjeit be lehet állítani és külön-külön maszkolni. A kép részekre bontható és az egyes részek nagyíthatók. A képfeldolgozónak saját memóriája van, így egy-egy képet pillanatképként tud rögzíteni és tárolni. Az előfeldolgozó algoritmus a képet átlapoló elemekre bontja és ezen belül alkalmazza azt az egyszerűsített templát módszert, amelyet Mérő–Vassy algoritmus néven ma már a nemzetközi irodalom többször is diszkutált, kiemelte egyszerűségét, sebességét és jó hatásfokát. Ez a helyi operátor két átlós egyenest használ templátként (mintaként), nem túlságosan zajos képek esetén kitűnő alapja a további képfeldolgozásnak (3. ábra). Az így kapott szálkaszerű, rövid, egyenes kontúrelemek illeszkedését a képben talált pontokhoz egy statisztikus becslő értékeli, ez súlyozó tényezőként fog szerepelni a későbbi feldolgozásban. Ennél az állásnál az algoritmus kétfelé

hat, attól függően, hogy milyen alkalmazásunk van. A szálkák maguk is jó kiinduló pont-
zoknak a minőségi összehasonlító módszereknek, melyeket durva módszereknek nevezünk
matékok, alakténvezők stb.). Ezek a durva összehasonlító módszerek a találgatást segítik
y nagyon gyorsíthatják az azonosítást a tárolt tárgymodellekkel. A szálkákból álló, eléggé
ös képet egy olyan algoritmus dolgozza föl, amely optimális utat keres föltételezett elága-
pontok között. Elégazási pontnak nevezzük azokat, amelyekben három vagy ennél több
nogén terület találkozik. Az elágazási pontot az algoritmus ott tételezi föl, ahol a szálkák
yszögének szórása helyi maximumot ér el.

Az elágazási pontok között utat kereső algoritmus a szálkákat hosszabb vonalszakaszok-
gyesíti. Ezek a csíkok. Néhány olyan matematikai feltételt állítunk, amelyek biztosítják,
y a csíkok a tárgyak kontúrvonalait egyértelműen és optimális módon kövessék (4. ábra).
algoritmus gondoskodik arról is, hogy a hibásan feltételezett elágazási pontokat javítsa
y kiiktassa.

A csíkok irányszögének, illetőleg irányszög-különbségeinek értékelésével alakítja ki az
oritmus a kontúrt képező egyeneseket és köríveket. A folyamat több lépcsőben tartalmaz
zonytalanságot, ezért több fajta interpretáció is lehetséges. Az algoritmus a főbb interpre-
ókat megtartja és azokhoz a korábban már említett szálka-súlyok alapján és az algorit-
ban számolt további bizonytalanságok figyelembevételével megbízhatósági súlyokat ren-
. Ezeknek az interpretációknak többek között a súlyok alapján is történő összehasonlítása
ismeretanyaggal a nyelvtani feldolgozás feladata lesz (4. pont).

3. Tárgy-modellezés — grafikus interakció

A tárgyak felismeréséhez helyzetüknek, irányuknak meghatározásához, a tárgyakkal való
nipulációhoz mind olyan geometriai ismeretanyagra van szükség, amelyet emberi beavatko-
l könnyen lehet kezelni (szimulálni, programozni, programot javítani). A geometriai ismer-
anyag display-megjelenítésének figyelembe kell vennie a számítógépes hardware korlátait,
ell végeznie a kétdimenziós nézet és háromdimenziós valóság közötti átalakításokat. A
dat némileg eltér a számítógépes grafika általános feladataitól, hiszen itt kisebb részlet-
ságra van szükség, néhány manipuláció más, viszont a munkának valós időben kell tör-
nie gyors, gazdaságos algoritmusokkal. A mi általunk alkalmazott eljárás különösen kitűnik
ejtett vonal és felületek gyors és egyszerű kezelésével, az interakció egyszerűségével. A rend-
r első- és másodrendű felületeket kezel, a testeket általában szabványidomokból teszi össze,
áthatásokat nem számol, mert ezekre a mi feladatainknál rendszerint nincs szükség, viszont
mítási igényük roppant nagy. A testek háromdimenziós, ún. drótvázas modelljének adat-
uktúrái nemcsak a geometriai adatokat tartalmazzák, hanem olyan kiegészítő információkat
amelyek a manipuláció számára később érdekesek lehetnek (pl. megfogási hely, a megfogás
nya, a szükséges erők stb.).

A tanítás kétféleképp is történhet: közvetlen a kamerával való ránézéssel és a megjele-
tt, feldolgozott kép kijelölésével, vagy pedig egyszerű modellépítő eszközökkel, melyek
bványelemeket hívnak, azokat forgatják, torzítják, paraméterekkel ellátva meghatározott
retűvé teszik. A rendszer mindezeket az eszközöket a kezelő rendelkezésére bocsátja, aki
yesen is alkalmazhatja őket.

A fölismerés és manipulálás céljaira nagyon alkalmas a hasonló rendszerekben már ko-
ban használt, ún. homogén koordináta módszer, amelyben mintegy négy dimenzióban je-
nítjük meg a háromdimenziós tárgyakat, és így a különböző nézetek, forgatások, egyéb li-
ris transzformációk egyszerű mátrix műveletekkel oldhatók meg. Így a display képernyő
természetesen a látvány-összehasonlító nyelvtani algoritmus a szabványos helyzetben ábrá-
t, háromdimenziós tárgydrótvázból megkapja a szükséges nézeteket, miközben a forgatások
án a rejtett vonal és rejtett felület algoritmus — a felületek irányítottságát és egymással

szembeni takartsági fokát kihasználva – számolja azt, hogy mi látható és mi nem az adott nézőpontból.

4. Felismerés

A felismerési eljárás feladata az, hogy megtalálja a legnagyobb valószínűségű azonosítást a háromdimenziós modellek és a kétdimenziós képek között, mely utóbbiak a szálkák, csíkok és kontúrvonalak interpretációiból születtek. A választott eljárás egy sajátos nyelvtani keresés.

Az előzők alapján tehát a rendszer ismeretanyaga azoknak a felületeknek az összessége, amelyek a háromdimenziós modellek különböző nézeteiből adódhatnak és természetesen ezek olyan kombinációi, amelyek értelmes képként szerepelhetnek a tudásanyag színterén, azaz az adott szerelési, anyagmozgatási feladatban.

A képleírások nyelvtana a szokásos elemhierarchiában épül föl, sajátossága az, hogy az egyes képelemek között szabadon lehet relációkat definiálni (ezek egyszerű képi relációk, mint pl. alatta, fölötte, mellette stb.), továbbá azok a becslési súlyok, amelyek a keresést vezérlik és gyorsítják. A nyelvtani keresési eljárást korábban Galló Válja dolgozta fel.

A felismerési folyamat eredménye egy olyan mátrix, amely leírja a tárgy transzformációját a modellkoordináta-rendszer és annak valódi helyzete között. A modell természetesen tartalmazza a tárgy valódi méreteit és a feladatot segíti, hogy a kameraállás koordinátái is ismeretesek legyenek. Ezek alapján egy, a legkisebb négyzetek módszerével dolgozó illesztő mátrixot lehet számolni a kétdimenziós nézetek élei és a modellélek között. Ez az illesztés azonosítja a megfelelő éleket.

5. Hardware és robotvezérlés

A vizuális bemenetet röviden leírtuk a 3. pontban. A számítógép jelenleg egy 40 kByte-os R–10, minidiszkkel, szalaggal és egy TEKTRONIX grafikus megjelenítővel. Tervünk az, hogy a következő időszakban át fogunk térni a GD'80-ra, amely egyben lehetővé teszi a PDP–11-es szabványokhoz való csatlakozást és az LSI–11-esekkel való vezérlést.

A manipulátornak két változata van, egy korábbi, ortogonális és egy most fejlesztett poláris rendszer, mind a kettő 6 szabadságfokú és egy fogó mozgással dolgozik; az erő- és nyomatékvisszacsatolás a Budapesti Műszaki Egyetem Folyamatszabályozási Tanszékén készül és hasonló ahhoz a megoldáshoz, amit a Draper Laboratórium alkalmaz, tehát egy erőmérőcellás hídrendszerből áll. A csuklóban való nyomaték-adónak nagyon nagy szerepe van a pontos illesztéseknél, tehát amikor a robotkéznek pontosan kell beillesztenie egyik tárgyat a másikba, ez a művelet megfelelő pontossággal szemmel nem vezérelhető, az ember itt rendszerint tapintási érzékszerveit használja. A számítógép és a robot között egy háromdimenziós CNC rendszer működik, amely az Intézetben kifejlesztett DIALOG rendszer egy változata. A robotvezérlés fontosabb jellemzői:

– a grafikus display segítségével irányítható és szimulálható;
– emberi működtetéssel és grafikus szimulálással tanítható;
– ugyanazokat a térbeli transzformációs algoritmusokat használja, mint a modellépítés és a felismerés, tehát a tárgy helyzetéből, a kép mozgásából és a vizuális felismerésből adódó hibák gyorsan számíthatók.

6. Kísérleti időadatok

Az egész felismerési folyamat körülbelül 30–50 másodpercig tart a 2. ábra tárgyának esetében. Egyes részidők: a rejtett vonal algoritmus különösen gyors, körülbelül 2 másodpercet vesz igénybe. A kontúrkereső algoritmus körülbelül 4 másodpercig dolgozik TV-képenként.

szálkákból csíkokat összeállító algoritmus 10–15 másodperces működésű. A nyelvtani felismerés, tehát a lehetséges felületeknek, nézeteknek a megtalálása 5–8 másodperc. Más, még szükséges on-line algoritmusok összesen kb. 1–2 másodpercet vesznek igénybe.

Mindezek az algoritmusok a ma már viszonylag lassú R–10-en voltak mérve. Ezek az adatok lehetővé teszik számunkra, hogy egyes lépések gyorsításával, párhuzamosításával, speciális hardware építéssel lényegesen tovább csökkentsük az időszükségletet azokon a pontokon, amelyek kritikusak.

7. Alkalmazások

Mint azt a bevezetőben is mondottuk, a rendszer elsősorban szerelési munkákra készült. Ez a feladat jó gyakorlásnak tűnik az egész problémára vonatkozólag. Egyelőre laboratóriumi kísérletekig jutottunk el. Az ipari megvalósítást a következő 3–5 évre tervezzük. Előkészítünk a közeljövőben egy olyan kísérletet, amelynél az IKARUS Gyár lemezfestő robotja számára ismerjük fel a konvejor szalagon beérkező lemezeket. Erre az automatára azért van szükség, mert a festőműhely közelében is egészségtelen a környezet. Foglalkozunk egy neuro-biológiai alkalmazással, amelynél neuronhálókat kell mikroszkópmetszeteken detektálni ember-gép kapcsolat keretében.

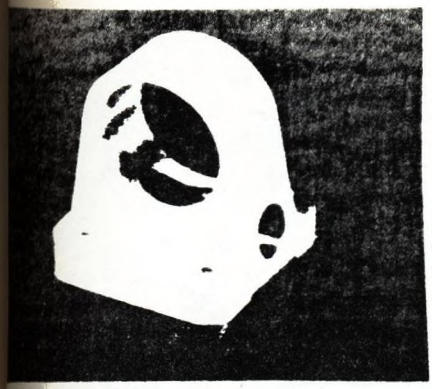
8. További irányok

Jelenleg azt tartjuk legfontosabb feladatunknak, hogy az elkészült laboratóriumi berendezésen elegendő kísérleti anyagunk gyűljön össze ahhoz, hogy a későbbi időszakban jó ipari berendezéseket és rendszereket tudjunk készíteni. Egyik érdekes kísérleti munkánk a megvilágításra vonatkozik: a megvilágítási módszerek, fényforrások optimális kiválasztása ahhoz, hogy az árnyék- és tükrözéshatások minimálisak legyenek, a zajt minél jobban csökkenteni tudjuk és a fontos részleteket a világítás segítségével is kiemeljük. A megvilágításnak és a vizuális felismerésnek egységes intelligens rendszerbe kell a későbbiekben illeszkednie. Hardware terveinkről már szóltunk. Lényegében véve elosztott funkciójú rendszerek felé törekszünk, felhasználva azokat az eredményeket, amelyeket az Intézetben részben a számítógépes grafikában, részben a numerikus vezérlésben is más csoportok elérnek.

A rendszer intelligenciájának fokozásában egyik legizgalmasabb kérdésünk a hasonlóságok és távolságok mértékeinek meghatározása. A távolság a statisztikus alakfelismerésben jól ismert és meghatározott fogalom. A metrikát a megközelítés statisztikus jellege határozza meg, az a metrika, amely a vizsgált cluster-eket a legmegbízhatóbban választja szét. A mi esetünkben a metrika inkább logikai, mint statisztikus jellegű és szorosan össze van kötve a tárgy és a feladat szemantikájával. Mit tekintünk hasonlónak, mely részleteket tételezünk fel azonosnak, ez a feladattól, helyzettől függő. Abban az irányban haladunk, hogy az egész rendszer alapkoncepciójának megfelelően az emberi kezelő számára adunk megfelelő interaktív nyelvi és grafikus eszközöket ahhoz, hogy az adott feladat optimális távolságait meghatározza.

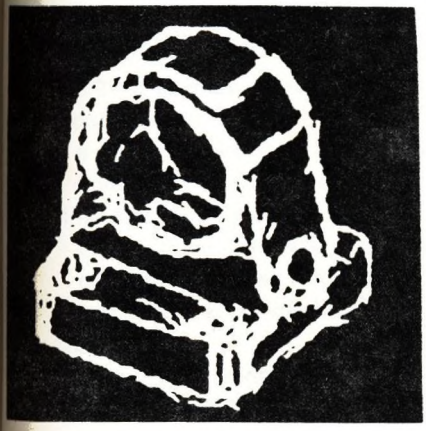
Irodalomjegyzék

1. A.P. Ambler, H.G. Barrow, R.M. Burstall, R.J. Poppelstone: A Versatile Computer-Controlled Assembly System. Proc. 3rd IJCAI (Stanford), 1973, pp. 298–307.
2. M. Báthor: Interactive Picture Manipulation. 2nd Hungarian Computer Science Conference, Budapest, 1977, pp. 168–177.
3. M. Báthor: Hidden-Line Algorithm for a Robot Experiment. Ph.D. Thesis, Budapest, 1977. (in Hungarian)
4. B.G.Baumgart: Geometric Modelling for Computer Vision. Stanford Memo AIM–249, 1974.
5. K.S.Fu: Stochastic Tree Languages and Their Applications to Picture Processing. International Symposium on Multivariable Analysis, Pittsburgh, 1978.
6. V.Galló: A Program for Grammatical Pattern Recognition. 4th IJCAI (Tbilisi), 1975, pp. 628–634.
7. V. Galló: Sistema dlya obrobotki spiskov dlya intelligentnovo robota. 2nd Hungarian Computer Science Conference, (Budapest) 1977, pp. 400–411 (in Russian)
8. P.P. Loutrel: A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra. IEEE Trans. Comp., C–19, 1970, pp. 105–213.
9. L.Mérő: A Ouasi–Parallel Contour Following Algorithm. Proc. AISBIGI Conf. on AI, Hamburg, 1978.
10. L. Mérő, T. Vámos: Real-Time Edge Detection Using Local Operators. 3rd IJCPR (Coronado) 1976. pp. 31–36.
11. L.Mérő, Z.Vassy: A Simplified and Fast Version of the Hueckel Operator. 4th IJCAI (Tbilisi), 1975, pp. 650–655.
12. J.L.Nevins et al.: Exploratory Research in Industrial Modular Assembly, Cambridge, Mass., 1977.
13. M.Potmesil: An Implementation of the Loutrel Hidden-Line Algorithm. Rensselaer Polytechn. Inst., TR CRL–49, Troy, N.Y., 1976.
14. L.G.Roberts: Machine Perception of 3D Solids. MIT Press, Cambridge, Mass. 1965. pp. 159–197.
15. A Rosenfeld, R.A.Hummel, S.W.Zucker: Scene Labelling by Relaxation Operations. IEEE Trans. SMC–6. 1976. pp. 420–433.
16. Y.Shirai: Analyzing Intensity Arrays Using Knowledge about Scenes, in the Psychology of Computer Vision. (Ed. P.H. Winston), New York, 1975.
17. A.Siegler: Computer Controlled Object Manipulation. 2nd Hungarian Computer Science Conference, Budapest, 1977. pp. 724–738.
18. E.Tanaka, K.S.Fu: Error-Correcting Parsers for Formal Languages. IEEE Trans. Comp. C–27, 1978. pp. 605–616.
19. T.Vámos: Industrial Objects and Machine Parts Recognition, in Applications of Syntactic Pattern Recognition. (Ed. K.S.Fu), Heidelberg, 1977. pp. 243–267.
20. T.Vámos: CAD–Marriage with AI Methods, Views Based on Dual Experiments. Prepr. of IFIP WG 5. 2 Conf. on AI and PR in CAD, (Grenoble), 1978. Session 5.
21. T.Vámos: Automatic Control and Artificial Intelligence (invited survey). Prepr. of 7th World Congress (Helsinki), 1978. 4. pp. 2355–2369.
22. T.Vámos, Z.Vassy: Industrial Pattern Recognition Experiment – A Syntax Aided Approach. Proc. Ist IJCPR, (Washington), 1973, pp. 445–452.
23. T.Vámos, Z.Vassy: The Budapest Robot – Pragmatic Intelligence. Proc. of 6th IFAC World Congress (Boston), 1975, Part IV/D, 63.1.
24. IDOS Manuals (PR, OP, MO/Ø3, PLM); Budapest, 1978.
25. Hajnal M.,Loványi I.,Mérő I.,Siegler A.,Vajta L.: Intelligens szem-kéz rendszerben alkalmazott képfeldolgozó berendezés. Mérés és Automatika, 7/1978. pp. 255–259.

