MŰSZAKI ÉS TERMÉSZETTUDOMÁNYI EGYESÜLETEK SZÖVETSÉGE
NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG

I. ORSZÁGOS KONGRESSZUS ELŐADÁSOK

I.

Szeged
1979. december 3–7.

I. ORSZÁGOS KONGRESSZUS

ELŐADÁSOK

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

NÉHÁNY MAGYARORSZÁGI KUTATÁS A SZÁMÍTÁSTUDOMÁNY MATEMATIKAI ALAPJAI TERÉN

Andréka Hajnal – Németi István
MTA Matematikai Kutatóintézet

A fenti témán belül elsősorban programozáselmélettel, annak *szemantikai* (jelentéstani) kérdéseivel, a programhelyesség-bizonyítás, programspecifikálás (feladatspecifikálás) elméletével, általában a számítástudományban megjelenő nyelvi jelenségek szemantikai – tartalmi oldalával foglalkozunk*. Az ismertetésre kerülő hazai kutatások világirodalmi hátteréről az 1. pontban szólunk. Az 1. pont egyben a fent nevezett téma népszerűsítő jellegű körülírását – ismertetését is tartalmazza.

1. Világirodalmi háttér vázlatos áttekintése

Az elmúlt évtizedben a számítástudomány, pontosabban a programozáselmélet kutatásában a *szemantikai* vizsgálatok kerültek előtérbe. Többek között az ún. „software válság", a „nagyüzemi programgyártás elméleti megalapozása", konkrétabban pl. a „programhelyesség-bizonyítás" tették a szemantikai vizsgálatokat aktuálissá. A programozáselmélethez kapcsolódó szemantikai kutatások szorosan kapcsolódnak a matematikai nyelvészet azon új ágához, amely a nyelvi jelenség szemantikai oldaláról kíván explicit és egzakt (matematikai) módon számot adni. Az explicitség igényét egyelőre csak a klasszikus logika *modellelmélete*, valamint az azzal analóg nemklasszikus modelleméletek ill. *denotációs* szemantikák elégítik ki.

A vizsgálandó újszerű jelenségkör (nyelvek szemantikája) és újszerű igények (explicitség) meglehetősen határozott követelményeket támasztanak az alkalmazandó matematikai apparátussal szemben. Az elmúlt 7 év szakirodalma azt mutatja, hogy igen sok kutató meggyőződése szerint a szemantikaelmélet matematikai alapjai a Kategóriaelmélet, Univerzális Algebra és Modellemélet ötvözése útján hozhatók létre.

A szemantikaelmélet fejlődésének rövid áttekintése:

A programozáselmélet, ezen belül a programozási nyelvek szemantikája a számítástudománynak egy olyan új ága, amely igen gyorsan központi jelentőségűvé vált. A területtel 1962–65-ben kezdtek behatóan foglalkozni (McCarthy, Naur, Ianov). Matematikai eszköztára 1969–70-ben kezdett körvonalazódni (Burstall-Landin, Scott-Strachey, Manna, Thatcher, stb.). Ekkor már látszott, hogy a „software" addigi matematikai elméletén, a tisztán szintaktikai matematikai nyelvészeten (Chomski) és algoritmuselméleten kívül itt még szükség van olyan matematikai eszközökre, melyeket a Kategóriaelmélet, Univerzális Algebra és a Modellelmélet (logikai szemantikája) területén lehet megtalálni. (Elősegítette ezt a felismerést az is, hogy a matematikai nyelvészet már korábban elkezdett szemantikával foglalkozni, és ennek során Richard Montague kimutatta, hogy modellelméletre van szükség, melyet azután sikeresen alkalmazott is.)

1969 után, amikor a terület igényesebb, matematikailag megalapozott művelése megkezdődött, a kutatás „divergálni" kezdett: az egyes kutatók más és más oldalról közelítették meg a kérdést, másképp formalizálták az alapfogalmakat (pl. program fogalma, program he-

* A „mi" itt egy meglehetősen „fuzzy" halmazt jelöl. Azokra a kutatókra gondolunk, akikkel a szóbanforgó témában a jelen szerzők együttműködnek.

lyességének fogalma, stb.). Scott folytonos hálói, Burstall-Landin univerzális algebrái, Manna-Emden modellelmélete (és interpretált programsémái), Goguen kategóriái, stb. nem hasonlítottak egymásra.

A divergálást később (1975–76 után) *konvergálás* váltotta fel, ami annak a jele, hogy valamilyen lényeget sikerült megfogni, és a különbözőnek indult kutatások eköré „konvergálnak": Például a „Fundamentals of Computer Science Theory Poznan 1977" kötet, a „Seminaire sur la sémantique dans l' informatique á Sophia-Antipolis 1977" anyagai, a Lecture Notes in Computer Science sorozat 1977 utáni kötetei (Springer), és a hasonló tárgyú újabb keletű konferencia-anyagok, „Research Report"-ok meglepően *egységes képet* mutatnak, és a matematikának egy olyan új (és már nem naív) „fejezetére" támaszkodik mindegyik, mely a Kategóriaelmélethez, Univerzális Algebrához és Modellelmélethez tartozik (mindháromhoz). Ez az új „fejezet" azonban már a Számítástudomány hatására jött létre a matematikán belül, és így önmagában is kerek része a fent nevezett három területnek.

Az univerzális algebra alapkönyve, G. Grätzer: Universal Algebra (second edition) Springer Verlag 1979 az 57. fejezetben (338. old.) tér ki a fent vázolt területre. (Az algebrai szemantika matematikai alapjairól ld. fenti könyv 378–381 old.)

2. A szemantikaelmélet Univerzális Algebrai eszközei, módszerei

Programozási és ehhez kapcsolódó nyelvek algebrai szemantikájával a nemzetközi Számítástudomány igen kiterjedten és mélyen foglalkozik. Részletes áttekintés található [18]-ban és [38]-ban. Burstall és Goguen legújabb munkái [25] alapján úgy tűnik, hogy az Algebrai Szemantika lesz az a közelítésmód, amelynek keretein belül megvalósítható lesz a *Struktúrált Program Specifikálás* „nyelvhierarchiáinak" [29] elmélete és módszertana is.

A programozási nyelvek algebrai szemantikájának kutatása során világossá vált, hogy először meg kell érteni *általában a nyelvek* algebrai szemantikáját, azaz ki kell dolgozni az algebrai szemantika általános elméletét. (Ez analóg azzal, hogy a programozási nyelvek szintaxisának vizsgálatához létre kellett hozni a szintaxis általános elméletét, amit ma legtöbbször Matematikai Nyelvészet névvel illetnek.) Szükség volt erre azért is, mert a Számítástechnikában nemcsak programozási nyelvek használatára van szükség, hanem más nyelvekre is, ld. pl. [37] vagy [14] bevezetését, vagy az MTA SZTAKI-ban kifejlesztett Struktúra Logika (SL) irodalmát, pl. [28].

Az Algebrai Szemantikának ma látszólag több különböző iskolája van, pl. az (1) ADJ csoport (IBM és UCLA) [38], az (2) Andréka-Gergely-Németi (a továbbiakban AGN csoport) [3, 4, 10, 15, 19]-ben elindított irányzat, a (3) Montague féle Univerzális Grammatika [40, 18] stb. A legismertebb ezek között az ADJ iskola, lényegében ennek változatai a Lawvere féle algebrai elméleteket módosító Elgot féle, és a francia „Szabad Magma" kutatások. [46]-ban bemutattuk, hogy a fent idézett (1), (2) és (3) iskola tulajdonképpen azonos. A három iskola egyesítésével ki tudtuk használni azt, hogy mindegyik más részleteket, más vonatkozásokat dolgozott ki jobban, melyek most együttesen alkalmazhatók már. Például az ADJ iskola nem dolgozott ki módszereket arra, hogy az algebrai vizsgálatok eredményei alkalmazhatók legyenek az eredeti szituációban, melyből az algebraizáláskor (absztraháláskor) kiindultunk. Annál részletesebben foglalkoznak ilyen módszerek, kritériumok keresésével az AGN dolgozatok (ld. pl. 13. old. és 18–20. old. [46]-ban).

Az Algebrai Szemantika egyik alkalmazási területe a *Struktúrált Programspecifikálás,* ld. pl.a Burstall-Goguen munkákat [25, 38]-ban 3.3 és 4.2 fejezet; a Dávid Gábor és munkatársaitól származó SL-t [28], stb. [46]-ban a 3. definícióval és az (i)-(iv) tételek részletes bizonyításával azt szerettük volna kezdeményezni, hogy a Struktúrált Programspecifikálás Burstall-Goguen féle elmélete általánosítható legyen az AGN munkákban vizsgált nagyobb

kifejezőerejű nyelvekre is. Ehhez [46] 3. definíciója általánosítja az AGN munkákban definiált Lv algebraosztályt úgy, hogy az új Lvg algebraosztály egy eleme egy elsőrendű elméletnek (vagy, ami evvel ekvivalens, modellosztálynak) feleljen meg. Így az AGN eredmények lehetővé fogják tenni, hogy az ADJ által használt „Algebrai Elméleteken" túlmenően tetszőleges elsőrendű elméletek egymásba való interpretálása útján nyerhető kategóriát (Nyelvhierarchiát, ld. [2]) vizsgáljuk és használjuk a Struktúrált Programspecifikálásban. A (iii) és (iv) tétel részletes bizonyításával [46]-ban egyben módszert is kívántunk javasolni a fent idézett AGN programhoz.