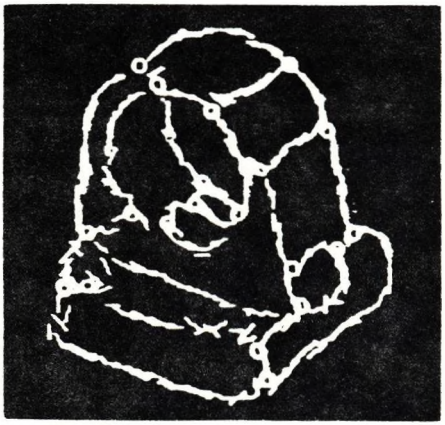


ábra: A tárgy képe

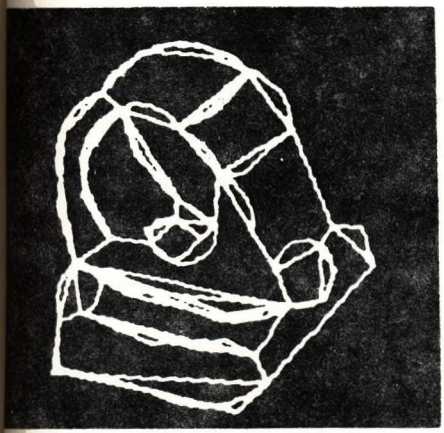
2. ábra: Digitalizált negatív kép



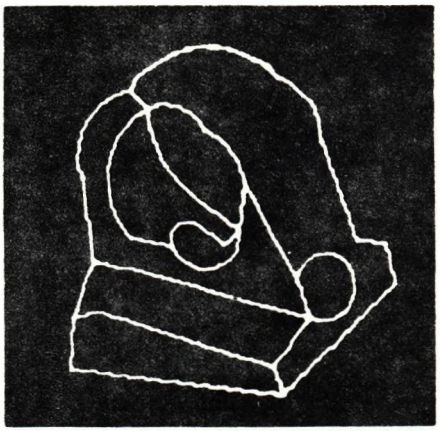
ábra: Szálka-kép



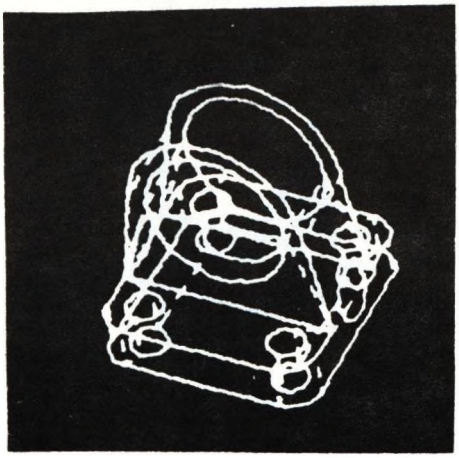
4. ábra: A szálkákból kapott csíkok az elágazási pontokkal



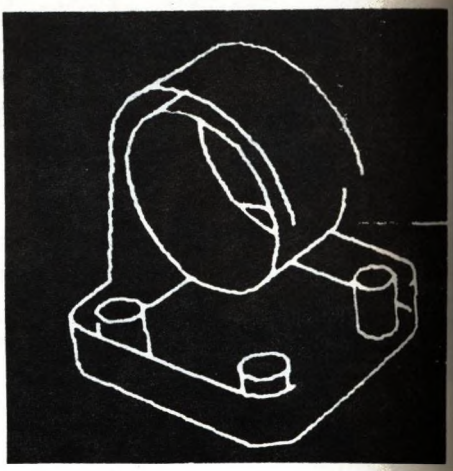
ábra: A csíkok minden lehetséges értelmezése



6. ábra: Körvonalrajz a csíkok alapján. A felismerés mintája



7. ábra: 3 dimenziós drótvázmodell



8. ábra: A modell rejtett vonalú változata

AZ R11 SZÁMÍTÓGÉP HARDWARE ÉS SOFTWARE JELLEMZŐI

Vinkovits László – Trencsényi István
VIDEOTON Fejlesztési Intézet

Az R11 nagy teljesítményű számítógép. 1 Mbyte-ig bővíthető memóriakapacitása, valamint ális hardware és software szervezése révén több, egymástól független feladatot szolgálhat gyidejűleg.
A központi egység hardware védelmi rendszere kizárja, hogy az egyes programok meg- ik egymást.

Alkalmazási területei és üzemmódja.

- Hagyományos számítóközponti tevékenységek, adatfeldolgozás műszaki számítások
  Batch rendszer
    job-ok egymás utáni futtatása
  Spool rendszer
    a lassú perifériákat diszken lévő file-ok helyettesítik. Ezáltal a lassú perifériák működése, az adatok feldolgozása és az eredmények listázása lényegesen hatékonyabb.
  Time sharing alkalmazások
    több konzol interaktív kapcsolata
    oktatói rendszerek futtatása
    interaktív programfejlesztés
- Adatátvitel
    számítógép hálózatok R10 M – R11
    Remote Job Terminál emuláció
    ESZR sor többi gépének elérése
    X 25 postai hálózat elérése
    Hierarchikus hálózatok:
    R11 – R10 M Remote Process Terminál elemekből
- Real-time feldolgozások
    Tranzakciókezelés – adatbáziskezelés
    Ipari folyamatirányítás

Multifunkciós üzemmódban a felsorolt alkalmazásokat részfunkcióként a gép egyidejűleg a látni, miközben betöltheti az R11 – R10 M modellekből felépített hierarchikus hálóza- legmagasabb szintű tagjának szerepét.
Az R11 központi egysége két nyomtatott áramköri kártyán realizált, 16 bites mikroprog- ozott processzor.
mzői:
- 32 (vagy 64) IT szint
- 155 utasítás
- 16 regiszter
- a központi egység, a tár és a perifériacsatolók ugyanarra a MONOBUS-ra csatlakoznak (Átviteli sebessége: 2,3 Mbyte)s.
- mikroprocesszoros perifériacsatolók (max 14 db)

– 1 Mbyte-ig bővíthető operatív tár
– automatikus mikrodiagnosztika
– távtöltési és távdiagnosztikai lehetőségek

Az achitektura sajátosságai:

– a memória szegmentáltsága
– multifunkciós szervezés
– közös újraindítható szubrutinok
– az egyes taskok saját stack-kal rendelkeznek
– a várakozási sarok kezelését a hardware végzi

Az R11 periféria készlete alapvetően megegyezik az R10 M-nél alkalmazott egységekkel, ezek kiegészülnek még nagykapacitású mágneslemez tárolókkal és nagy teljesítményű adatátviteli vonalakkal.

Az R11 alapsoftware kétszintű:

Az R11 1. szintű software rendszere a bevált, sokszáz rendszeren sikerrel alkalmazott R10 software komponenseken alapul, kiegészítve új, jobbhatásfokú szolgáltatásokkal, mind az operációs rendszerek, mind a programozási nyelvek és alkalmazói rendszerek területén.

Az R11 2. szintű software az architektúra sajátos vonásait kihasználó MTM 2 multifunkciós multitask monitorra épül. Az MTM 2 több felhasználói funkciót képes vezérelni. Ezek mindegyike saját óperációs területtel rendelkezik és ugyanazokat a központi monitor szolgáltatásokat használja.

A programozást a következő magasszintű nyelvek támogatják:

– MAS makroassembler
– FORTRAN IV
– RTL
– COBOL
– BASIC
– MAG mikrogenerátor

A programkészítés eszközei:

– EDI forráskönyvtár kezelő
– LIB diszkes könyvtár kezelő
– LED 2 szerkesztőprogram
– GRG csoportgeneráló
– Rendszergeneráló
– FMS 2 file kezelő rendszer
– Különféle utility programok

Alrendszerek:

– Batch Job Control alrendszerek
– DEBUG program belövő alrendszer
– TSE time sharing alrendszer
– TCS terminálkezelő alrendszer
– TTMS szinkron adatátviteli eljárások kezelése
– DMS 600 nagy adatbáziskezelő és tranzakciófeldolgozó rendszer.

A rendszer szolgáltatásait egészítik még ki a különféle felhasználói könyvtárak és programcsomagok, melyek közül megemlítjük a fix és lebegőpontos, egyszeres és duplapontosságú matematikai programkönyvtárat, a decimális műveletvégzés könyvtárat, a tudományos és statisztikai számítások FORTRAN szubrutinok könyvtárat.

A monitor alapelve

A multifunkciós (multitask monitor a felhasználónak olyan központi szolgáltatásokat biz-
ít, amelyeket valamennyi task rendelkezésére lehet bocsájtani. Ezek a szolgáltatások segítik
felhasználót problémáinak megoldásában.
monitor által nyújtott főbb szolgáltatások a következők:

- programok kezelése
- könyvtárak kezelése
- fizikai input-output műveletek kezelése,
- task kezelés,
- taskok csoportjainak (funkcióknak) a kezelése,
- dinamikus memória kezelés,
- események kezelése,
- üzenetek kezelése,
- idő és időzítések kezelése,
- eltérülések kezelése,
- megszakítások kezelése,
- operátori párbeszéd vezérlése,
- file katalógus kezelés
- a monitor dinamikus inicializálása

Minden szolgáltatás supervisor primitivek halmazából áll. Ezen primitívekhez makrók
útján férhetünk hozzá (makróassembler). A supervisor modulok CSV utasítással történő köz-
etlen hívása továbbra is lehetséges, de nem ajánlatos mivel a makrók használata egyrészt sok-
al flexibilisebb, másrészt a különböző monitorok közötti kompatibilitást egyszerűbb biztosí-
ani ezen a szinten. Bizonyos supervison primitivek csak privilegizálási üzemmódból érhetők el.

A következő néhány fogalom értelmezése segítséget nyújt a monitor alapelvének megér-
téséhez.

Logikai gép (process)

Az R11 feldolgozó egysége olyan fizikai gép, amely egyszerre csak egyetlen utasítást hajt
végre. A regiszterek elmentési és helyreállítási mechanizmusa azonban lehetővé teszi, hogy ezt
olyan n darab logikai gépből álló halmaznak tekintsük, amelyek közül egy adott pillanatban
csak egy aktív.

Taskok és programok

Egy program egy utasítás sorozatból áll. Egy task egy végrehajtó programnak egy végre-
hajtási prioritási szintnek és ennek végrehajtásával megbízott logikai gépnek az összerendezéséből áll.

Két programtípust különböztetünk meg:
nem reentrant program; ez esetben a memóriába a programnak annyi képét kell betölteni,
mint ahány task hívja.
Reetrant program; ez esetben csak egyetlen egyszer kell betölteni a program képét a memóri-
ába függetlenül a hívó taskok számától.

Egy task kontextje

Minden task állapota egy kontext-tel jellemezhető, melyet a központi memóriába helye-
zünk el és ahová a task cseréjekor mentjük a task újraindításához szükséges információkat.

Kontext csere

A kontext csere olyan művelet, amely egyrészt a futó task újraindításához szükséges in-
formációkat elmenti a futó task kontext-jébe, másrészt újra inicializáljuk a gyorsregisztereket
a megszakító task kontextjének tartalmával.

Ezt a műveletet automatikusan hajtja végre egy mikroprogramozott ütemező azonnali taskok esetén, míg elhalasztott taskok esetén a monitor programozott ütemezője hajtja végre.

Azonnali task

Azok a taskok amelyek az IT szinthez rendeltek.

Elhalasztott taskok

Azok a taskok, amelyek nincsenek IT szinthez rendelve.

Szegmentálás

A látszólag szimultán feldolgozott taskok szeparálásának szükségessége a szegmentáláson alapuló védelemhez vezetett. A programokat és (vagy adatokat változó hosszúságú) (max. 32 Kszó) és áthelyezhető memória particiók tartalmazzák. Ezeket szegmenseknek nevezzük.

Szegmens

Egy szegmens egy bázissal és hosszal definiált folyamatos memóriaterület.

Szegmens leíró

Minden egyes szegmenst két szó ír le:

- bázis cím,
- hossz.

A hardware kezeli a bázisokat és hosszakat s így lehetővé válik a szegmensek áthelyezhetősége.

Szegmens címzés

Egy szegmens nem érhető el a felhasználói üzemmódban, csak ha leíróját előzőleg privilégizált módban betöltöttek a megfelelő regiszterekbe.

Funkció (task csoport)

Egy funkció olyan taskok együtteséből áll, melyek bizonyos számú közös szegmensen osztoznak, amelyek a csoport környezetét képezik (virtuális gép). Minden csoporthoz egy szegmens leíró tábla van hozzárendelve, amelyhez a felhasználó nem férhet hozzá. Ezen szegmens leíró táblára mutató pointer minden task kontextjében megtalálható.

Szegmens típusok

- program szegmens,
- stack szegmens,
- megosztott szegmens
- adat szegmens

A taskok közötti szinkronizálás és kommunikáció

Kommunikáció ugyanazon csoport taskjai között

Ugyanazon csoporton belül a kommunikáció történhet a megosztott adatszegmens segítségével vagy a monitor segítségével küldött üzenetekkel.

Kommunikáció csoportok között

Két lehetőség adott: — kommunikáció közös file-okon keresztül
— kommunikáció üzenetekkel (monitor szolgáltatás)

Szinkronizáció ugyanazon csoport taskjai között

A taskok szinkronizálása a csoporton belül definiált eseményekkel és szemaforokkal történhet.

Kompatibilitás

Az R11 monitora kompatibilis az R10/R12 multitask monitorával. A software lehetőséget nyújt R10/R12 funkció definiálására. Ez azt jelenti, hogy az R10/R12 multitask monitorának standard funkció szimuláltak az R11-en.

MŰSZAKI FELÜLETILLESZTÉSI FELADATOK MEGJELENÍTÉSI PROBLÉMÁI ÉS A MEGVALÓSÍTÁSHOZ SZÜKSÉGES KÉT ALGORITMUS BEMUTATÁSA

Wildner Dénes
ÉGSZI

Bevezetés

A dolgozat első részében röviden ismertetjük egy ezévben elkészült építőmérnöki szintvonalrajzoló program főbb kialakítási szempontjait illetve a megvalósított feladat legfontosabb jellemzőit. A második és harmadik részben a program számára kidolgozott két algoritmust ismertatunk be. Az első módszer megadott alappontokra mint sarokpontokra háromszögelemhálózatot generál, miközben figyelembe veszi a ponthalmaz nem konvexségéből, illetve folytonossághiányaiból eredő előírásokat. A második bemutatásra kerülő eljárás egy speciális B-Spline függvényt állít elő szintvonalgörbe rajzoláshoz, szintén figyelembevéve a mérnök-felhasználói gyakorlat szabta feltételeket.

A feladattal szemben támasztott igények rövid ismertetése

Az építőmérnöki számítástechnikai feladatok nagy hányadát még ma is olyan számítások adnak ki, melyek valamilyen síkbeli szerkezet (födém- és alaplemezek ill. faltartók) alakváltozásainak, igénybevételeinek a szerkezet diszkrét pontjaiban számított értékeit eredményezik – általában valamilyen mozaikelemes módszer révén. Az eredmények jól kezelhetősége érdekében sürgető igény mutatkozott egy olyan plotter-program készítésére, mely valamilyen, jól felhasználható módon megjeleníti a diszkrét pontjaival jellemzett elmozdulási, igénybevételi, stb... felületeket, s ki tudja elégíteni e speciális felhasználói területnek a megjelenítéssel szemben támasztott igényeit. A szükséges előkészítő munka elvégezte után e feladat megvalósítására egy olyan szintvonalrajzoló programtervet készítettünk, melynél az alábbi legfontosabb szempontokat vettük figyelembe:

– Az alappontokból álló ponthalmaz általában egy vagy több, jól meghatározott peremvonallal van körülhatárolva.

– A peremvonallal körülzárt halmaz nem szükségszerűen konvex halmaz: bevágásokkal, törésekkel rendelkezhet.

– Esetenként a ponthalmaznak csak egy kivágott részletére kíváncsi a felhasználó mérnök, ekkor a megvizsgálandó ponthalmaz nem azonos a teljes ponthalmazzal.

– A pontokra illeszthető felület adott egyenesek mentén esetenként feltűnő törésekkel illetve nyíródásokkal kell, hogy rendelkezzen (pl. élteher alatti nyomaték- ill. nyíróerő függvény).

– A megjelenítés a törtvonalaknál jóval vizuálisabb folytonosgörbe szintvonalakkal történjen, melyek pontossága azonban nem kell, hogy meghaladja sem a megelőző számítás, sem a felhasználó-mérnök pontosságigényét. Különösen akkor nem, ha az jelentős számítási idő növekedést okoz.

– Sok esetben a megelőző számítások melléktermékként a ponthalmazhoz valamilyen végeselemes hálózat az indításkor már rendelkezésre áll.

A módszer kiválasztásánál az első két és az utolsó jellemző meghatározó szerepet játszott. Nem konvex halmaz nem fedhető le egykönnyen egy összefüggő folytonos felülettel. Ezért úgy döntöttünk, hogy első lépésben a pontseregre egy generált háromszög-elemekből álló mozaikfelületet illesztünk, majd az egyes szintsíkok és a mozaikfelület metszéséből adódó nyílt vagy zárt sokszögvonalra kellő símaságú, a sokszögvonal töréspontjaiban kollokáló B–Spline típusú görbéket illesztünk, melyek egymásba-metsződés lehetőségét a kellő pontsűrűségű sokszögvonal felvétellel szürjük ki.