Az alábbiakban az „elmélet" és a „nyelv" szavakat szinonímákként használjuk. (A két fogalom tényleg összefügg: nincs elmélet nyelv nélkül, és ha jobban meggondoljuk, ez fordítva is igaz.) Ennek a szóhasználatnak az az oka, hogy az első *nyelvhierarchiákat* Lawvere *„elmélet-kategóriáknak"* nevezte el (algebrai elméleteknek nevezte a kategória objektumait). [46]-ban a (iii) tétel azt bizonyítja, hogy az Lvg algebraosztály elemei pontosan az elsőrendű elméletek. Azaz, minden elsőrendű elmélet megfelel egy Lvg-beli algebrának és megfordítva. Definiáltuk a bázishomomorfizmus fogalmát, úgy, hogy ezek speciális homomorfizmusok az Lvg-beli algebrák között. Kiderül, hogy a bázishomomorfizmus pontosan megfelel az elméletek (ill. nyelvek) közötti interpretáció fogalmának [46, 50], melyet Burstall–Goguen [25] elméletmorfizmusnak nevez. Így tehát Lvg egy kategóriává (naívan fogalmazva: hierarchiává) válik, melynek objektumai elméletek ill. nyelvek, és morfizmusai (a hierarchiabeli kapcsolatok) az elméletek ill. nyelvek közötti interpretálások azaz fordítófüggvények. Az Lvg kategória tehát a korábban oly sokszor emlegetett nyelvhierarchiának [2, 25, 29, 33, 34, 42] egy precíz, matematikailag kezelhető „modellje", mely a bázishomomorfizmusok révén a szemantikai kapcsolatokat is tartalmazza. Ilyen modellt nyújt [25] is, melynek ereje abban rejlik, hogy ún. „Elmélet Eljárásokat" (Theory Procedure) tudnak definiálni és kezelni benne. Ez azért sikerült nekik, mert kategóriájuk teljes és koteljes. Ugyanezzel a tulajdonsággal az általunk javasolt Lvg kategória is rendelkezik, tehát a „Burstall-Goguen program" itt is végrehajtható. Az Lvg előnye, hogy a benne objektumokként szereplő nyelvek kifejezőerejére nincs *semmi* olyanfajta korlátozás, mint Burstall-Goguennél (vagy bárki másnál, aki a Lawvere-féle algebrai elméletek kategóriáját használja nyelvhierarchiának).

Összefoglalva: Ismeretes, hogy a Kategóriaelmélet Lawvere-féle ágát („algebrai elméletek") elég kiterjedten alkalmazzák a számítástudományban bizonyos célok elérésére [25, 38, 18]. A fentiekben vázoltuk, hogy ezek a célok elérhetők úgy is, hogy kategóriaelmélet helyett *algebrai logikát* használunk, ami lényegében azt jelenti, hogy a *Cilindrikus Algebrák* elméletét használjuk. Az egymástól eltérő közelítésmódok nyilván kiegészítik egymást, alkalmas kombinálásuk új lehetőségeket nyújthat.

3.Különböző Nyelvtanokkal megadott nyelvek Algebrai Szemantikája

A vizsgálatok során kiderült, hogy az, hogy a nyelv szintaxisát milyen nyelvtannal adják meg, az a szemantikai vizsgálatok során (a szemantika kezelhetőségének szempontjából) egyáltalán nem közömbös. Ez explicitté az algebraizálás során válik, különösen akkor, ha az algebraizálás folyamatát részletesen vizsgáljuk, és ha az algebrai szemantika és az eredeti „primer" szemantika kapcsolatát is vizsgáljuk. Az ide vonatkozó AGN és ADJ eredményekre építhető nyelv- és szemantikadefiniálási módszerrel foglalkozik [46] 8–20 oldal. A környezetfüggő nyelvtanok esetét külön vizsgálja [46] 8. fejezete.

Kiegészítésül még megemlítenénk az alábbiakat:

A *szemantikának* az AGN csoport eddig három lényeges arculatát vizsgálta:

α) Univerzális Algebrai (ld. [3] 4IV, [15, 19, 10] 42).

β) Kategóriaelméleti (ld. [43, 44, 45, 20, 21, 12]).

γ) Általános Modellelméleti vagy „Absztrakt Modellelméleti" (ld. [6, 9, 12, 31, 37] 44, [47, 48, 49]).

A cél nyilván az α, β és γ közelítésmódok együttes alkalmazása az 1971–73-as AGN publikációkban kitűzött program megvalósítására. A [46, 49] és [48]-ban az α és γ közelítésmódok összekapcsolásával és az eredeti célokra való alkalmazást előkészítő ismeretek elmélyítésével foglalkoztunk.

4. Programok és Programsémák Szemantikája, Programhelyesség-bizonyítás Modellelméleti szemantikai vonatkozásai, teljességi kérdések

Programsémák szemantikájának megadásával és általában a szemantikamegadás módszertanával foglalkozik pl. [47, 48] II rész és [49]. *Az utalások mostantól kezdve* a [48] II. részére vonatkoznak (oldalszámmal, fejezetszámmal, stb. fogunk utalni).A programsémák szemantikájával kapcsolatos problémakör közérthető és intuitíven alátámasztott körvonalazása [36], mely megoldásokat azonban még nem javasol. Tehát bevezetés. A kérdést elsősorban a *Programhelyesség-bizonyítás* szempontjából nézzük. Az alapgondolatokat [42] tartalmazza.

Programsémák szemantikájának egy konkrét megadási módját tartalmazza a 3. és 5. fejezet valamint [16, 17, 23, 24]. Az idézett szemantika használhatóságát ([37] értelmében) bizonyítjuk a 3. és 5. fejezetekben és [24, 22]-ben. A 4. fejezet és [23] a szokásos vagy *sztenderd* Programszemantika használhatatlanságát bizonyítja (és ennek okait vizsgálja).A sztenderd szemantika használhatatlanságát már [37] (109 old.) és [16, 17] is bizonyította, de az ottani bizonyítással szemben A. Salwicki azt a *kifogást* támasztotta, hogy az adatstruktúráról feltettük hogy teljesíti a *Peano Axiómákat*. Ezért a 4. fejezetben bizonyítjuk, hogy a [37]-ben kimondott negatív eredmények akkor is fennállnak, ha a Peano Axiómákat *elhagyjuk*, továbbá akkor is, ha a bizonyítandó E halmazba csak olyan kimenőfeltételeket veszünk be, melyek egyetlen literálból állnak (pl. nincs kvantor), és melyek *kielégíthetősége bizonyítható*. Ugyanakkor kiköthetjük E-ről, hogy konstans (azaz sztenderd) adatokra termináló programokból álljon. Az E halmazt [37] Def.4.4 értelmében használjuk.Mindezen megszorítások ellenére a klasszikus sztenderd szemantika „használhatatlan" marad.