A kidolgozott módszer sajátossága egyrészt egy olyan háromszög-elem generáló program, mely — ellentétben az irodalmakban közöltekkel — nem törekszik kellő kövérségű, nagyobb pontosságot eredményező elemek fölvételére, hanem az összes alappont felhasználása az elsődleges célja. Fontos szempont az is, hogy a kitüntetett vonalak (perem, törésvonal, stb...) mindig elemhatárra essenek. A program másik sajátossága a szintvonalrajzolásnál alkalmazott speciális, $\underline{p}(t) = \underline{a}_o + \underline{a}_1 t + \underline{a}_{k-1} t^{k-1} + \underline{a}_k t^k$ alakú B–Spline függvény, ahol a feladattól függően k értéke más és más lehet (k=3 esetben a jólismert kubikus B–Spline).

Jelen dolgozatunkban, anélkül, hogy teljességre törekednénk, csupán e két, fent említett eljárás algoritmusát fogjuk bemutatni, de megjegyzésül közöljük, a teljes program 1979 közepére készül el az ÉGSZI Mérnöki Számítások Osztályán, háromféle (ekvidisztáns, százalékos és Renard sor szerint sorbafejtett) szintvonalosztás-típussal, kitüntetett csomópontok, kontúr-vonalak, csomóponti feliratok, koordináták, mozaikháló, stb ... feltüntetésének lehetőségével. A program bejátszása az intézet SIEMENS 4004/151G típusú számítógépen, BS 2000-es dialóg operációs rendszerben készült. A rajzokat az itteni CalComp 925/936 típusú rajzgépen készíthetjük el.

A háromszög-elem hálózat generáló eljárás bemutatása

Mindazokban az esetekben, amikor nem rendelkezünk használható mozaikelem-hálózattal ennek az eljárásnak a használata szükséges ahhoz, hogy szintvonal görbéket készíthessünk. Mivel a háromszög-elem felvétel kellő sűrűségű vonalhálózatot szolgáltat, a szintsíknak ezen élek menti metszéspont-koordinátái könnyedén számíthatók és a mérnöki pontosság-igényt általában kielégítik, e síkelemű mozaikhálózat fölvétele mellett döntöttünk. Azzal a kiegészítéssel azonban, hogy esetleges nagyobb pontossági igények esetén a szomszédos elemek sarokpontjainak figyelembevételével másod- és harmadfokú interpolációk is alkalmazhatók legyenek, esetlegesen olyan általánosított inverz-felírásmóddal, melyben az egyes csomópontok figyelembevételénél más és más súlyt alkalmazzunk, attól függően, hogy az alapszakaszon, vagy azon kívül fekvő pont az illető.

A háromszögelem-generáló algoritmus alapelve egy olyan sokszögvonal-spirál készítése, mely vagy a külső peremtől, a koordinátarendszernek megfelelő primitív körbenjárás szerint, vagy egy belső peremről ennek ellentettje szerint indulva folyamatosan körbe-körbe járja a kijelölt ponthalmazt és minden egyes alapszakaszhoz egy harmadik sarokpontot választ ki a lehetséges alappontok közül. Azokat a csomópontokat, melyeket az eljárás más körbezárt elemekkel, letiltja a további kiválasztás elől. A folyamat mindaddig tart, amíg az összes alappont letiltódik, a spirál egy pontban zárul avagy elemgenerálás nélküli üres kört végez.

A háromszög harmadik sarokpontját a rendelkezésre álló pontok közül aszerint választja ki, hogy a három pont köré írható körben ne legyen további kiválasztható pont, és a

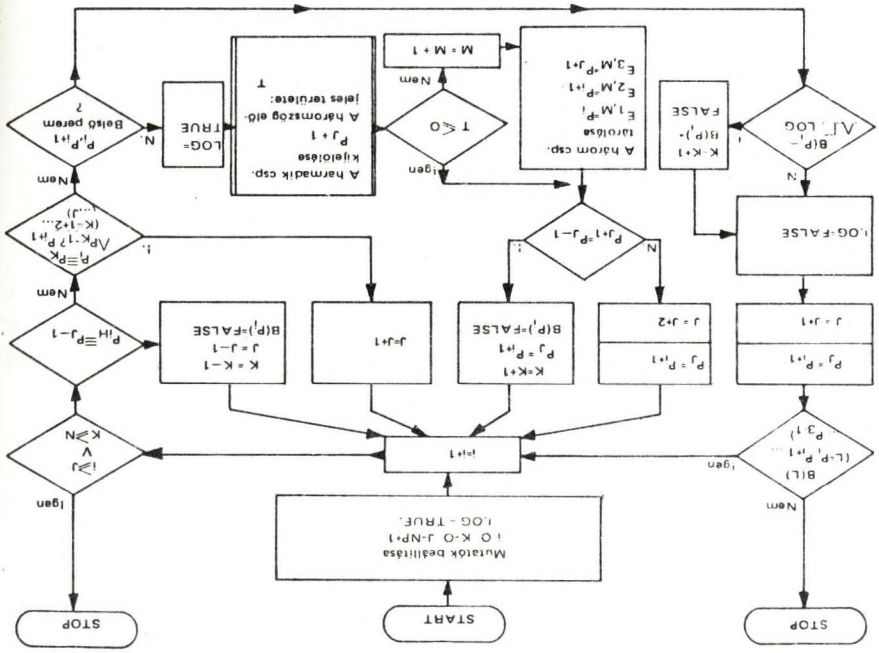
$$T = \frac{1}{2} \sum_{i,j,k} (x_k - x_i) y_j$$

módon számított előjeles terület ne legyen negatív. Ha mégis találkozunk olyan ponttal, mely

említett körön belül fekszik, ezen új pont egy új kört definiál, s a kiválasztási ciklus folytódik. A számítási idő csökkentése érdekében a vizsgálat valójában a három pont köré írható kör sugár-négyzeteinek összehasonlításával történik. A gépi nulla területű elemek szintén nem tárolódnak, de az elemgeneráló sokszögvonal úgy tekinti, mintha valóságos elemet generált volna (pl. I/2 ábra 1–2–3 elem). Igy a halmaz peremének leírásához nem szükséges minden perem-pontot megadni. Elegendő csupán azon perempontok leírása, melyek törést okoznak a peremvonalban. Az I/2 ábrán egy „I" jelöli a P vektor kezdeti-értékként feltüntetett 2 pontját, mely nem tartalmazza a szintén peremen fekvő 2, 27, 29 jelű csomópontokat.

A belső peremek (pl. lyukak, repedések) töréspontjai a külső peremmel (azaz a felvett koordinátarendszer forgatóértelmével) ellentétes körbenjárási sorrendben írandók fel.

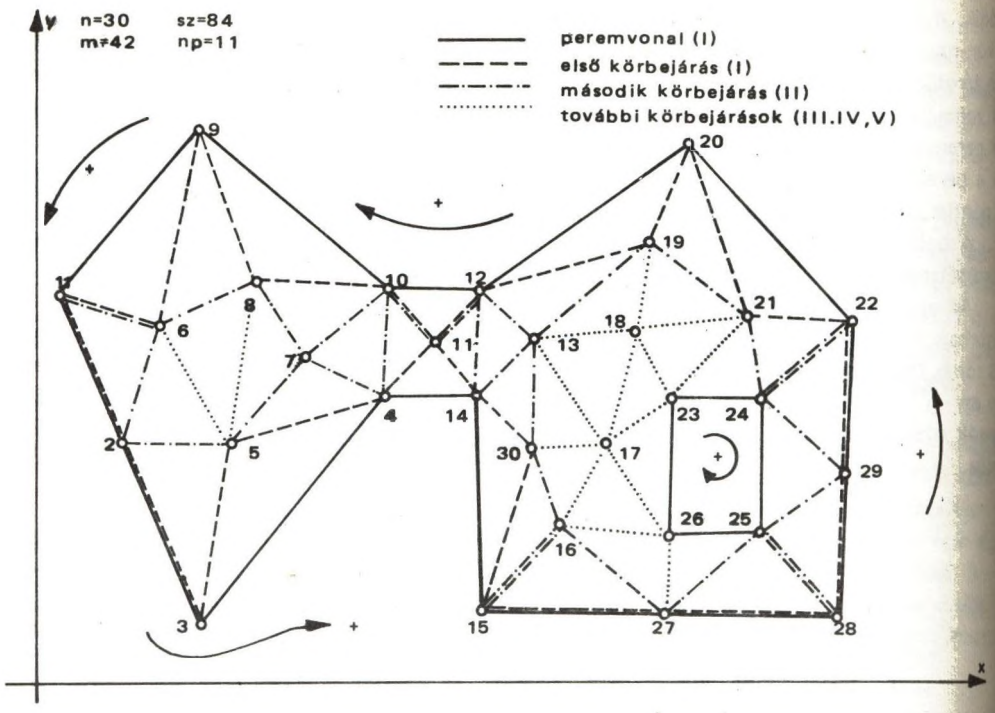
Az elemgenerálás mentén az I/1 ábrán lévő folyamatábrával szemléltetjük. Egy P jelű vektorban fokozatosan tárolódnak a sokszögvonal-spirál töréspontjai mindaddig, amíg az E jelű tömb fel nem töltődik a generált háromszögelemek sarokpont-hármasaival. A P vektorban egy „i" és egy „j" index jelzi folyamatosan az aktuális régi alapszakasz ill. az újonnan generált oldalszakaszok kezdőcímét, míg „k" változó a már körbezárt csomópontok számát tárolja.



I/1. ábra
A háromszög-elem generáló eljárás folyamatábrája

A további jelölések magyarázata:

N:	Az alappontok száma (bemenő érték)
NP:	A külső perempontok száma (bemenő érték)
B (N):	Logikai vektor. Bemenő értéke N PB. True. Kimenő értéke K DB. False
P (5000):	Egész típusú vektor az elemgeneráló sokszög-spirál tárolására. Indításkor az első NP+1 eleme a külső peremet írja le (ld. I/2. ábra)
M:	A generált elemek száma (kimenő érték)
E (3,M):	A generált háromszög-sarokpontok (kimenő érték)
K:	A körbezárt csomópontok száma (kimenő érték, általában: N)



P VEKTOR ELEMEI:

												I.								
1	2	3	14	15	28	22	20	12	10	9	1	2	3	5	4	11	14	30	15	27
											II.									
28	29	22	21	20	19	12	11	10	8	9	6	(1)	5	[3]	4	10	11	12	14	13
												2		7		III.				
30	16	(15)	25	(28)	24	(22)	19	(20)	12	11	10	7	8	6	[9]	2	7	10	[4]	12
		27		29		21		13							5	8	IV.		11	
13	[14]	(30)	◇26	(27)	◇24	[29]	21	18	(19)	12	11	10	7	[8]	[5]	[6]	[21]	(17)	◇23	
	17	16		◇25		23			13	18	17	26	[16]	24	23	(18)	[13]	24		
													25			26	25			

JELMAGYARÁZAT: ZÁRÓDÁS–TIPUS JELÖLÉSEK:

(15) A 15-ös csomópontnál a megelőző- és követő oldalélek azonosak

[3] A 3-as csomópontnál a következő alapszakasz azonos a generált előző elem jobboldali oldalélével

◇24 Belső peremszakasz

12 11
Az i-ik körbejáró sokszögvonal a 12–11 szakaszon záródik.
11 12

I/2. ábra

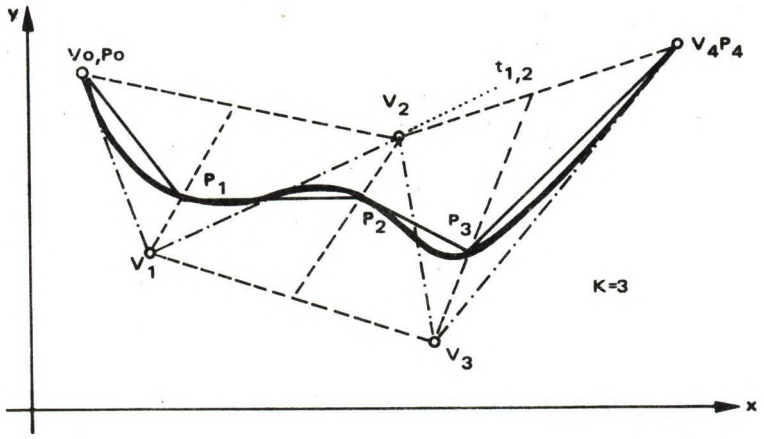
Háromszög-elem generáló mintapélda

A feladatban alkalmazott B-Spline függvény* ismertetése

Adott egy $\underline{p}_o, \underline{p}_1, \ldots \underline{p}_n$ pontokkal megadott sokszögvonal. Olyan görbe illesztendő a p_i pontokra, mely folytonos, deriváltjaiban is illeszkedő szakaszokból áll és az egyes görbe-szakaszokat egy

$$\underline{p}_l(t) = \underline{a}_o + \underline{a}_1 t_l + \underline{a}_{k-1} t_l^{k-1} + \underline{a}_k t_l^k \qquad (1)$$

alakú egyenlet írja le, ahol $t \epsilon (0,1)$ a relatív szakaszkoordináta.



A megoldás menete két fázisból áll. Első lépésben egy olyan másik $(\underline{v}_o, \underline{v}_1, \ldots \underline{v}_n)$ sokszögvonal generálandó, melynél tetszőleges $\underline{v}_{l-1}$, $\underline{v}_{l+1}$ pontokban deriváltjaiban is érintkező k-ad fokú parabóla $\underline{p}_i$ ponton megy át. Ez az alábbi egyenlet megoldásával történik:

$$\underline{\underline{K}}\ \underline{v} = \underline{p} \qquad \text{, ahol } \underline{\underline{K}} \text{ egy olyan hiperdiagonál} \qquad (2)$$

mátrix, ahol a belső pontokban $\underline{p}_i = \frac{1}{2k}(\underline{v}_{i-1} + 2(k-1)\underline{v}_1 + \underline{v}_{i+1})$, míg a végpontban $\underline{p}_i = \underline{v}_i$. Zárt görbe esetén $\underline{v}_o = \underline{v}_{n+1}$, azaz $\underline{\underline{K}}$ egy ciklus mátrix-szá fajul, mivel

$$\underline{p}_n = \frac{1}{2k}(\underline{v}_{n-1} + 2(k-1)\underline{v}_n + \underline{v}_1) \text{ ill. } p_o = \frac{1}{2k}(\underline{v}_1 + 2(k-1)\underline{v}_o + \underline{v}_n) \qquad (3)$$

azaz a megoldandó 2 egyenletrendszer együtthatómátrix-sza alábbi formát ölti:

$$\underline{\underline{K}}_{nyílt} = \frac{1}{2k} \begin{bmatrix} 2k & o & & & & \\ 1 & (2k-1) & 1 & & & \\ & 1 & (2k-1) & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & (2k-1) & 1 \\ & & & & o & 2k \end{bmatrix} \qquad \underline{\underline{K}}_{zárt} = \frac{1}{2k} \begin{bmatrix} (2k-1) & 1 & & & 1 \\ 1 & (2k-1) & 1 & & \\ & 1 & (2k-1) & 1 & \\ & & \ddots & \ddots & \ddots \\ & & & 1 & (2k-1) & 1 \\ 1 & & & & 1 & ((2k-1) \end{bmatrix} \qquad (4)$$

* Bernstein-Beziér bázisú spline interpolációs függvény

A második lépésben a $\underline{p}_l(t)$ vektor-függvény $\underline{a}_i^{(l)}$ együtthatóvektorait számítjuk ki az egyes $l = 1, 2, \ldots n-1$ (zárt görbe esetén n) szakaszokra külön-külön az alábbi peremfeltételek alapján:

$$\underline{p}_l(0) = \frac{1}{2k}(\underline{v}_{l-1} + 2(k-1)\,\underline{v}_l + \underline{v}_{l+1}) \qquad \underline{p}_l'(0) = \frac{1}{2}\underline{v}_{l+1} - \underline{v}_{l-1}$$

$$\underline{p}_l(1) = \frac{1}{2k}(\underline{v}_l + 2(k-1)\,\underline{v}_{l+1} + \underline{v}_{l+2}) \qquad \underline{p}_l'(1) = \frac{1}{2}\underline{v}_{l+2} - \underline{v}_l \tag{5}$$

Ezáltal az együttható-vektorok az alábbi módon számíthatók:

$$\begin{bmatrix} 1 & . & . & . \\ 1 & 1 & 1 & 1 \\ . & 1 & . & . \\ . & 1 & k-1 & k \end{bmatrix} \cdot \underline{\underline{A}}_l = \frac{1}{2k} \begin{bmatrix} 1 & 2(k-1) & 1 & . \\ . & 1 & 2(k-1 & 1 \\ -k & . & k & . \\ . & -k & . & k \end{bmatrix} \cdot \underline{\underline{V}}_l, \text{ ahol} \tag{6}$$

$$\underline{\underline{A}}_l^{(k)T} = \left[\underline{a}_o^{(l)}\ \underline{a}_1^{(l)}\ \underline{a}_{k-1}^{(l)}\ \underline{a}_k^{(l)}\right]^T \quad \text{ill.} \quad V_l^T = \left[\underline{v}_{l-1}\,\underline{v}_l\,\underline{v}_{l-1}\,\underline{v}_{l+2}\right]^T \tag{7}$$

Innen

$$\underline{\underline{A}}_l^{(k)} = \frac{1}{2k} \begin{bmatrix} 1 & 2(k-1) & 1 & . \\ -k & . & k & . \\ K\,(k-2) & 2k(2-k) & k\,(k-2) & . \\ -k^3+3k-1 & 2k^2-6k+3 & -k^2+2k-3 & 1 \end{bmatrix} \underline{\underline{V}}_l \tag{8}$$

Abban a jól ismert, harmadfokú esetben, amikor k=3

$$\underline{\underline{A}}_l^{(3)} = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & . \\ -3 & . & 3 & . \\ 3 & -6 & 3 & . \\ -1 & 3 & -3 & 1 \end{bmatrix} \tag{9}$$

Az illesztési feladat a fentiekből láthatóan mindig megoldható, mivel a (3) alapján összeállítható, nyitott görbe esetén kvázi-kontinuáns, zárt görbe esetén ciklikus mátrix $k \geqslant 2$ esetben mindig reguláris, hiszen az alappontok száma csupán az azonos főátlóbeli számhármasok számát növeli. Az invertálandó mátrix formája nem változik.

Belátható az is, hogy a jelzett $\underline{\underline{K}}$ mátrix független a csomópontok koordinátájától. Mindez rögtön sugallja a megoldási metódust törés és nyíródásvonalak esetére. Az előbbi esetben

$$\underline{p}_l = \underline{p}_{l+1} = \underline{p}_{l+2},$$

azaz háromszor megismételjük a töréspont koordinátáját. Úgy mond nulla sokszögvonal-szakaszokat alkalmazunk. Nyíródás-vonal esetén megkülönböztetünk bal- és jobboldali töréspontot, ahol mindkettő a jelzett vonalon fekszik, de általában nem esnek egybe.

az a jobboldali töréspontot szintén háromszor kell megadni. A kettőt egyenes szakasz köti
ze.

A B–Spline függvény felvételekor a leggyakrabban alkalmazott eset a (9)-ben külön is
tüntetett kubikus interpoláció. A jelzett „k" értékének növelésével valójában a függvény
ollokációs görbületét növelhetjük meg pl. olyan esetekben, amikor a kapott ponthalmaz
ösen tagolt morfológiával rendelkezik.

odalomjegyzék

1] Sheng-Chuan Wu, J.F. Abel & Greenberg: An Interactive Computer Graphics Approach to Surface Representation. Communications of the ACM, Vol 20, No. 10, 703–712 (1977. október)

2] H.Hahn, J. Radloff: Ein Verfahren zum Berechnen und Zeichnen von Niveankurven einer reellen Funktion zweier Verändelicher. Elektronische Rechenanlagen, 14. Jahrgang, No. 3, 128–133 (1972. június)

3] Hiroshi Akima: A Method of Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures Communications of the AVM, Vol 17, No. 1, 18–20. (1974. január)

4] M.Mor,T. Lamdan: A New Approach to Automatic Scanning of Contour Maps. Communications of the ACM, Vol 15, No. 9, 809–812 (1972. szeptember)

5] D.F.Rogers, J.A.Adams: Mathematical Elements for Computer Graphics. McGraw-Hill, 1976.

PERSONAL COMPUTEREK

Zámori Zoltán
KFKI

Összefoglalás:

Az előadás beszámol a számítógép-spektrum „legnagyobb áthaladó képességű sávjának" – a personal computereknek – a jelentőségéről. Ismerteti az ezen a vonalon felvetett hazai célkitűzéseket, amelyek elsősorban a közoktatás irányában nyitják meg a lehetőségeket.

A nagyteljesítményű és olcsó (ma csupán 10–20 dollár áru) mikroprocesszorok 1974/76-ban történt megjelenése elvben lehetővé tette, hogy velük komplett univerzális számítógépeket hozzanak forgalomba a konvencionális – még kis és közepes integráltsági fokú integráltáramkörökből felépülő számítógépek árának tört részéért. Az első figyelemreméltó konstrukciók 1977 nyarára készültek el s 1978-ban kerültek százezres példányszámban forgalomba. Összefoglaló kategória nevük personal computer lett, amely név azt az aspektusukat kívánja hangsúlyozni, hogy az áruk már a személyi tulajdont és használatot is lehetővé teszi szemben az eddig csak intézmények által installálható konvencionális gépekkel., Számítástechnikai teljesítményük máig elérte és fokozatosan meghaladja a klasszikus kisszámítógépek teljesítményét.

A personal computerek mai ára 600 és 1200 dollár körül van.

Ebben az árban a processzoron és a memória (általában 8 bites 2–4 MHz-es processzor, 8–16 KByte ROM, 4–16 KByte RAM) kívül bennefoglaltatik a display és teljes alfanumerikus klaviatúra, valamint a háttértárként szolgáló – bár alacsony (500 Baud) átviteli sebességű, de a nehézkesen kezelhető és költséges lyukszalag perifériákat maradéktalanul kiváltó – digitális felírást alkalmazó közönséges kazettás magnetofon.

Az eddig közel félmillió példányszámban piacra hozott personal computerek legnépszerűbb típusai ma a következők:

TRS–80	μP:Z–80, ROM:12 KByte, RAM:4/16 KByte Keyboard, Display, Magno, BASIC	699 $
PET–2001	μP:6502, ROM:16 KByte, RAM:8 KByte Keyboard, Display, Magno, BASIC, IEC bus	795 $
OHIO SBII	μP:6502, ROM:8 KByte, RAM:4 KByte Keyboard, BASIC, TV és Magnó csatlakozás lehetősége, tápegység és dobozolás nélkül	279 $

A harmadik tételként említett DB II (teljes nevén Superboard II) personal computer adatai azt jeltik, hogy már ma is beszerezhető 300 dollár alatt egy univerzális számítógép minden lényeges – itthon még nem, vagy csak kisintegráltságú elemekből aránytalanul magas áron előállítható – komponense. Az egyetlen nyomtatott lemezen megvalósított SB II a processzoron és tekintélyes nagyságú (egy BASIC interpretert is magábafoglaló) memórián kívül tartalmaz egy komplett írógépszerű klaviatúrát (hazai változatban egyedül az utóbbi 30 000 Ft) valamint olyan interface áramköröket, amely egy közönséges TV-nek display-ként illetve kazettás magnónak háttértárként való alkalmazását engedi meg.

A nagyteljesítményű és sokoldalúan használható számítógépeknek ez az alacsony ára arra figyelmeztet, hogy a mikroelektronikai elemeken alapuló közismerten hasznos (kalkulátor, pénztárgép, egységárra beszorzó mérleg, programmozható óra stb.) és az amerikai szóhaszná-

latban, gadget-nek minősülő haszontalan (TV játék, diéta, bioritmus, horoszkóp készítő stb.) eszközökön túl a programozható univerzális számítógépek — s ezzel a számítástechnikai alkalmazási ismeretek — legszélesebbkörű elterjedésére számíthatunk.

Ez hatásossá, számunkra és minden technológiailag nem eléggé fejlett ország számára szinte fenyegetővé akkor válik, ha a mindenféle termelési folyamatok automatizálását olcsón megengedő mikroelektronikai eszközök — maguknál az eszközöknél is jóval nehezebben (időigényesebben) megszerezhető — széles körű alkalmazási ismeretekkel párosulnak. Amíg erre nem kerül sor, addig mindenütt csak a professzionális alkalmazások megvalósulására lehet számítani. Ez a termelési struktúra egészét tekintve lassú, nem fenyegető folyamat, melyet professzionális szinten — mint a hazai számítástechnikai sikerek is jól példázzák — lokalizált szakismeretekkel is követni tudunk.

Amint azonban az alkalmazási ismeretek az olcsón elérhető számítógépeken keresztül a széles tömegekhez — elsősorban a fogékony ifjúsághoz is elérnek, úgy minden a termelő ember jelenlétét feleslegesen megkövetelő tevékenység gépekre való áthárításának, s így a termelékenység szinte beláthatatlan növelésének nyílik majd meg az útja. Ugyanis a sokoldalúan felhasználható és olcsón előállítható félvezető mikroelektronikai eszközök zöme már ma — és hónapról hónapra tökéletesedő formában — a rendelkezésünkre áll.

Az olcsón elérhető personal computernek nevezett univerzális számítógépeknek ebben az ismeretközvetítő folyamatban van társadalmi jelentősége.

A 600—800 dollár egy komplett gép — vagy a 279 dollár a még kiegészítéseket igénylő SB—II gép — esetén még nem tekinthető eléggé olcsónak. Nem annyira, hogy mindenkihez elérő gyerekjáték legyen.

Azonban a Texas Instruments által az elmúlt karácsonyra kihozott és a jelen előadáson is hangos demonstráció céljából a pódiumról bemutatott „Speak & Spell" nevű készülék 50 dolláros ára már olcsónak minősíthető. A készülék piros plasztikdoboza egy dedikált mikroprocesszort, 32 KByte (a szintén megvásárolt 15 $ áru bővítéssel 48 KByte) ROM memóriát, egyszerű kivitelű klaviatúrát, valamint hangszórót tartalmaz. Az összesen benne levő közel 50 KByte ROM memória tehát mindössze 1/20-ad része a KFKI R—40-es típusú nagyszámítógépében levő összesen 1 MByte RAM memóriának. (A pontos hasonlításhoz még megjegyezhető, hogy a ROM: RAM memória árak viszonya 1:4). Ez a tény — mint ahogy maga az eszköz is — magáért beszél. Ebből kell konklúziókat levonni a jövőre vonatkozólag.

Év eleje óta folyt a találgatás, hogy milyen áru és teljesítményű lesz A TI-nek az NCC megnyitásáig bemutatásta ígért personal computere. Az ennél kialakítandó árpolitika ugyanis az egész szakma pozícióját determinálja. Technológiailag ugyanis — például a Speak & Spell készülék fényében — a Texas Instruments (70 000 termelő dolgozó, szemben az IBM 330 000 zömében vevőszolgálatos dolgozójával) már ma képes lenne egy 200 dollár körüli számítógép piacra vitelére. Az egyébként is uralkodó számítástechnikai konjunktúra figyelembevétele alapján megfogalmazott marketing politika azonban nem tette ezt számára ésszerűvé. A végül is a bejelentett és ősztől forgalmazott 16 bites processzorból, 16 KByte RAM és 26 KByte ROM alapkészletből felépülő, színes grafikus (256x192 felbontású) display-t is magába foglaló TI 99/4 típusszámot nyert personal computer ára 1150 dollár lett. A beszédkimenetet is lehetővé tevő opció ára 150 $. Mindez árban és minőségben szép teljesítmény, de még nem a kiszámíthatatlan következményekkel járó bravúr. Az pár évet még várat magára, de beköszöntésének puszta lehetősége sem lehet közömbös számunkra.

Mindezeket tisztán látva, az OMFB a múlt év végére a KFKI-val kidolgoztatta egy olyan univerzális számítógép műszaki specifikációját (lásd a függeléket), amelyekkel e hamarosan mindenhová benyomuló szakma széles körű alkalmazási lehetőségeiről a hazai ifjúságot szervezetten — legkézenfekvőbben az iskolákon keresztül — jóelőre tájékoztatni lehetne.

Ugyancsak a függelékben közöljük a fent említett jelentésből azt a táblázatot, amely megmutatja, hogy egy itthoni konstrukció esetén – az egyik piacon levő amerikai perconal computert példának választva – milyen import illetve hazai költségösszetevőkre kellene számítanunk.

A melléklet specifikációt tartalmazó beszámoló jelentést z az OMFB az év elején szétküldte a hazai számítástechnikai gyártó vállalatoknak, amelyek nyugati kooperációs tárgyalások nyomán már nyár végére be tudtak nyújtani egy olyan javaslatot, melynek eredményeként így nagy teljesítményű kooperációban gyártott és hazai forgalmazású personal computerek már a közeljövőben legalább az iskolákban – tehát a jövő generáció versenyképes felkészítése szempontjából a legfontosabb helyen – itthon is megjelenhetnek.

A leendő ISKOLASZÁMÍTÓGÉP műszaki specifikációja

1. Az ISKOLASZÁMÍTÓGÉP-nek teljesértékű tárolt programozású univerzális számítógépnek kell lennie.
(Tehát nem kalkulátor, TV game, mikroprocesszor kit, stb.)

2. Az ISKOLASZÁMÍTÓGÉP ára nem lehet több, mint egy színes TV készülék árának 2–5 szöröse.
(A sorozatgyártás volumenétől függően.)

3. Az ISKOLASZÁMÍTÓGÉP számítástechnikai teljesítményének az IBM 360/30 illetve az ESZR gépcsalád R–20 típusú középkategóriájú gépének a nagyságrendjében kell lennie.
Itt a gép processzorának, operatív memóriájának és input/output pontjainak sebessége illetve szélessége által meghatározott úgynevezett belső számítástechnikai teljesítményről van szó, amelyet az alkalmazott perifériák (magno, disk, kinyomtató) professzionális (de igen költséges) vagy nem professzionális (de azokat mindenben szimuláló, csak lassabb, viszont olcsóbb) válfaja kisebb vagy nagyobb mértékben leronthat.

4. Az ISKOLASZÁMÍTÓGÉP céljaira külföldi gyártmánylistából kiválasztott, vagy külön erre a célra itthon konstruált számítógépnek az iskolákban való bevezetéshez három év múlva ezres nagyságrendben rendelkezésre kell állnia.

5. Az ISKOLASZÁMÍTÓGÉP megjelenítő egysége a költséges speciális írógépek (Teletype, Telex, Kódvezérelt villanyírógép) helyett TV monitor vagy közönséges (de a későbbiekben célszerűen video-szinkron jelbemenettel is ellátott) televíziós vevőkészülék legyen, s az ilyen megjelenítő teljesítménye semmibe se maradjon el a mai professzionális display-k teljesítményétől.

6. Az ISKOLASZÁMÍTÓGÉP és a felhasználói közti kommunikáció épp úgy teljes alfanumerikus klaviatúrán történjen, mint a konvencionális nagygépek esetén.
(Ez teljes ASCII karakterkészletű írógépszerű klaviatúrának s nem pedig bináris, oktális vagy hexadecimális billentyűzetnek felel meg.)

7. Az ISKOLASZÁMÍTÓGÉP háttértára közönséges olcsó kazettás magnetofon legyen. Ezen őrizhetők és terjeszthetők a különböző demonstrációs programok és adatfile-ok.

8. Az ISKOLASZÁMÍTÓGÉP rendelkezzen TV készüléken megjeleníthető grafikus, de legalább szemigrafikus display kimenettel.

9. Az ISKOLASZÁMÍTÓGÉP rendelkezzen ROM-ban őrzött monitorprogrammal, amely a gépi kódú programozást, editor programmal, amely a szövegszerkesztést teszi lehetővé, és egy magas szintű programozási nyelv – a mai divat szerint BASIC – ugyancsak ROM-ban tárolt interpreterével.

10. Az ISKOLASZÁMÍTÓGÉP operatív tára minimálisan 4 K Byte legyen, amely 64 KByte-ig egyszerűen bővíthető.

11. Az ISKOLASZÁMÍTÓGÉP időméréshez, az olcsó (1 $) V/F konverterek egyszerű használatához, illetve a különböző ipari feladatok szimulálására írt task-ok ütemezéséhez rendelkezzen real time órával.

12. Az ISKOLASZÁMÍTÓGÉP a vezérlési, folyamatszabályozási és automatizálási feladatok demonstrálásához rendelkezzen legalább két byte széles programozható input/output port-tal (amely I/O interface chip-pek hozzáadásával tovább bővíthető), legalább egy 16 bit széles-számláló ill. időmérő bemenettel, s egy-egy legalább 8 bit pontosságú analóg be- ill. kimenettel.

13. Az ISKOLASZÁMÍTÓGÉP rendelkezzen IEC/IEEE 488 Standard/ bus kontrollerrel és csatlakozóval, amelyen keresztül hozzá kívánatra nagyteljesítményű professzionális perifériák (disk, floppy, line printer, később különböző ipari műszerek) köthetők.

14. Az ISKOLASZÁMÍTÓGÉP-hez rendszeresíteni kell egy forrasztásnélküli összekötést megengedő szerelőlapot (breadboard), amelyen keresztül kapcsolók, relék, motorok, mérőműszerek stb., azaz általánosan megfogalmazva: érzékelő és beavatkozó szervek illeszthetők (interface-elhetők) a géphez.

15. Az ISKOLASZÁMÍTÓGÉP-ben legyen lehetőség arra, hogy hozza bővítésként soros aszinkron adatátviteli egységeket kössenek, amelyek segítségével a későbbiekben megszülető közcélú adatbankokat is elérheti a telefonhálózaton keresztül.

16. Az ISKOLASZÁMÍTÓGÉP központilag és folyamatosan ellátandó majdan olyan kazettákon forgalomba hozott demonstrációs programcsomagokkal, amelyek betöltése és futtatása semminemű szakértelmet nem követel, s ugyanakkor mindegyike ékesen és didaktikailag magas szinten demonstrálja a számítógép alkalmasságát a legkülönbözőbb termelési tevékenységek megkönnyítésére illetve automatizálására, s ezzel számítástechnikai ismeretek megszerzésére inspirál.

Megjegyzés: A tanulmányterv vitán bemutatásra és részletes ismertetésre kerülő egyik (a jelen szerződés keretében behozott PET 2001) mikroszámítógép a fenti specifikáció követelményeinek majdnem minden pontban eleget tesz.

I. TÁBLÁZAT

A Commodore PET 2001 típusú számítógép főbb alkatrészeinek listája. (A tőkés alkatrészek ára kis (egyes) tételre vonatkozik.)

Tőkés eredetű alkatrészek

Típ.	Megnevezés	á	db	Ár
6502	Mikroprocessor	11.95 $	1	11.95 $
6520	I/O LSI	10.00	2	20.00
6522	I/O LSI	9.25	1	9.25
2114	RAM 1kx4 bit	6.95	18	125.10
2716	PROM 2kx8 bit	25.00	8	200.00
74LSxx	TTL IC különböző	0.40	45	18.00
		tőkés import összesen:		384.30 $
		ami az EMO 84 FT/ $ átszámítási+ faktorával:		32.281.00 Ft

Hazai beszerzésű alkatrészek ill. részegységek

Nyomtatott áramköri lemez	Sárisáp TSz	3.000,– Ft
Tápegység (5V, 5A)	HTSz	5.000,–
Klaviatúra	TKI	30.000,–
TV monitor	Orion	25.000,–
Kazettás magno (kis átalakítással)	BRG	2.500,–
Doboz	Kontakta	3.000,–
	hazai részegységek összesen:	68.500,– Ft
	Tőkés alkatrészek és hazai részegységek	100.780,– Ft
	A PET 2001 komplett számítógép USA ára	795,– $
	Ez 100 Ft/$ átszámítási+ faktorral	79.500,– Ft

+Ez magában foglalja a szállítást, vámot, adót, kereskedelmi hasznot is.