A most idézett viták, sikerek és balsikerek talán érzékeltetik, hogy a *szemantikamegadás* egy olyan tevékenység, melyhez *módszertanra* lenne szükség.

A *szemantikamegadás módszertanával* foglalkozik a 2. fejezet 14–60. old. és [49] 2. fejezete. Valamely L nyelv *szemantikájának megadása* tulajdonképpen egy definíció, másszóval valamely ún. metanyelven írt szöveg. Ezt a szöveget (mely tehát a kérdéses L nyelv szemantikáját hivatott megadni) mi az L nyelv *prezentálásának* nevezzük, ld. 2.3.Def. (23.old.) és [49]. A 2.3 definíció előtt és után különböző konkrét prezentálásokat (ismert szemantikamegoldásokat) tekintünk át, és megpróbáljuk a prezentálás módjából származó kellemes és kellemetlen következményeket számbavenni az egyes konkrét esetekben. Végül a 2.6. definícióban bizonyos előírásokat javaslunk, melyeket bármely nyelv prezentálása (szemantikamegadása) során szerintünk célszerű betartani. A 2.1 tételben a javasolt előírások betartásának és be nem tartásának következményeit vizsgáljuk. Ezután a bevezetett módszer segítségével analizáljuk különböző programszemantikák és más, a Számítástudományban használatos szemantikai rendszerek tulajdonságait, és főleg ezek okát. Kicsit naívnak tűnik a kérdés, hogy

miért nem teljes a sztenderd programszemantika, a sztenderd Cselekvési Logika, vagy a sztenderd magasabbrendű logika. A Számítástudományban azonban egy olyan válasz, hogy az L nyelv nemteljes, *nem* elégít ki minket, ezen túlmenően azt is tudni szeretnénk, hogy „ha tehát ez a helyzet, akkor most mit lehet tenni?", miért nem teljes az L, hogyan lehet olyan L' nyelvet létrehozni az eredeti L helyett, mellyel céljainkat mégis el tudjuk érni. (Ezt a gondolkodásmódot tükrözi a [37] tanulmány, mely, miután a II. részben felméri bizonyos $\mathbb{D}$ nyelvek tulajdonságait, a III. részben bevezet olyan $\mathbb{D}^S$ nyelveket, melyek már mentesek az eredeti $\mathbb{D}$ nyelvek hiányosságaitól.) A 2.4 tétel bizonyítása és a[49] 2.4. tétel bizonyítása, úgy érezmegmagyarázza például, hogy a magasabbrendű sztenderd szemantikák *miért* nem teljesek. Úgy tűnik, hogy nem igaz az, hogy „A Magasabbrendű Nyelvek" nem teljesek. Az ilyen néven ismert tételek csak annyit bizonyítanak, hogy az a matematikai modell, amit a magasabbrendű nyelvekről első kísérletként alkottak, anomáliákat mutat. De nem maga a vizsgált nyelv, hanem annak csak *egyik* matematikai modellje vagy prezentációja mutatja az anomáliákat. Tehát egyszerűen arról van szó, hogy bizonyos jelenségek matematikai modellezése nehezebb, *több körültekintést igényel,* mint más egyszerűbb jelenségeké.

Azt az eredményt kaptuk például, hogy a magasabbrendű nyelvek kutatásánál a *matematikai* vizsgálat ma ott tart, hogy a *fogalom* megalkotása *még nyitott* probléma. Tehát a magasabbrendű nyelvek matematikai elmélete nehezebbnek bizonyult, mint a zérus- és elsőrendű nyelveké, és ennek megfelelően még a *fogalomalkotási* fázisban vagyunk: ez még nem fejeződött be. (A „Minőségi Matematikában" a súly egyre inkább a fogalomalkotási fázisra tevődik át: ld. [33] és [11].) A fogalomalkotás során elkövethető „meggondolatlanságok"-ra példa a következő: A programsémák sztenderd szemantikájába be van építve a következő axióma: A programíró és programhelyesség-bizonyító folyamat ideje abszolut szinkronban van a jövőben majd végrehajtandó összes lehetséges folyamat idejével. (Folyamatok idejének szinkronitását [51] értelmében használtuk.) Ez az axióma minden magyarázat nélkül, mint magától értetődő dolog szerepel a definícióban *implicit* módon (részletesebben ld. [37] bevezetését az implicitségről). A tudománytörténetből kiderül, hogy „magától értetődő dolog" a tudományban ritkán ártalmatlanok.