Instruction/programming manual

12" video display

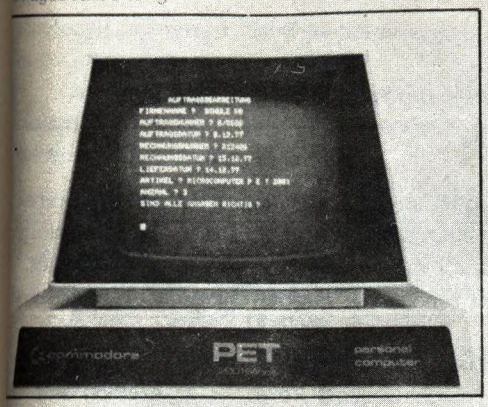
Power supply

Cassette recorder

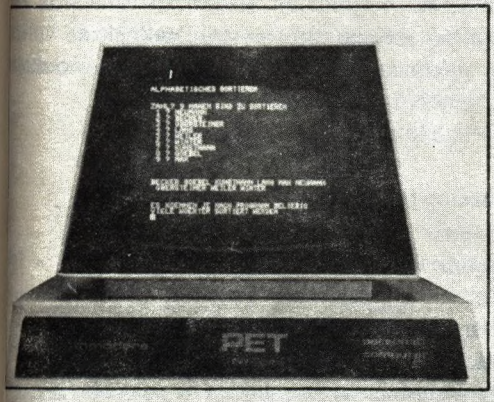
Computer with built-in keyboard


Computer
PET2001
PET
commodore
PET
personal
computer

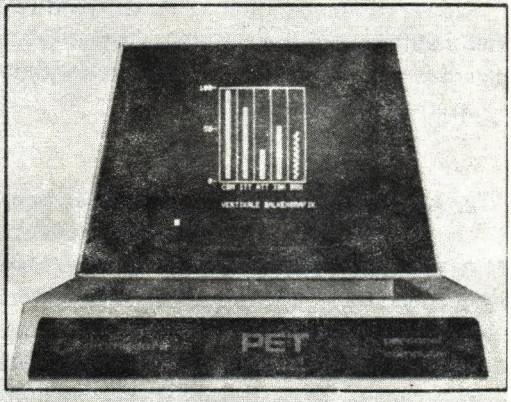
Auftragsbearbeitung



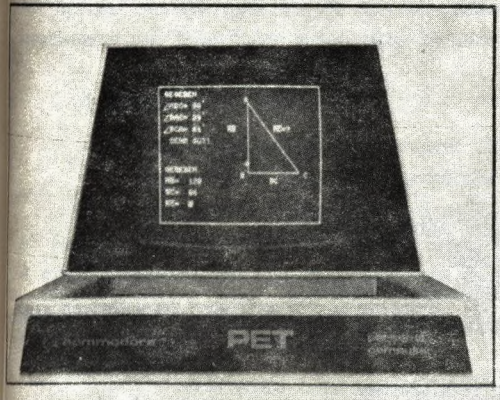
Alphabetisches Sortieren



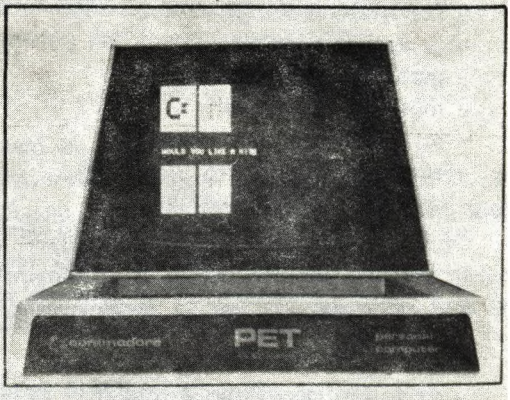
Grafik



Trigonometrisches Lernprogramm



17 + 4 (Blackjack)



A RENDSZERELVŰ ÉPÍTÉST SEGÍTŐ SZÁMÍTÓGÉPES RENDSZER

dr. Zeley István
ÉGSZI

1. A rendszerelvű építés — az építészet jövője

Az építészet rendszerelvű megközelítése az egyetlen út a társadalom építési igényeinek magasszintű kielégítésére. A rendszerszemlélet alkalmazása jelenti az építés valódi iparosítását, ami nem kizárólag — és nem elsősorban — a műszaki megoldásokra terjed ki (előregyártás, gépesítés) hanem a szervezési, szervezeti, működési rendszerek iparban alkalmazott módszereire is.

A rendszerelvű építés építési rendszerekben valósul meg. Műszaki oldalról az építési rendszer az építési elemek — rendszerkomponensek — meghatározott körét, és az ezekkel való tervezési, kivitelezési, gyártási és szervezési megoldásokat jelenti.

Szervezeti szempontból az építési rendszer műszaki megoldásait alkalmazó vállalatok zárt körét a rendszerépítő szervezetnek nevezzük. Működési szempontból az épületeknek az építési rendszerben való megvalósítása egységes folyamat, amely áttöri a korábbi — sokszor indokolatlan — szervezeti határokat.

2. A rendszerelvű építés irányítási és információs rendszere

A rendszerelvű építés bevezetésének — hosszabb távon pedig általánossá tételének — feltétele a hatékony irányítási rendszer kidolgozása.

A rendszerelvű építés irányítási rendszerét három szinten kell megszervezni:

- az építési rendszerek környezeti rendszere
- az építési rendszerek közti kapcsolatok rendszere
- az építési rendszereken belüli irányítási rendszer.

A környezeti irányítási rendszer főbb funkciói:

- az ágazati irányítás
- gazdasági, jogi szabályozás
- műszaki szabályozás
- az építési piac megszervezése
- az építés feltételeinek biztosítása
- az ipari háttér kialakítása.

Az építési rendszerek közti irányítási rendszer néhány funkciója:

- csereszabatos rendszerkomponensek tervezése
- a csereszabatos rendszerkomponensek kölcsönös szállítása
- a tervező, gyártó és kivitelező kapacitásokkal való, építési rendszerek közti gazdálkodás
- az építési igényeknek a megfelelő építési rendszerhez való irányítása.

Az építési rendszeren belüli irányítás funkcióinak két fő csoportja:

- az építési rendszer kifejlesztése
- az építési rendszer alkalmazása építmények létrehozásához.

3. A rendszerelvű építést kiszolgáló számítógépes rendszer

A rendszerelvű építés lehetőségeinek kiaknázása megköveteli a számítástechnika segítségét.
Mindegyik irányítási szinthez számítógépes információs rendszert célszerű kidolgozni és alkalmazni.

A rendszerelvű építés környezeti irányítási rendszerét és az építési rendszerek közti kapcsolatot kiszolgáló számítógépes rendszer koncepciója még nem készült el — ez a rendszerelvű építés fontos kutatási feladata.

Az építési rendszereken belüli funkciókat segítő számítógépes rendszer általános koncepciója a SZÁMGÉP az elmúlt években kidolgozta, és az ÉVM jóváhagyta. Ennek alapján a SZÁMGÉP, illetve az ÉGSZI a számítógépes rendszer több modulját elkészítette, és egyes építési rendszereknél megkezdődött azok bevezetése.

A cikk további részében az építési rendszereken belüli folyamatok számítógépes rendszerének általános elveit ismertetjük, részletesebben kitérve a kulcsfontosságú, és már elkészült modulokra.

3.1 Az építési rendszerek alkalmazásának reálfolyamatai

Az építményeknek egy építési rendszer műszaki megoldásaival és szervezésével való megvalósítása a következő főbb folyamatokat tartalmazza:

- vállalkozás és kapacitás-gazdálkodás
- épülettervezés — termeléselőkészítés
- gyártás
- kivitelezés.

A számítógépes információs rendszer e reálfolyamatokat követi.

A mai építési gyakorlatban egy-egy építmény megvalósításához egyszeri, sokszor esetleges kapcsolat jön létre a tervezők, a kivitelezők, a gyártók és szállítók között. A kapacitások megkeresése, a szervezetek közti egyeztetés erőforrás veszteséggel jár. A megvalósítás során a szervezeti határok megszakítják az irányítási és információs folyamatot. A tervező munkája a végrermékének a kivitelezési tervdokumentációt tekinti, a kivitelező pedig a terveket a termelésirányításhoz kell, hogy alakítsa, kivitelezési folyamatokká szervezze.
Az építési elemek gyártása és szerelése között kevés az információs kapcsolat és az elemek beszerzésének lehetősége előre nem látható módon hat vissza a megvalósításra.

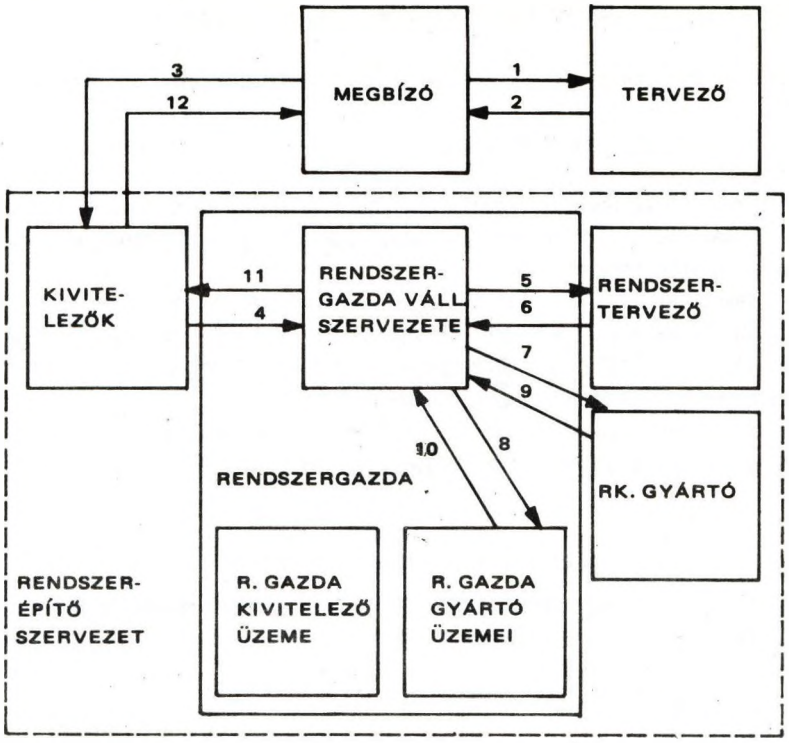
A rendszerelvű építésnél

- a szervezetek közti kapcsolatok állandó és
- a megvalósítás reál- és információs folyamatai egységes rendszert alkotnak (1. ábra).

A rendszergazda megszervezi és nyilvántartja a rendszerépítő szervezet tervező, kivitelező és gyártó vállalatainak az építési rendszer rendelkezésére bocsájtott kapacitásait.
A rendszertervezővel együtt kidolgozza és karbantartja az építési rendszer műszaki megoldásait tartalmazó rendszerdokumentációt. Az építményeknek az építési rendszerben való létrehozása egységes folyamatként szervezi.

3.2 Az építési rendszer alkalmazását segítő számítógépes rendszer

Az építési rendszerek integrált megvalósítási folyamatainak megfelelően integrált számítógépes információs rendszert terveztünk, amely adatbankra épül, és a vállalkozás, kapacitásgazdálkodás, tervezés, gyártás és kivitelezés információs részrendszereiből áll.



Az építési rendszer alkalmazásának egyik változata: a rendszergazda koordinációs tevékenységet folytat.

1. ábra A rendszerépítő szervezet

Lehetővé válik, hogy az épülettervezés és a termelésirányítás ugyanazt az adatbázist használja, és így a tervezés eredménye a kivitelezés irányításához közvetlenül, átalakítás nélkül felhasználható, a kivitelezési ütemterv pedig a rendszerkomponens gyártás kiinduló adata lehet.

3.21 Az adatbank

A számítógépes rendszer középpontjában az adatbank áll, és ez biztosítja a részrendszerek közti kapcsolatot.

Az adatbank többszintes, hierarchikus felépítésű, és az alapanyagtól kezdve a végtermékig, az épületig tartalmazza a megvalósítás minden fázisának elemeit.

Egyszerre végtermék és folyamatszemléletű: megtalálhatók benne a megvalósítás egyes szakaszainak eredményei, és a létrehozásukhoz szükséges folyamatok adatai.

Az adatbank tartalmazza az építési rendszer elemei közti kapcsolatokat, és alkalmas azok többféle struktúra szerinti kezelésére.

Az építési rendszer adatbankjának egyes elemeit az alábbiakban felsoroljuk. Az elemek közül néhányat többféle struktúrába szerveztünk, ezért a felsorolás nem jelent alá, illetve fölérendeltséget.

Az adatbank elemei:

– épületterv
– hálóterv

– funkcionális egység
– alrendszer egység
– kivitelezési tevékenység
– folyamattétel
– ÉKN-tétel
– rendszerkomponens
– alkatrész
– alapanyag

Az adatbank szerepe kettős. Egyfelől tartalmazza a feldolgozásokhoz szükséges törzsadatokat, mindegyik részrendszerhez, másfelől alkalmas a feldolgozások eredményeinek a következő feldolgozáshoz való tárolására.

Az iparosított, rendszerelvű építéshez az ipari termelésirányításban bevált elvek szerint kialakított adatbázis-kezelő rendszert alkalmazunk. Ez elsődlegesen 4 alapadattárat kezel:

– cikktörzsadattár
– munkahely törzsadattár
– darabjegyzék struktúra adattár
– művelet struktúra adattár

A cikktörzsadattár az egyes tételek törzsadatait (azonosító, megnevezés, jellemzők) tartalmazza, a darabjegyzék struktúra adattár pedig az e tételek közti kapcsolatokat.
A munkahely törzsadattár az e kapcsolatok realizálásához (a tételek egymásba épüléséhez) szükséges szakmák, gépek, munkahelyek törzsadatait (azonosítók, megnevezés, jellemzők, pl. kapacitások) tartalmazza, a műveleti struktúra adattárban pedig a megvalósítási normaidőket tároljuk, egyfelől a beépítendő cikkhez, másfelől a munkát végző munkahelyhez kapcsolódva (2. ábra).

Egyes tételeknél minden adatot előre fel tudunk vinni az adatbankba (pl. rendszerkomponens), más tételek bizonyos adatait pedig csak valamely feldolgozás eredményeként tárolhatjuk ott (pl. az épület).

A számítógépes információs rendszer felépítését és működését az adatbankra támaszkodva ismertetjük. Az adatbank egyes tételeinek tartalmát és a köztük lévő kapcsolatot az adatbankra épülő feldolgozásokkal párhuzamosan tárgyaljuk.

3.22 Épülettervezés-termeléselőkészítés

A számítógépes rendszer integráltságának döntő láncszeme *a tervezés és a termelés egységes rendszerbe való szervezése.* A termelésirányításnak három szintje van: az építési rendszer szintjén való termelésirányítás (vállalkozás, koordinálás), a kivitelezés irányítása és a rendszerkomponens gyártás irányítása.
A tervezés eredményének közvetlenül ki kell szolgálnia mindhárom szintet. A tervezés végterméke nem a mai értelemben vett kivitelezési tervdokumentáció, hanem a termeléselőkészítésnek a termelésirányításhoz közvetlenül felhasználható minden adata.
A tervezés-termeléselőkészítés egységes rendszerként való kezelését nemcsak a rendszerelvű építés folyamatainak integráltsága indokolja, hanem az az ágazati célkitűzés is, hogy a *kivitelezési tervek* készítése fokozatosan váljék a *kivitelező vállalatok feladatává.* Számítógépes rendszerünk a mai szervezeti elkülönültség mellett is segíti a tervezés és a termelés közti kapcsolat szervezettebbé tételét, és segíti majd a kivitelező vállalatokat a tervezési feladatok átvállalásában – legalább építési rendszereiknél.

A következőkben a számítógépes tervezés-termelésirányítási rendszert részfolyamatokká bontva vizsgáljuk.

2. ábra Az építési rendszerek integrált információs rendszerét kiszolgáló adatbank felépítése

3.221 Épülettervezés

Nyílt építési rendszereknél nem lehet típusépületeket előre tárolni az adatbankban, azokat esetenként kell megtervezni az adatbankba előzetesen felvitt elemekből (tervezési egységekből, rendszerkomponensekből).

Az információkezelés szempontjából nézve a tervezés eredménye nem más, mint az új tételként felvett épület, és a hozzá kiválasztott tervezési egységek közötti viszony (struktúra) definiálása. (Beépülési mennyiségek, térbeli elhelyezkedés.) A tervezett épület struktúrájának az adatbankban történő rögzítésével az épület „számítógépes modell"-jéhez jutunk. A termeléselőkészítés, termelésirányítás feldolgozásai ezután már teljes egészében az adatbankból történetnek.

Az adatbankból való épülettervezésnek két szintje van:

– az automatizált és
– az interaktív

épülettervezés.

Az automatizált épülettervezés kutatási stádiumban van, egyes építési rendszerek néhány alrendszerére készültek működő programok. Ezek segítségével az adott elemválasztékból, pontosan megadott követelményértékekhez automatikusan lehet beválogatni az azokat legjobban közelítő teljesítménnyel rendelkező tervezési egységeket. Ott alkalmazhatók, ahol a kiválasztás lépései egzaktan algoritmizálhatók.