A programozási nyelvek szemantikájában a *nemsztenderd* közelítésmód [22, 24, 37] úgy tűnik, hogy jóval használhatóbb, mint a sztenderd [14, 36, 40], ld. pl. [32, 37, 16, 17, 22, 27, 48, 49]. A sztenderd szemantika művelői részéről bizonyos *értetlenség* tapasztalható. Erről az értetlenségről részletesebben szól [52]. A [48] II. rész, [49] egyik célja a fenti helyzet javítása olymódon, hogy a szemantikamegadás alapjainak részletes vizsgálata révén bizonyítjuk, és precízen, részletesen kifejtjük azokat a meggondolásokat, melyeket [32, 23, 24] és [37] bevezetése a nemsztenderd szemantika indoklásaként elmondott. A nemsztenderd és sztenderd szemantika viszonyát vizsgálva azt kaptuk, hogy a nemsztenderd szemantika egy jelenségkör mélyebb, alaposabb és igényesebb vizsgálatát-megértését tükrözi, következésképp használhatóbb, és jobban ki van téve a „köznapi gondolkodás" tehetetlenségéből származó támadásoknak. Mint ilyen, a tudományos történetében nem áll egyedül (sőt, úgy találtuk, hogy a szemantikaelmélet történetén belül sem). Ld. még (i)-(vi) oldal és 82–83 oldalakat [48] II. rész-ben.

5. A Számítástudományon belül néhány látszólag különböző területen hasonló szemantikai problémák merültek fel. Például:

a) *Problémamegoldáselmélet* (ld. [2, 6, 9, 31, 33, 51]). Itt ún. Nyelvhierarchiák elméletére-kezelésére-generálására lenne szükség a [2]-ben elmondott módon. Tulajdonképpen ugyanez a helyzet a Struktúrált Feladatspecifikálás esetében [29, 25, 28] (mint már említettük [46]

kapcsán). Magasabbrendű nyelvek mélyebb megértésére az AGN munkákban OSDT hierarchiának nevezett nyelvi rendszereknél (ld. [34] 139–140 old., [2] 96. old.) szükség lenne.

b) *A Cselekvési Logika* kidolgozatlansága, ld. [29, 6], stb. szintén nehézségeket okoz a Számítástudomány bizonyos területein (ld. [48] II. rész 73. old., [49] 42.4.).

c) *Nyelvek szemantikája, természetes nyelvek gépi kezelése* (ld. [41, 53, 35, 40]) szintén a magasabbrendű nyelvek megértését sürgeti.

d) *A Mesterséges Intelligenciakutatások*ban a magasabbrendű tételbizonyító programok lényeges szerepet játszanak és ugyanakkor a [2]-ben javasolt nyelvhierarchiák kidolgozatlansága nehézségeket okoz.

Kutatásainkat a fenti problémák is motiválják. Úgy tűnik, hogy a [48] II. rész és [49] alapján a felsorolt a) – d) témákban is haladás remélhető. Részletesebben ld. [48] II. rész (i)–(ii) old., 73–74 old.,

Irodalomjegyzék

[1] Andréka, H. Balogh, K. Lábadi, K. Németi, I. Tóth, P.: Autokód szintű szemantikus program ellenőrző rendszer továbbfejlesztésének terve, NIM IGÜSZI tanulmány. 1974. aug.

[2] Andréka, H. Gergely, T. Németi, I.: Problémaorientált nyelvhierarchia és beállító logika. KFKI–72–46, 1972.

[3] Andréka,H. Gergely, T. Németi, I.: Purely algebraic construction os first order logic. KFKI–73–71, 1973.

[4] Andréka, H. Gergely, T. Németi, I.: Vizsgálatok az algebrai logika területén. KFKI–73–24, 1973.

[5] Andréka, H. Gergehyl, T. Németi, I.: Az n-edrendű nyelvek néhány kérdéséről, Matematikai Lapok, 24. évf. 1–2 szám, 1973.

[6] Andréka, H. Gergely, T. Németi, I.: Magasszintű mesterséges intelligencia tudásreprezentációjának eszközei. Preprint, 1974.

[7] Andréka, H. Gergely, T. Németi, I.: Easily Comprehensible Mathematical Logic and its Model Theory. KFKI–75–24, 1975.

[8] Andréka, H. Gergely, T. Németi, I.: A számítógépek nem-numerikus felhasználásának egy irányzatáról. Információ Elektronika, 1975/1, 52–57.

[9] Andréka, H. Gergely, T. Németi, I.: On the role of mathematical language concept in the theory of intelligent systems. Advanced Papers of the Fourth International Joint Conference on Artificial Intelligence, Tbilisi 1975. MIT Publications Dept, Massachussetts, 9–14.

[10] Andréka, H. Gergely, T. Németi, I.: On universal algebraic construction of Logic. Studia Logica XXXVI, 1–2. 9–47. 1977.

[11] Andréka, H. Gergely, T. Németi, I.: Hozzászólás a matematika alkalmazásairól szóló vitához melyet a Bolyai János Matematikai Társulat rendezett, 1977.

[12] Andréka, H. Gergely, T. Németi, I.: An approach to Abstract Model Theory, Abstract. J. Symb. Logic, 1978.

[13] Andréka, H. Németi, I.: The generalized completeness of Horn predicate logic as a programmint language. Dept. of A.I. Research Rep. No 21, Univ. of Edinburgh 1976.Acta Cybernetica, Tom 4. Fasc 1, Szeged 1978. 3–10.

[14] Andréka, H. Németi, I.: Programhelyesség-bizonyítási és programspecifikációs módszerek teljességéről. SZKI Elméleti Labor és MTA MKI tanulmány, 1977. dec.

[15] Andréka, H. Németi, I.: On universal algebraic logic. Preprint 1978.

[16] Andréka, H. Németi, I.: Completeness of Floyd Logic. Bull. Sec. Logic Vol. 7 No 3, Polish Acad. Sci., 1978.

[17] Andréka, H. Németi, I.: A characterization of Floyd-provable programs. Proc. Coll. Logic in Programming, Salgótarján 1978. Colloq. Math. Soc. J. Bolyai, North-Holland. To appear.
[18] Andréka, H. Németi, I.: Applications of Universal Algebra in Computer Science, a Survey & Bibliography. To appear in CL & CL, 1979.
[19] Andréka, H. Németi, I.: Systems of varieties definable by schemes of equations. Algebra Universalis 1979. To appear.
[20] Andréka, H. Németi, I.: Los lemma holds in every category. Studia Sci. Math. Hung. To appear.
[21] Andréka, H. Németi, I.: Generalization of variety- and quasivariety concepts to partial algebras through category theory. Dissertationes Mathematicae (Rozprawy). To appear.
[22] Andréka, H. Németi, I. Sain, I.: Many-sorted model theory to turn negative results on program schemes to positive. Prepring 1978.
[23] Andréka, H. Németi, I. Sain, I.: Completeness Problems in Verification of Programs and Program Schemes. Mathematical Foundations of Computer Science „MFCS' 79, Olomuc". Springer 1979. To appear.
[24] Andréka, H. Németi, I. Sain, I.: On Languages for Reasoning About Programs. Fundamentals of Computation Theory, FCT'79. LNCS. Springer 1979.
[25] Burstall,R.M. Goguen, J.A.: Putting Theories together to make Specifications. Proc. Fifth Int. Joint. Comp. on Artificial Intelligence. MIT, Cambridge, Mass., pp. 1045–1058.
[26] Cartwright, R. McCarthy, J.: Recursive programs as functions in a first order theory. Preprint Stanford Univ. 1979.
[27] Csirmaz, L.: On definability in Peano Arithmetic. Preprint No 11/1979, MTA MKI. Submitted to Bull. Section of Logic. Wroclaw.
Csirmaz, L.: Structure of program runs of non-standard time. To appear in Acta Cybernetica Szeged.
[28] Dávid,G.: Proving correctness and automatic synthesis of parallel programs. MTA SZTAKI Preprint, 1979.
[29] Dömölki,B.: Structured Abstract Models. INFELOR Preprint 1974. Előadás: Luppa sziget 1974.
[30] Gergely,T.: A mesterséges intelligencia kutatás logikai eszközeiről. Rendszerelméleti Konferencia'73, Problémamegoldási Szekció, Sopron, 1973, 34–42 old.
[31] Gergely,T.: Role of a General Language Concept in the Construction of an Abstract Cognition Theory. In J. Rose and C.Bilcin (Eds.) Modern Trends in Cybernetics and Systems 2 (Springer 1977) 131–152.
[32] Gergely, T.: May the theory of programming be first order? In the collection of abstracts of the Conference „Logic in Programming" Salgótarján, 1978.
[33] Gergely,T. Németi,I.: Az általános rendszerelmélet formalizálásának és alkalmazásának logikai alapjai. Rendszer Kutatás, Közgazdasági és Jogi Könyvkiadó, 1973.
[34] Gergely,T. Németi,I.: On the role of general system theory in the cognitive process. Progress in Cybernetics and Systems Research, Vol. 2, 1975.
[35] Gergely,T. Szabolcsi,A.: How to do thighs with model theory. CL & CL, to appear.
[36] Gergely, T. Szőts, M.: On the incompleteness of proving partial correctness. Act Cybernetica, Tom4, Fasc 1, Szeged, 1978. 45–57.
[37] Gergely,T., Ury,L.: Mathematical Programming Theories. SZÁMKI Preprint, 1978.
[38] Goguen,J.A.: Some ideas in Algebraic Semantics. UCLA Preprint, 1978, Los Angeles.
[39] Hayes,P.J.: A logic of actions. Machine Intelligence 6, 495-520, Edinburgh Univ. Press, 1971.
[40] Márkusz,Zs. Szőts,M.: Semantics of programming languages defined by Universal Algebraic tools. Proc. Coll. Logic in Programming, Salgótarján 1978. Colloq. Math. Soc. J. Bolyai, North-Holland. To appear.

[41] Montague,R.: The Proper Treatment of Quantification of Ordinary English. Approaches to Natural Languages: Proceedings of the 1970 Stanford Workshop on Grammar and Semanti 1973.

[42] Németi,I.: Nagy és egyben komplex programrendszerek bejáratása. Számológép 1971/3. 59–70. old.

[43] Németi,I.: Parciális algebrák és absztrakt modellelmélet. SZÁMKI kiadvány 1976 No 1690/

[44] Németi,I.: From hereditary classes to varieties in abstract model theory and partial algebra. Beiträge zur Algebra und Geometrie 7 (1978), 69–78.

[45] Németi,I. Sain,I.: Cone-injectivity and some Birkhoff-type theorems in categories. Contribu to Universal Algebra Proc. Coll. Esztergom 1977. Colloq. Math. Soc. J. Bolyai, North-Holla To apprear.

[46] Németi,I. Sain,I.: Connections between algebraic logic and initial algebra semantics of CF languages. Proc. Coll. Logic in Programming, Salgótarján 1978. Colloq. Math. Soc. J. Bolya North-Holland. To appear.