Az interaktív épülettervezésnél a tervező a számítógéptől csak segítséget kap. Az adatbázisból lekérdezheti a tervezési lépésnél figyelembevehető elemválasztékot, az elemek, egységek jellemző paramétereit. A kiválasztott elemek típusát, mennyiségét közvetlenül az adatbankban rögzítheti. Egy-egy tervezési szakasz (pl. egy alrendszer tervének elkészítése) után gépi ellenőrzés kérhető. A műszaki megoldás helyességét, a hatósági előírások teljesítését, a költségnormák betartását, a beválogatott termékek beszerezhetőségét, stb. kirajzoltatás után pedig a formai megjelenést értékelhetjük.

Az ellenőrzéshez a program általában lebontja a betervezett összetett tervezési egységek darabjegyzékét, majd pedig elvégzi a rendszerkomponens szinten tárolt műszaki, költség, stb. adatok összegzését. Az ellenőrizendő értékeken túlmenően előállítja a következő tervezési szakaszhoz szükséges kiinduló adatokat is (pl. terhelések, hőszükséglet, stb.). Az összetett rendszerkomponens-szükséglet azonnal összevethető a rendszerépítő szervezet készlet, vagy gyártókapacitás-adataival, tehát a tervezés közben figyelembe lehet venni a megvalósítás feltételeit is.

A tervvariáció készítés – ellenőrzés többszöri ismétlésével, iteratív lépésekkel válik véglegessé a tervezett épület „számítógépes modell"-je. Ezzel a tervezés folyamata tulajdonképpen be is fejeződött. Az eredmény megjelenése a termelésirányításhoz szükséges formában kell, hogy történjék, ezért azt a termeléselőkészítés rész-folyamatában tárgyaljuk.

3.222 Termeléselőkészítés

Az épülettervezés során az épületnek mint végterméknek a *„konstrukciós terv"*-et hoztunk létre, és ennek számítógépes modelljét vittük fel az adatbankba. A termeléselőkészítés feladata ennek megvalósítási, *„technológiai terv"*-vé való átalakítása, a megvalósítás ütemezése, az erőforrások megtervezése céljából.

A mai gyakorlatban ez a kivitelezési tervdokumentáció átalakításával történik – azaz a termeléselőkészítés a kivitelezési tervezés befejezése után kezdődik.

Vonatkozik ez a meglévő számítógépes rendszerekre is. Minden működő számítógépes termelésirányítási rendszer a tervdokumentáció (költségvetés) átdolgozásával, adatainak más rendszerben való újbóli rögzítésével kezdődik.

Az építési rendszereknél is az épület *„konstrukciós terv"*-ét készíti el a tervező. A számítógépre szervezett adatbank segítségével, automatizált vagy interaktív feldolgozásokkal alakítjuk át ezt az épület „technológiai tervvé"-vé.

Az adatbank – mint láttuk – a teljes építési folyamat legfontosabb összefüggéseit képezi le, modellezi. Az építési rendszer elemeit az adatbankban különböző struktúrák szerint tároljuk. A tervezés-termeléselőkészítés számítógépes rendszere

– a tervezés
– a termelés és
– a költségvetés-készítés

eltérő struktúráján alapul.

Az egyes tételeknek a három *struktúra* szerinti elrendezését, szintjeit a 3. ábra mutatja be.

A tervezés-termeléselőkészítés szempontjából az adatbank alábbi tételeit elemezzük:

– tervezési egység
– tevékenység
– folyamat és
– ÉKN tétel.

A tervezési egységek (funkcionális vagy alrendszer egység) a több rendszerkomponenst tartalmazó, és különböző épületeknél ismételten előforduló egységek (pl. homlokzati mező, hajó, keretállás, vagy pl. tantermi egység). Az épület konstrukciós terve tervezési egységekből áll.

A tevékenység a kivitelezés egyszerre, egy helyen végzett munkáit jelenti. Az épület technológiai terve (hálóterve) tevékenységekből áll.

A konstrukciós terv technológiai tervvé való átalakításához *a tervezési egységek és a tevékenységek közti összefüggést kell megkeresni.*

A tervezési egységek általában nem követik a kivitelezés-technológiát. Pl. a homlokzati mezőt egységként tervezik be, de nem egyszerre szerelik, hanem bizonyos rendszerkomponens csoportjait külön-külön szakaszban. Ezeket a rendszerkomponens csoportokat viszont nem tervezési egységenként, hanem az egész építményben egyszerre, egy tevékenységben szerelik be. E rendszerkomponens csoportokat, amelyek részét képezik mind a tervezési egységeknek, mind a tevékenységeknek, *„folyamat"*-nak nevezzük. A folyamatok segítségével hozhatjuk létre a rendszerelvű építésben a tervezés és a kivitelezés közti közvetlen információs kapcsolatot (4. ábra).

Az adatbank fenti tételei között a *kapcsolatok* a következők:

A rendszerkomponensek alkatrész és alapanyag tartalma meghatározott.

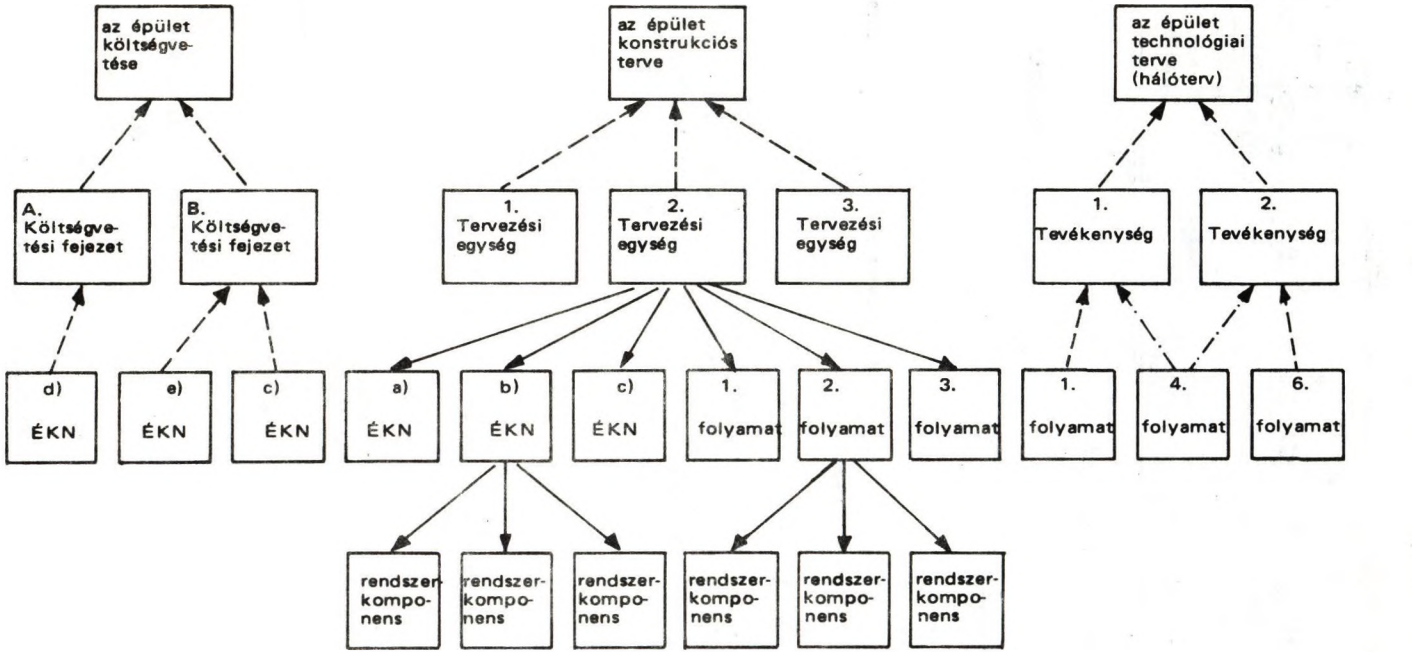
A folyamatokhoz egyértelműen hozzárendelhetjük a rendszerkomponenseket és alapanyagokat.

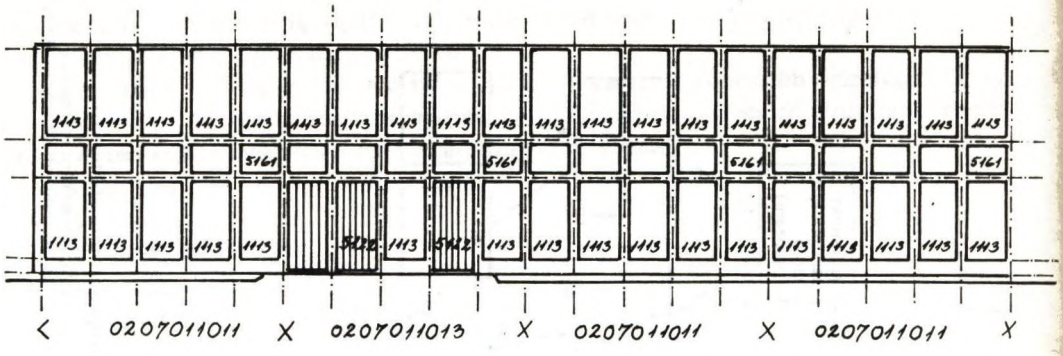
A tervezési egységekbe tartozó folyamatok típusát és mennyiségét egyértelműen meghatározhatjuk (folyamattétel).

A tevékenységekbe tartozó folyamatok típusa lehet adott, de több tevékenységhez is kapcsolhatók ugyanazok a folyamatok. Előző esetben automatikusan sorolhatók be az épület folyamat-tételei a tevékenységekbe, utóbbi esetben interaktív módon. A tevékenységbe tartozó folyamatok mennyisége viszont a tervezési egységek típusától és számától függően épületről-épületre változik.

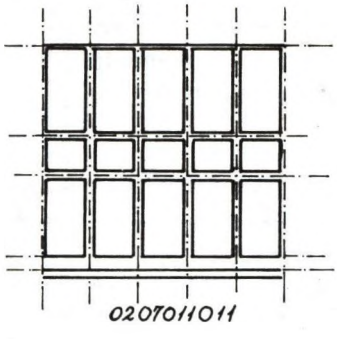
A tervezési egységre meg kell határozni a hozzá tartozó ÉKN tételeket, és ezt a struktúrát külön tárolni kell. Ideális eset az volna, ha egy folyamatnak egy költségvetési tétel (ÉKN tétel) felelne meg. A mai költségvetés-készítési előírások mellett erre nincs mód. Igy minden tervezési egységnek két „darabjegyzéke" lesz: egy a folyamatokból, egy pedig az ÉKN tételekből áll.

3. ábra Az építési rendszer tervezési-termeléselőkészítési adatainak struktúrája

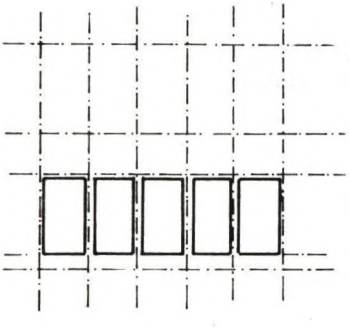




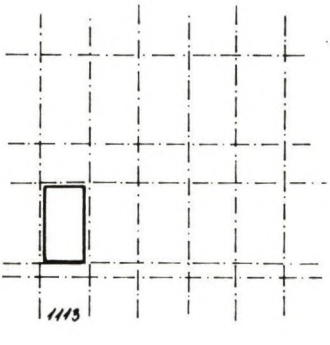
ALRENDSZER



TERVEZÉSI EGYSÉG



FOLYAMATTÉTEL



RENDSZERKOMPONENS

4. ábra

A rendszerelvű építés számítógépes termeléselőkészítési rendszere tehát a megtervezett épület adatbank tárolt „számítógépes modell"-jét használja kiinduló adatként. Az adatbank ezen új tételét, az „épületterv"-et a program lebontja a különböző szintekig, majd felépíti a kivitelezési és költségvetési struktúrákat.

A technológiai előkészítéshez, ütemezéshez a folyamattételekből újabb struktúrát építünk fel, ahol az új tétel a tevékenység, alárendeltjei pedig a hozzá tartozó folyamattételek.

Az így kialakított tevékenységek újbóli lebontásával és a rendszerkomponens szinten való összegzésével elkészítjük a tevékenységek szerinti rendszerkomponens konszignációt.

Kiszámítjuk és tevékenységenként összesítjük a munkaerő-szükségletet, a műveleti adattár felhasználásával, és meghatározzuk a tevékenység időtartamát.

Elvégezzük a háló időütemezését, majd kiszámítjuk az ütemezett erőforrás-szükségletet.

Az adatbank munkahely adattárában tárolt szerelési kapacitásadatokkal erőforrás-tervezést végzünk – azaz a hálót korlátokkal futtatjuk le. A folyamatokhoz rendelt rajzi szimbólumokból összeállítjuk a tevékenységekhez rendelt szerelési rajzot. Ez, eltérően a kivitelezési tervdokumentáció mai tartalmától, csak az egyszerre végzendő munkákat ábrázolja.

A tevékenységekhez rendelt rajzok, ütemterv és erőforrás-szükséglet alapján készülnek a munkautalványok.

A költségvetéskészítéshez az épület számítógépes modelljét lebontjuk az ÉKN tételek szintjéig, majd az azonos ÉKN tételeket összesítve és költségvetési fejezetenként összevonva elkészítjük a költségvetési kiírást.

A költségvetés beárazását a vállalati anyagárakkal, az ÉKN tételekben található díjakkal és mennyiségekkel, szükség esetén az inputként megadott organizációs adatokkal végezzük el.

A rendszer teljes kiépítése után tehát *egyetlen aktuális inputtal* (az épület műszaki terve – azaz az épületet alkotó tervezési egységek típusa, mennyisége és koordinátája) *a tervezés és termeléselőkészítés minden eredményét* számítógéppel, *automatikusan állíthatjuk elő* (5. ábra)

Termelésirányítás

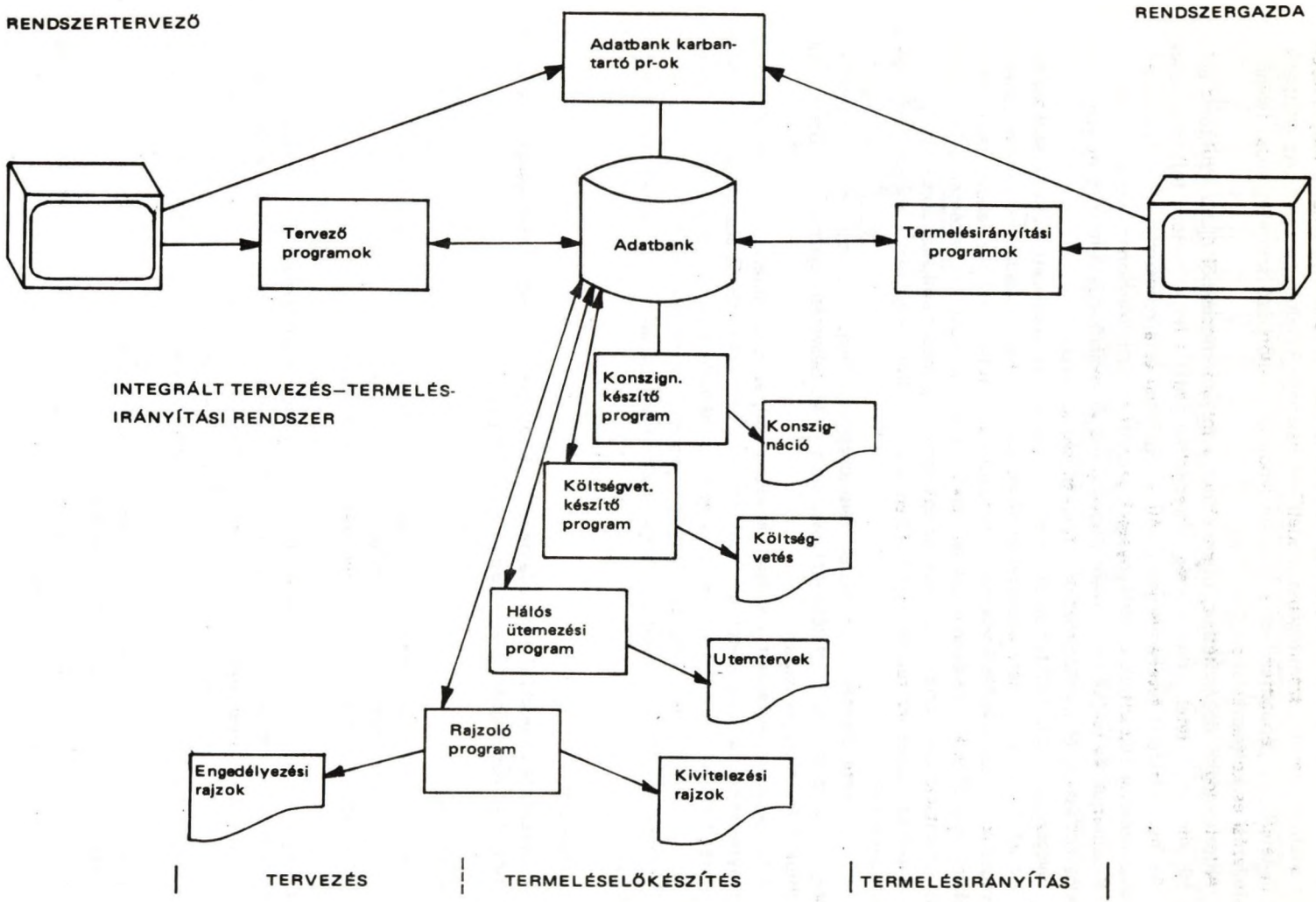
A tervezés-termeléselőkészítés adatbankban tárolt eredményeit a termelésirányítás különböző szintjeihez használjuk fel az integrált számítógépes rendszerben.