[47] Sain,I.: A szemantikamegadás modellelméleti és univerzális algebrai módszereiről. Preprint, SZKI Elméleti Labor, 1978.

[48] Sain,I.: Programozási nyelvek szemantikája vizsgálatának algebrai és modellelméleti eszköz I, II rész. SZKI Elméleti Labor tanulmány 1979. márc.

[49] Sain,I.: There are general rules for specifying semantics: Observations on Abstract Model Theory. CL & CL. To appear.

[50] Sain, I.: Nyelvhierarchiák és elméletmorfizmusok. Tanulmány 1979, SZKI–SZÁMKI.

[51] Pask,G.: Conversation Theory. Elsevier, N.Y. 1976.

[52] Sánthané, T.E. Szőts,M.: Report on Conference on Logic in Programming, Salgótarján, 197 Preprint, 1979. Megjelent magyarul: Számítástechnika, X. No 2–3, 1979.

[53] Szabolcsi,A.: A természetes nyelv szemantikájának modellelméleti kezelése. Bölcsészdokto dolgozat, Budapest, 1978.

[54] Szigeti,J.: Factorization systems in categories of Arbib-Manes automata. MTA MKI Preprint 1979.

A CEDRUS INTERAKTIV TERMINÁLRENDSZER MÉRÉSE ÉS MODELLEZÉSE

Arató A.—Burány K.—Sarkadi-Nagy I.—Telbisz F.
MTA Központi Fizikai Kutató Intézet

1. Bevezetés

A KFKI 1978-ban üzembe helyezte a CEDRUS interaktív terminál rendszert. A rendszer célja elsősorban a software fejlesztés támogatása és ennek érdekében a következő szolgáltatásokat nyújtja:

— interaktív szövegszerkesztés, ami a programok javításán kívül speciális utasításokkal támogatja a dokumentáció készítést is;

— „job"-ok leadása a kötegelt feldolgozás számára, a leadott job-ok követése, illetve azok eredményeinek lekérdezése.

A rendszer rövid leírása, illetve néhány implementációs kérdés ismertetése [1]-ben és [2]-ben található.

A CEDRUS rendszerrel, elkészülte után néhány hónapos próbaüzemet tartottunk. Ez azt jelentette, hogy a rendszer 4 terminállal naponta több órát (4—5) üzemelt. Ennek a célja elsősorban az esetleges hibák felderítése volt. 1978 november óta a rendszer 10 terminálról napi 12—14 órán keresztül vehető igénybe.

Az interaktív rendszerekben — mint a CEDRUS is — a válaszidő az egyik legfontosabb paraméter, ami a rendszer működését jellemzi. A rendszer válaszidejének az emberi munkatempóval összhangban kell lennie, és a várakozási időknek az emberi tűrőképesség határain belül kell lenniük. A felhasználó kb. 5 sec körül kezdi elveszíteni a türelmét, és 10 sec-nál hosszabb időket gyakorlatilag tűrhetetlennek érez.

Ebből kiindulva, nagyon fontosnak éreztük azt, hogy a rendszer működéséről megbízható adatokat gyűjtsünk, hogy ezeket a későbbiekben a rendszer hangolásánál, esetleges módosításoknál és a terhelhetőség határainak a megállapításánál felhasználhassuk. Az alábbiakban ismertetjük ezen adatgyűjtésnél használt módszereket, és a gyűjtött adatokat. A rendszer szimulálására kidolgoztunk egy módszert, amivel a módosítások várható hatását kívánjuk előre megbecsülni.

2. A CEDRUS, mint két gépes rendszer

A CEDRUS szövegszerkesztő program két gépen működik. A gépek közötti feladatmegosztásnak az a célja, hogy minden funkciót abban a gépben valósítsunk meg, ahol az a leggazdaságosabban elvégezhető. Igy, mivel az ESZR gép fejlett operációs rendszerrel és file kezeléssel, valamint viszonylag bőséges háttértárral rendelkezik, a file kezelést és a rekord szintű javítást a nagygép végzi, de a front-end processzor felől beérkező kérelmeket sorosan dolgozza fel, egyidejűleg mindig csak egy felhasználóval foglalkozva, mivel a párhuzamos feldolgozás egyrészt jóval nagyobb memóriaigényt, másrészt gyakori task-kapcsolást jelentene, ez utóbbi pedig az OS-nél meglehetősen lassú és nehézkes folyamat (300—500 utasítás).

A front-end processzor végzi párhuzamos feldolgozással a rekordon belüli javításokat, valamint a felhasználói parancsok szintaktikus analízisét és dekódolását. A nagygép kódolt formában kapja már ezeket az utasításokat. A két gépes rendszert az 1. ábra illusztrálja.