Vállalkozás

A rendszergazda vállalkozási, koordinálási feladatainak ellátásához a kiinduló adat
– az épülettervben meghatározott erőforrás-szükséglet, és
– a rendszerépítő szervezet rendelkezésére álló kapacitás.

Építési rendszerenként változhat, hogy a rendszergazda milyen mélységű kapacitás-szerződést köt a gyártókkal, kivitelezőkkel. A gyártók esetében vonatkozhat ez alrendszerekre, vagy rendszerkomponensekre, a kivitelezők esetében pedig az alrendszerekhez rendelt normatíva szintű, vagy a folyamatok szintjén mért kivitelezési kapacitásokra. A rendszergazda a vállalkozási, koordinálási feladatokhoz olyan tételekig bontja le az épület számítógépes modelljét, ahol össze tudja vetni az erőforrás-szükségletet a rendelkezésre álló kapacitásokkal. Ennek alapján kiszámítjuk a lehetséges megvalósítási határidőt és az ütemezett erőforrás-szükségletet. Az előbbit határidő ajánlatadásra, az utóbbit pedig a rendszerkomponensek és a kivitelező kapacitások biztosítására használja fel.

Az épület számítógépes modellje alapján árajánlat is készíthető – ez az épület költségvetési végösszegét jelenti, anélkül, hogy részletes költségvetést nyomtatnánk ki.


RENDSZERTERVEZŐ
RENDSZERGAZDA
Adatbank karbantartó pr-ok
Tervező programok
Adatbank
Termelésirányítási programok
INTEGRÁLT TERVEZÉS–TERMELÉS-IRÁNYÍTÁSI RENDSZER
Konszign. készítő program
Konszignáció
Költségvet. készítő program
Költségvetés
Hálós ütemezési program
Ütemtervek
Rajzoló program
Engedélyezési rajzok
Kivitelezési rajzok
TERVEZÉS
TERMELÉSELŐKÉSZÍTÉS
TERMELÉSIRÁNYÍTÁS

5. ábra

Kivitelezés

A korlátos erőforrásokkal ütemezett hálóterv, és az ütemezett erőforrás-szükséglet, a tervezés közvetlen outputjaként elkészített munkautalványok és kivitelezési szakaszok szerinti tervrajzok a kivitelezés irányításának kiinduló adatai.

Magát a termelésirányítást célszerű az integrált számítógépes rendszer részeként, az adatbankhoz kapcsolt rendszerrel végezni, de más vállalati számítógépes termelésirányítási rendszer is alkalmazható, amely kiinduló adatait — az építési rendszerben megvalósuló épületekre vonatkozóan — az ismertetett tervezés-termeléselőkészítési rendszertől kapja.

Gyártás

A rendszergazda által ütemezett rendszerkomponens szükséglet a gyártók számára a megrendelést jelenti. A gyártó vállalatok saját számítógépes termelésirányítási rendszerükkel segíthetik a gyártást, és e rendszer egyik inputja a rendszergazdától kapott megrendelés. Ha a rendszergazda is gyárt rendszerkomponenseket, számítógépes gyártásirányítási rendszerét célszerű az építési rendszer adatbankjához kapcsolni. Ilyen esetben az épületterv darabjegyzék-lebontását nem a rendszerkomponensekig végezzük, hanem azokat tovább bontjuk alkatrészekre, alapanyagokra. A gyártás műveleti adatait, valamint a munkahelyek kapacitás-adatait az építési rendszer adatbankjában tároljuk, és ezek segítségével ütemezhetjük a gyártást.

Az integrált számítógépes rendszer az építési rendszer alkalmazásának teljes folyamatát felöleli — a tervezést, vállalkozást, kivitelezést és gyártást.

3.3 A rendszerelvű építés hardware és software bázisa

Az ismertetett számítógépes rendszer hardware bázisa az ÉGSZI Siemens számítógépe. A számítógép kiépítettsége lehetővé teszi az interaktív üzemmódot, elsősorban a tervezés-termeléselőkészítés és a vállalkozás területén.

A Siemens-hez terminálok kapcsolhatók, és ezek révén megvalósítható a több szervezetnél működő integrált számítógépes információs rendszer. A megvalósítás koncepciója az, hogy mind a rendszergazda, mind a rendszertervező — de esetleg a legfőbb kivitelezők és gyártók is — terminállal csatlakoznak a központi számítógéphez, és azon keresztül érik el az építési rendszer adatbankját. Mivel a tervezés és a termelésirányítás ugyanazon adatbankra támaszkodik, amelynek adatait a rendszergazda és a rendszertervező közösen tartja karban, a számítógépes rendszer integráltsága biztosítható.

A rendszer software-je a Siemens könyvtári programrendszerein alapul. Az adatbankot az ISI adatbáziskezelő rendszerre szervezzük. Ezt használja a tervezés, és a termeléselőkészítés is. A termeléselőkészítéshez az ISI-hez kapcsoltuk a SINETIK MPM rendszerű hálótervezési programcsomagot, és ezzel végezzük az ütemezéseket.

A gyártásirányítást az ISI megfelelő moduljaival segítjük.

A rendszerépítő szervezet kapacitásainak nyilvántartását a GOLEM adatbáziskezelő rendszerre szervezzük.

Mindhárom programrendszer BS 2000-es változatban, interaktív módon, távadatfeldolgozással is működik.

EGY ÚJ PROGRAMOZÁSI SZEMLÉLETMÓD, A DINAMIKUSAN STRUKTÚRÁLT VEZÉRLÉS

Zsombok Zoltán
KSH ÁSzSz

A programozásban elterjedt vezérlési módok közül kétségtelenül az utasítások *egymásutánisága* és a különféle feltételtől függő *vezérlésátadó* utasítások képviselik a legkorábbi formákat. A ciklusszervezésre specializált, majd a *szubrutinkezelést* megvalósító utasítások jelentették a következő lépcsőt, amely logikusan vezetett el a paraméterek átadását is fejletten megoldó *eljáráskezeléshez,* amely a korai vezérlési módok – szokásos kifejezéssel a Neumann-féle gépműködés – csúcsát is jelenti. A paraméterek átvételének is számos fejlettségi szintje ismeretes. A FORTRAN által elterjesztett *érték szerinti hívást* az ALGOL a jóval dinamikusabb *név szerinti hívással* egészítette ki.

A legutóbbi évtizedben előretörő programvezérlési mód a *makro elv.* Azért sorolhatjuk a vezérlési módok közé, mert pl. egy programgenerálásra használatos makrohívás a programutasítások működése előtt fejti ki tevékenységét, mivel a makrohívások általában a programutasítások előtt hajtódnak végre, hiszen éppen azok generálására valók.

Mi az – általában csak kifejezések képzésére használt – eljáráshívást *utasítások felírására is* használni fogjuk, pl. a következő módon: Egy IF a THEN b ELSE c utasítást átírva az if (a, b, c) alakra, rögtön felismerhető, hogy az „if" nevű eljárás hívásáról van szó, amely úgy működik, hogy első paraméterének értékétől függően vagy a második, vagy a harmadik paraméterét hajtja végre.

Az eljáráshívást ezért mindenféle végrehajtandó, kiszámítandó, kiértékelendő jelölésére fogjuk használni, s megvizsgáljuk az egymásbaskatulyázott *eljáráshívások által megvalósított vezérlési módot.* Jó példa erre a tendenciára az ALGOL és elveinek újra feltalált változata, a struktúrált programozás, valamint néhány általános makronyelv. Utóbbi egy példáját lásd [1] alatt.

Az egymásba skatulyázott eljáráshívás gondolatát ki fogjuk terjeszteni a különböző fázisokban végrehajtandó műveletekre is, pl. makrokra, preprocesszor utasításokra, programfutás közben történő programtranszformációkra stb. Ily módon rendkívül *dinamikus* program-*struktúrákhoz* jutunk. Mindezen megfontolások miatt bemutatandó modellünket *dinamikusan struktúrált vezérlésnek,* rövidítve *dsc*-nek nevezzük.

Az elmélet alapgondolatainak első publikációját lásd [2] alatt, későbbi változatai előadásaim során hangzottak el, de valamennyi jóval korábbi stádiumot mutat a jelenleginél.

1. Az utasítások általános formája

Minden *hívást* – a szokásos –

$$a\ (p_1, p_2, \ldots, p_n) \qquad (1)$$

alakban írunk, ahol a a hívott eljárás, makro stb., közös szóval *algoritmus neve,* p_1, p_2, ..., p_n pedig a hívás első, második, stb. n-edik *aktuális paramétere.* (Az „aktuális" jelzőt nem mindig tesszük ki.)

Egy ugyancsak gyakori struktúrát, a *szekvenciát*

$$(p_o, p_1, ..., p_n) \qquad (2)$$

módon jelöljük.

Azt, hogy egy hívás makrora vagy eljárásra vonatkozik-e, egy k természetes számmal adjuk meg, amely egyfajta *sorrendiséget* határoz meg az ily módon megjelölt hívások között. Ezt a k természetes számot a következőképpen helyezzük el:

$$(p_o, p_1, ..., p_n)^k \qquad (3)$$

s k-t *fokszámnak* nevezzük, habár nem azonos a hatványkitevővel. Megengedjük, hogy mindhárom alakzatuk p_o, p_1, ..., p_n paraméterei egyaránt lehessenek (1), (2) vagy (3) alakúak, továbbá ún. *terminálisok*, azaz vizsgálódásunk körén kívül eső, struktúrálatlan jelsorozatok. Csak megemlítjük, hogy a terminálisok általában adatokat (számokat, szövegeket) jelölnek, de terminálisok alkotják algoritmusaink neveit is.

Az (1) és a (2) alakzatot hagyományosan értelmezzük, s ezekre vezetjük vissza (3)-at is. Előbb azonban megmutatjuk, hogy (1) és (2) a (3) forma speciális eseteként fogható fel. Ugyanis

a) A (2) alatti *szekvenciát* (3) speciális esetének tekintjük, mégpedig a k=0 esetnek, de a 0 fokszámot megállapodásszerűen nem jelöljük.

b) Az (1) alatti *algoritmushívást* (3) olyan speciális eseteként tekintjük, amelyben k=1 és p_o valamely terminális a-val azonos.

Általánossága miatt a (3) alakzatot egyszerűen *utasításnak* nevezzük, de nem felejtjük el, hogy ez az utasításfogalom magában foglalja pl. a *program* fogalmát is (mint egy jócskán összetett utasítás), de a kifejezések, műveletek, makrohívások, deklarációk stb. fogalmát is. Ezért, ha utasítás végrehajtásáról beszélünk, akkor mondhatnánk kiszámítást, kiértékelést stb. is. E fogalmakat egymással egyenértékűeknek tekintjük.

A következőkben arra fogunk választ adni, hogy miképpen hajtódik végre egy $(p_o, p_1, ..., p_n)^k$ utasítás, ahol k nemnegatív egész, és a p_o, p_1, ..., p_n paraméterek vagy utasítások vagy terminálisok.

2. Utasítások alapvető végrehajtási szabályai

Mindenek előtt felhívjuk a figyelmet egy igen hasznos és elterjedt elvre. Ebben a vonatkozásban egy makrohívás pontosan ugyanúgy viselkedik, mint egy aritmetikai kifejezés. Nevezetesen, ha egy aritmetikai kifejezés egyik komponense egy függvénykifejezés vagy akár csak egy művelet, akkor mindig előbb kiszámítjuk a függvénykifejezés, ill. a művelet eredményét, s a kapott értéket a függvénykifejezés, ill. a művelet helyére téve mehetünk tovább a számításban. Egy makrohívás végrehajtása (kifejtése) során kapott szöveget a makrohívás helyére tesszük. Mindkét esetben ugyanaz történt, egy hívás során kapott *értéket a hívás helyére* tettük. Az ily módon viselkedő programrészleteket *értékkifejezéseknek* szokás nevezni. Az értékkifejezésekből álló programok végrehajtása általában programtranszformációt okoz, ami jól használható pl. programgenerálásra. Mi pedig éppen azt szeretnénk elérni, hogy modellünk aknázza ki ezt a lehetőséget is, ezért kimondjuk, hogy a *dsc minden utasítása egyben értékkifejezés* is, vagyis minden utasításnak a következőképpen kell végrehajtódnia:

– először meghatározott sorrendben végrehajtunk bizonyos komponenseket (rendszerint paramétereket),

– végezetül kiindulásunk helyére valamilyen értéket helyettesítünk, aminek utasításnak vagy terminálisnak kell lennie.

Ezt a követelményt *általános végrehajtási elvnek* nevezzük, s az alábbiakban bemutatott összes szabályunknak ebben a szellemben kell eljárnia.

Ezen előkészítés után a legalapvetőbb szabályok:

1. szabály: Ha p terminális, akkor végrehajtása során semmi sem változik, értéke önmaga.

2. szabály: Ha a $(p_o, p_1, ..., p_n)^k$ utasításban k=1, és p_o végrehajtása során valamely a $(p_1, ..., p_n)$ hívás az a algoritmus szabályai szerint.

3. szabály: Ha a $(p_o, p_1, ..., p_n)^k$ utasításban nem áll fenn egyszerre, hogy k=1 és a p_o terminális, akkor végre kell hajtani ebben a sorrendben a $p_1, ..., p_n$ paramétereket is, s ezek $p'_1, ..., p'_n$ értékeiből kapjuk utasításunk értékét: $(p'_o, p'_1, ..., p'_n)^{k \div 1}$ (Megjegyezzük, hogy a $\div$ műveleti jel szokásos jelentése: $k \div 1 = k-1$, ha $k > 0$, de $0 \div 1 = 0$.)

Vegyük észre, hogy a 3. szabály voltaképpen három esetre vonatkozik: a k=0 szekvenciára, a k=1 és "p'_o nem terminális" esetre, továbbá a $k > 1$ utasításokra. Figyeljük meg azt is, hogy egy utasítás végrehajtása – a 3. szabály szerint – általában paramétereinek végrehajtását is megkívánja. A három szabály hatását egy programra így mondhatjuk el:

Megkeressük a programban az első olyan részletet, amely (1) vagy azzal ekvivalens alakú (azaz k=1 és p_o terminális). Ezt a hívást – egyelőre nem ismert módon – végrehajtjuk. Ezután két eset lehetséges: a) Ha végrehajtott hívásunk egy másik elsőfokú hívás p_o paramétere volt, s eredménye terminális lett, akkor ezen újabb hívás következik végrehajtásra az előbbi módon. b) Ha a feltétel nem áll fenn, akkor a végrehajtott hívást követő programrészben kell keresni (1) vagy azzal ekvivalens alakú hívást. Minden olyan utasítás, amely nincs az előbbi módon végrehajtott hívások paramétereiben és fokszáma nem zérus, eggyel kisebb fokszámú lesz.

Ily módon először csak az elsőfokú utasítások eredményezhetnek algoritmus végrehajtást, de ha az egész program értékét is végrehajtjuk, akkor már a másodfokúak következnek, mivel azokból az előbb elsőfokú lett, és így tovább. Ebből már érzékelhető a dsc makrokezelési technikája.

3. Az algoritmusok hívásának és a paraméterek átvételének szabályai, a primitívek

A 2. szabályban jeleztük, hogy az algoritmusok végrehajtásának vannak szabályaik. A következőkben ezekkel fogunk foglalkozni.

Az algoritmusok két csoportba sorolhatók. Az egyik csoportot azok alkotják, amelyek működése leírható a már tárgyalt utasítás végrehajtási szabályokkal, míg a másik csoportot azok, amelyekkel ez nem tehető. Az első csoport tagjait *definiált algoritmusoknak*, a többit *primitíveknek* nevezzük. A primitívek voltaképpen az egyes programnyelvek speciális utasításai, (mint pl. a bevezetőbeli „if",) valamint az ún. *standard eljárások.*

Az egyes primitíveknek módjukban áll tetszőleges szabályok szerint működni, mindössze azt várjuk el, hogy paramétereik felhasználása és a hivás értékének vonatkozásában feleljenek meg az előző pontban kifejtett általános végrehajtási elvnek.

Az ún. definiált algoritmusokat egyetlen utasítással határozzuk meg, pontosabban azt mondjuk, hogy a *definiált algoritmusok törzse egyetlen dsc-utasítás.* A törzs belsejében az aktuális paraméterek átvételére ún. *formális paramétereket* alkalmazunk, nevezetesen olyan φ (p) alakú hívásokat, amelyek az eljáráshíváshoz meglepő hasonlatosságokat mutatnak. φ egy terminálist jelöl, a paraméterek átvételére szolgáló algoritmus *nevét,* paraméterének pedig olyan utasításnak kell lennie, amely – végrehajtása során – egy i természetes számot ad értékként.

E jelölésekkel a kövatkező szabályokat írhatjuk fel:

4. szabály: Ha az a algoritmus törzse a b utasítás, akkor az a $(p_1, ..., p_n)$ *hívás* úgy hajtódik végre, hogy végrehajtjuk a b törzset, s ennek értékével helyettesítjük a hívást.

5. szabály: Egy φ (p) *formális paramétert* p végrehajtásával kezdünk végrehajtani, amelynek egy i természetes számot kell eredményeznie. Ezután végrehajtjuk annak a hívásnak az i-edik aktuális paraméterét, amelyik által hívott algoritmus törzsében van ez a formális paraméter. Az i-edik aktuális paraméter értéke lesz a formális paraméter értéke.

A bemutatott öt végrehajtási szabály alapján most már minden olyan utasítást végre tudunk hajtani, amely ismert algoritmusokat hív. Legyen pl. az a algoritmus törzse a $(\varphi(2), \varphi(1))^2$ utasítás, és vizsgáljuk meg, hogy hajtódik végre az a (b,c) utasítás.

a (b,c) végrehajtását — a 4. szabály alapján — törzsének végrehajtásával kell elvégezni. A törzs $(\varphi(2), \varphi(1))^2$ a 3. szabály szerint φ (2) és φ (1) végrehajtásával, majd a fokszám 1-gyel való csökkentésével intézendő el. Mivel φ (2) és φ (1) az 5. szabály szerint c és b, ezért a törzsének értéke $(c,b)^1$ lesz, egyszerűsített írásmóddal c(b). Vagyis az a(b,c) hívás értéke a c(b) utasítás. Vegyük észre, hogy a végeredmény nem a c(b) hívás értéke, hanem a c(b) hívás maga!

4. A végrehajtási mechanizmus formális jelölése

Mivel az előbbi gondolatsort a sok egymásba skatulyázás miatt elég nehéz fejben követni, bevezetünk egy megfelelő formalizmust.

Ha X végrehajtása során az y értéket kapjuk, akkor ezt így jelöljük:

$$x \to y$$

s azt mondjuk, hogy x *értéke* y. Ha x-et úgy kell végrehajtani, hogy közben az $x_1, x_2, ..., x_n$ utasításokat kell (pontosan ebben a sorrendben) végrehajtani, s ezek értékei legyenek rendre $y_1, y_2, ..., y_n$, amelyek felhasználásával végül is y-t kapjuk értékként, akkor ezt így jelöljük:

$$x \Rightarrow [x_1 \to y_1, x_2 \to y_2, ..., x_n \to y_n, y] \qquad (4)$$

E formulát x *végrehajtásának* nevezzük, s gyakran $x \to y$ módon fogjuk rövidíteni.

E formalizmussal előbbi feladatunk megoldása így fest:

$$a(b,c) \Rightarrow [\ (2) \to c, \varphi(1) \to b, c(b)]$$

Végrehajtási szabályaink is pontosíthatók. Példaképpen a 3. szabályt mutatjuk be. Ha k = 1 vagy p'_o nem terminális, akkor

$$(p_o, p_1, ..., p_n)^k \to [p_o \to p'_o, p_1 \to p'_1, ..., p_n \to p'_n, (p'_o, p'_1, ..., p'_n)^{k \to 1}]$$

5. Algoritmusok jelölése, lokális algoritmusok

Az a nevű algoritmust, melynek törzse b, így jelöljük: (a:b), amelyet sok esetben csak az algoritmus nevének feltüntetésével rövidítünk: a.

Az egymásba skatulyázott eljárás hívásokban való gondolkodás egyenes velejárója a blokkstruktúra is, ezért mi is megengedjük az algoritmusok egymásba skatulyázását. Ha pl. egy a algoritmus b törzsének végrehajtása tartalmára szükség van egy b törzsű a_1 és egy b_2 törzsű a_2 algoritmusra, akkor az a algoritmust így írjuk:

$$(a: (a_1 : b_1), (a_2 : b_2) \setminus b) \tag{5}$$

Az ilyen kapcsolatban lévő algoritmusokra azt mondjuk, hogy a *logikálisai* a_1 és a_2. A \ *jelet* arra a célra vezetjük be, hogy *egy, a jobboldalán álló utasítást az utasítás által elérhető algoritmus struktúrával kapcsoljon össze.* Jelen esetben a \ jel a lokális algoritmusok és a törzs közé kerül, jelezve, hogy a lokális algoritmusok elérhetők a törzsben. Azt, hogy a lokálisok kívülről nem érhetők el, következő megállapodásunk, ill. jelölésünk biztosítja: Egy algoritmus struktúrában mindazok az *algoritmusok érhetők el, amelyek nincsenek más algoritmusba zárva, de maguk zártak. A zártság a végzárójel meglétét jelenti,* ezért ha éppen egy algoritmus törzsét (pl. a fenti b-t) hajtjuk végre, akkor az a algoritmust a \ jeltől jobbra eső rész elhagyásával „megnyitjuk", s az így kapott algoritmus struktúrában a_1 és a_2 már elérhető:

$$(a : (a_1 : b_1), (a_2 : b_2) \tag{6}$$

Gondoljuk meg még, hogy a törzs végrehajtásakor nemcsak a lokális algoritmusok válnak elérhetővé, hanem az aktuális paraméterek is. S mivel ezt az elérést a φ nevű formális paraméter valósítja meg, az aktuális paramétereket célszerű egy φ nevű algoritmus törzseként feltüntetni, s híváskor a lokálisok mellé írni:

$$(a : (a_1 : b_1), (a_2 : b_2), (\varphi : p_1, \ldots, p_n) \tag{7}$$

Ezzel az aktuális paramétereket – ahogy jeleztük – rendkívül hasonlatosan lehet kezelni az algoritmusokkal.

A hozzáférés összefoglalásául vizsgáljuk meg a következő algoritmus struktúrát: $(a : (a_1 : (a_2 : b_2) \setminus b_1), (a_3 : (a_4 : b_4), (\varphi : p_1)$ Jelöléseink értelmében, ha ezt egy \ p jelsorozat követné, akkor azt mondhatnók, hogy p számára elérhetők az a_1, a_4 algoritmusok és a p_1 aktuális paraméter, de nem érhető el a, a_2 és a_3.

Most már formalizálva is felírhatjuk a 4. és az 5. szabályt:

4. szabály: Ha az a algoritmus $(a : a_1, \ldots, a_m \setminus b)$ alakú, akkor:

$$a\,(p_1, \ldots, p_n) \rightarrow$$
$$\rightarrow [(a : a_1, \ldots, a_m, (\varphi : p_1, \ldots, p_n) \setminus b \rightarrow (a : a_1, \ldots, a_m, (\varphi : p_1, \ldots, p_n) \setminus b', b']$$

5. szabály: $(a:a_1, \ldots, a_m, (\ : p_1, \ldots, p_n) \setminus (p) \rightarrow$
$$\rightarrow [(a:a_1, \ldots, a_m, (\varphi : p_1, \ldots, p_n) \setminus p \rightarrow (a:a_1, \ldots, a_m, (\varphi : p_1, \ldots, p_n) \setminus i,$$
$$p_i \rightarrow p'_i, (a:a_1, \ldots, a_m, (\varphi : p_1, \ldots, p_n) \setminus p'_i\ ]$$

Láthatóan a 4. pontban bevezetett formalizmus alkalmas a szabályok leírására. Figyeljük meg, hogy a lokális algoritmusok és az aktuális paraméterek elérhetők a törzsben, s így a formális paraméter számára is, de elérhetetlenek az aktuális paraméterek végrehajtásakor, a törzs végrehajtása előtt és után.

6. Algoritmusok létrehozása, újradefiniálás

Az eddigiekből azt lehetne hinni, hogy az algoritmusokat egyszerűen úgy hozzuk létre, hogy a bemutatott jelöléssel leírjuk őket. Így egy \ jel baloldalán lesznek az algoritmus deklarációk, a jobboldalán pedig az azokat felhasználó utasítások. Mi azonban az egyszerű deklarációnál jóval dinamikusabb módon fogjuk algoritmusainkat definiálni.

Vessünk be egy speciális algoritmust, amit δ-val jelölünk és *definíciónak* nevezünk, s amelynek $\delta(p_1, p_2)$ hívása a következőképpen hajtódik végre: Elvégezzük a $p_1 \rightarrow a$ és a $p_2 \rightarrow b$ végrehajtásokat, s értékükből felépítjük az (a : b) algoritmust. Azáltal, hogy a definíció paramétereit végrehajtjuk, a definíció *folyamat lesz,* amelynek többek között meghatározott kezdete és vége van. Magának a definíciós utasításnak nincs értéke, helye üres lesz.

A definiálási folyamatot abban a pillanatban tekintjük *megkezdettnek, amint befejeződött az első paraméter végrehajtása.* A folyamat a második paraméter végrehajtásával együtt fejeződik be. Azért, hogy a definíció az algoritmust az 5. pontban bevezetett jelöléssel állítsa elő, megállapodunk abban, hogy a folyamat kezdetén leírjuk a „ ‚(a:" jelsorozatot, amit a folyamat végén megtoldunk a „\b)" jelsorozattal. A két jelsorozat együtt: ", (a:\b)", ami egy feleslegesnek látszó vesszőtől és egy \ jeltől eltekintve valóban korábbi jelöléseiknek felel meg. A két feleslegesnek látszó jel haszna a következő példában derül ki.

Hajtsuk végre a δ (a,(δ (a_1, b_1), b,δ (a_2, b_2))) utasítást! Az egyszerűség kedvéért legyen a, a_1, a_2, b, b_1, b_2 mind terminális. Figyeljük meg, hogy a definíció második paramétere egy szekvencia, amelynek első és harmadik paramétere újabb definíciók, a második paramétere pedig b.

Az a terminális egyszerűen értékelődik ki, mert értéke önmaga, s máris elkezdődött a definíciós folyamata. Leírjuk: ", (a: ". A szekvencia végrehajtása következik. Első eleme egy definíció, amelynek eredménye, mint láttuk: ", (a_1: \ b_1)" és a definíció értéke üres. A szekvencia második paraméterének értéke b, ami azonban még nem az algoritmus törzse, így nem írhatjuk le, ugyanis a szekvenciának még egy paramétere van, egy újabb definíció, amelynek értéke üres, eredménye: ", (a_2 : \ b_2)". Ezzel befejeződött a szekvencia végrehajtása, értéke "(‚b,)" lett, vagyis egy szekvencia két üres paraméterrel. Most már leírhatjuk definíciós folyamatunk végét: "\(‚b,))". Utasításunk végső értéke természetesen üres, de eredménye a jelsorozatok egybeírásával:

$$‚(a :, (a_1 :\ b_1), (a_2 :\ b_2) \ (‚b,))$$

amely egy a nevű algoritmus (‚b„) törzzsel és két lokális algoritmussal. Most már az is világos, hogy a feleslegesnek látszó vessző az előzőleg befejezett algoritmustól való elkülönítésre szolgál, míg a \ jel is hasznos, ha vannak lokális algoritmusok is.

Az azonos névvel definiált algoritmusok közül mindig csak a később definiált érhető el, vagyis az, amely az elérhető algoritmusok sorában jobbra áll. Ez a szokásos *újradefiniálás.*

7. Hatáskörök és környezet, a definiálás formális szabálya

Tekintsük át, hogy egy-egy algoritmus elérhetősége szempontjából mely mozzanatokat kell számításba vennünk!

a) A 4. és az 5. szabálynak megfelelően egy b törzsben lévő hívás számára elérhetőek annak az algoritmusnak a *lokálisai,* amelynek törzse b.

b) Egy a algoritmus törzsében elérhetők mindazon *globális* algoritmusok is, amelyek elérhetők annak az algoritmusnak a törzsében, amelynek a lokálisa.

c) Előrhető egy algoritmus, ha az őt létrehozó *definíció már befejeződött,* de egy őt lokálisává tévő algoritmus definíciója még nem fejeződött be.

a)-t és b)-t az ALGOL-ból vettük át, de c)-nek még az igénye sem merül fel a hagyományos programnyelvekben, hiszen a definíciók ott nem folyamatszerűek. c)-vel elértük, hogy a lokális algoritmusok ne csak a törzs végrehajtása, hanem létrehozása során is használhatók legyenek. c)-t sajnos, már nem tudjuk leírni az ALGOL hatáskör módszerével, t.i. megadva azt, hogy egy-egy algoritmus a program mely területein használható. Helyette bevezetjük a *környezet fogalmát: A vezérlés valamely pillanatában az éppen végrehajtás előtt álló utasítás számára elérhető algoritmusokat hívjuk környezetnek.*

A környezetet előbbi algoritmus struktúráink formájában írjuk, de a módszer még nem problémamentes. Alkalmazzuk ugyanis eléggé mechanikusan eddigi módszerünket a következő feladatra!

Hajtsuk végre az a(δ(b,c)) utasítást, feltéve, hogy a ilyen: (a:d \ φ (1)), vagyis van egy d lokálisa is és törzse egyetlen formális paraméterből áll.

A 4. szabály szerint a hivásakor elérhetővé kell tennünk a lokálisokat és az aktuális paramétereket. A φ (1) törzset ebben a környezetben kell végrehajtani. Így végrehajtandó (a:d, (φ : δ (b,c)) \ φ (1). amely egy definiálást fog eredményezni, s módszerünk szerint azt a környezet folytatásaként kell leírni: (a:d, (φ : δ (b,c)), (b:c) A baj akkor kezdődik, amikor a törzs végrehajtását befejezzük, mivel elérhetetlenekké kell tennünk az aktuális paramétereket és a lokális algoritmusokat, amit azzal érünk el, hogy a környezet végén lévő megnyitott algoritmust eltüntetjük. Ekkor azonban eltűnik frissen létrehozott b algoritmusunk is, holott c)-ben azt igértük, hogy használható lesz.

A problémát úgy oldjuk meg, hogy két komponensből tesszük össze a környezetet. Az egyik (jelöljük f-fel) fogja tartalmazni a felépített és az építés alatt lévő algoritmusokat, c)-nek eleget téve.

A másik részt, amelyet g-vel jelölünk, fogják azok az algoritmusok alkotni, amelyek egy-egy hívás során válnak elérhetővé, vagyis az a) és a b) szerint. Igy a 4. és az 5. szabályt úgy kell értelmezni, hogy ott, ami környezetként szerepel, az a g rész. A két környezetfelet pontosvesszővel választjuk el egymástól, s így azt, hogy egy p utasítást az f, ill. g környezetfélben hajtódik végre, így jelöljük: f; g \ p

Ezt az alakzatot *kifejezésnek* nevezzük, s sejteni lehet, hogy a dsc éppen ilyen kifejezések transzformációjának rendszerét írja le.

Utolsó szabályként a definíció működését mutatjuk be.

6. szabály:

$$f ; g \setminus \delta\ (p_1, p_2) \rightarrow [f ; g \setminus p_1 \rightarrow f'; g \setminus a,$$
$$f', (a : ; g \setminus p_2 \rightarrow f', (a:a_1, ..., a_m; g \setminus b,$$
$$f', (a : a_1, ..., a_m \setminus b) ; g \setminus \lambda\ ]$$

A szabályt tulajdonképpen már ismertettük, most csak a jelöléssel kapcsolatban hívjuk fel a figyelmet egy s másra. f'-t azért különböztettük meg f-től, mert p_1 végrehajtása közben – esetleges belső definíciók miatt – megváltozhat a környezet, még mielőtt a definíciós folyamat elindult volna. Ugyanakkor, ha p_2 végrehajtása közben jönnek létre pl. a_1, ..., a_m algoritmusok, akkor ezek már az a algoritmus lokálisai lesznek, mivel a definíció már elkezdődött. λ-val jelöltük, hogy a definíciós utasítás helyettesítési értéke üres.

Figyeljük meg azt is, hogy az f ; g környezet g fele nem változott. E környezetféllel a 4. és az 5. szabály foglalkozik, de g a szabály végén is ugyanaz, mint az elején. Általis igaz:

$$\text{Ha } f\ ;\ g \setminus p \rightarrow f'\ ;\ g' \setminus p'.$$

Befejezésül bemutatunk egy, a programok optimalizálásának tárgykörébe eső példát. gáljuk meg, mit eredményez a

$$\delta\ (h,\ f\ ((g,\ x^2)^2))$$

definíció, ahol f és g két függvény, x-szel pedig a (φ ,1) utasítást jelöltük (kizárólag az kinthetőség érdekében).

A második paraméter értéke lesz a törzs. f ((g, x^2)^2) végrehajtásához végre kell h nunk az f algoritmusát a (g, x^2)^2 paraméter értékével, azaz g(x)-szel. Vigyázzunk, mert g(x) értékéről van szó, hanem a g függvényről magáról, az x szabad változóval! Tehát á níció egy olyan (h:$f_o g(x)$) algoritmust hoz létre, amelyben két függvényt találunk egyes egy új függvénnyé, amit fog-vel jelöltünk.

E probléma már messze vezet mondanivalónktól. J. Backus foglalkozik egy cikkéb ilyen kérdésekkel, programoknak bizonyos algebrai szabályokkal való kezelésével. Elkép *funkcionális programozásnak* nevezi. A.P. Ershov *kevert programozásnak* nevezi azt a le get, amikor egy programot a nem változó paraméterekkel részben végrehajtva olyan úja ramot kapunk, amely azon áron, hogy kevésbé rugalmasabb, optimálisabb. [4]

* * *

Nem lehetett célunk az elmélet teljes pontossággal történő ismertetése, ezért inká áttekintést adtunk tárgyáról, gondolatairól, hatóköréről és módszeréről. Láthattuk, hogy egységbe foglalja az algoritmikus és a makronyelvek lényeges struktúrális tulajdonságait, makro-koncepció egy általánosítását is adja. Az elmélet esetleges felhasználási lehetőség

— a bemutatott formális eszközökkel a szabályok teljes mértékben axiomatizálhat így segítségével tisztázhatók a programozás egyes kérdései;

— a leírt modell magja lehet valamely, rendszerfejlesztésre használható hardware-s eszköznek, ahol a nagyfokú dinamizmus jól kamatoztatható lehet.

Irodalom

[1] P. Lindblad: IMP IV (Interpretative Macro Processor) Kobenhavns Universitets Mat Institut, 1971. K.U.A.I.B. 23.

[2] Zsombok Z.: Program-dinamika, Információ Elektronika, 1976. No 2.

[3] J. Backus: Can Programming Be Liberated from the von Neumann Style? A Funct Style and Its Algebra of Programs, CACM V 21, No 8.

[4] A.P. Ershov: On the Essence of Compilation, 1977, kézirat.

I–II. kötet ára: 150.– Ft