3. Válaszidők a szerkesztő programban

A szerkesztő program válaszidejének azt az időt nevezzük, ami a parancs elküldését lentő „RETURN" billentyű lenyomásától addig az időpontig telik el, amikor a képernyőn megjelenik a válasz első karaktere. A teljes válaszidő a következő összetevőkből áll:

$$T = \sum_{i=1}^{n} t_S^i + \sum_{i=1}^{m} t_E^i + 2t_{CH} + t_Q + \sum_{i=1}^{b} t_H^i \qquad (1)$$

itt:

t_S^i — egy utasítás szintaktikus analíziséhez és dekódolásához szükséges front-end proces CPU idő

t_E^i — egy utasítás végrehajtásához a front-end processzorban szükséges CPU idő

t_{CH} — a kommunikációs buffer átadásához az ESZR gép csatornájánál szükséges idő

n,m — sztochasztikus változók és a kisgépben egyidejűleg befutó kérelmekből felépülő sc kat írják le (n+m terminálos maximális száma)

t_Q — a nagy gépbe való átjutás során eltöltött várakozási idő

t_H^i — a parancs feldolgozása a nagy gépben (Host idő)

b — egy utasítás végrehajtásához szükséges buffer cserék száma.

Méréseink szerint az utasítások szintaktikus analízisének ideje, bár függ az utasítás tí sától, nem több 10 msec-nél, és az egyéb adminisztrációs idő: t_E még ehhez képest is elh nyagolható. n és m értéke gyakorlatilag 1-nél szintén soha nem nagyobb, mivel a front-en processzor igen gyors az emberi reakcióidőkhöz képest. Mivel a két gép közötti kommuni káció sebessége 250 kbyte/sec és a kicserélt bufferek mérete 512 byte, egy buffer csere i 4 msec. t_Q és t_H analitikusan nem írható le, méréseink és szimulációnk főleg ezzel a két mennyiséggel foglalkoznak. t_H-nál a fő késleltetést a mágneslemezek hozzáférési ideje adja Ez az EC 5056-os lemezeknél átlagban 90 msec, és minden egyes válaszhoz több I/O műv let szükséges. Mivel, mint ez a később ismertetendő mérésekből látható, t_Q is több sec le így a válaszidő kifejezés az első három tagot elhanyagolva, a következő lesz:

$$T = t_Q + \sum_{i=1}^{b} t_H^i \qquad (2)$$

Mivel az elhanyagolt tagok olyan nagyságrendűek, hogy hatásukat úgy sem veszi ész a terminál felhasználója, a fejezet elején definiált válaszidő helyett az alábbi mennyiséget kintettük válaszidőnek: azon idő, amelyik az utasítás buffernek az ESZR csatorna adapter „felmenő" sorába történő beállítása és a választ tartalmazó buffer visszaérkezése között el telt. Bár ez a (2) kifejezéstől 2 t_{CH} -ban eltér, ezt a tagot itt is elhanyagoljuk.

4. Software és hardware mérési módszerek

A bemenő utasítások eloszlását és a válaszidőt a front-end processzor software-je és hardware-je segítségével mértük. A front-end processzor adaptere lehetőséget ad az ESZR szelektor csatornáján független fizikai címek emulálására. Az egyik ilyen cím szolgált a m re. Amennyiben a mérés és modellezés csak az ESZR gépben működő programok elemzé

korlátozódik, mint ahogy ezt a 3. pontban leírtuk, ez a mérési módszer a front-end processzor programokkal és hardware-rel csaknem ideálisan pontos eredményt ad.

A mérési eredmények csak azért nem ideálisak, mert a front-end processzorban nincs háttértár. A mérési adatokat az ESZR gépen mágnesszalagra írtuk. Ezt egy egyszerű segédprogrammal oldottuk meg, mivel a front-end processzorból az adatok egy emulált kártyaolvasón keresztül kerülnek a nagygépbe. Hogy a program hatását a mérési eredményekre csökkentsük, ez a segédprogram alacsonyabb prioritáson fut, mint a CEDRUS, és az adatokat nagyobb blokkokba rendezi. A mérési elrendezést a 2. ábra mutatja.

A mérőprogram regisztrálja minden utasítás buffernek a buffersorba való beállításának idejét, és azt az időt, amely ettől a kezdeti pillanattól a válaszbuffernek az ESZR gépből való visszatéréséig eltelik. Az időt a front-end processzor real-time órájával mérjük. A link buffer ESZR gépből való megérkezése után a mérő program egy mérő rekordot állít össze (CMR—CEDRUS Measurement Record). Öt ilyen rekordot egy blokkba gyűjtünk (CMB—CEDRUS Measurement Block). Ezeket a blokkokat egy segédprogram (IEBGENER) 3600 byte-os blokkokba rendezi. Azért, hogy a mágnesszalagra írás miatti késés ne befolyásolja a mérési adatokat, a front-end processzorban külön memóriaterület van CMB számára. Még ha a gépek közötti buffercsere maximális sebességgel (25 kártya/sec) történne is, a mágnesszalagra felvitel kis gyakorisággal történik (1 felírás 9 sec-enként). Ha figyelembe vesszük, hogy a mágnesszalagok más csatornára kapcsolódnak, mint a mágneslemezek, látható, hogy ez a mérés-adatgyűjtő rendszer csak nagyon kicsit perturbálja a rendszer működését.

Az utasítások válaszidejének eloszlása a 3. ábrán látható. Az utasítások 76%-ra kapott válasz 5 sec-on belül van, és 10 sec-on belül az utasítások 88%-ára megérkezett a válasz.

5. A CEDRUS terminálrendszer modellezése

A CEDRUS ESZR gépben működő része modellezésének egyik célja az volt, hogy meghatározzuk a válaszidőket arra az esetre, ha a rendszerben gyorsabb lemezek lennének, hogy ezen gyorsabb lemezek esetén hány terminál dolgozhat egyidejűleg, még elviselhető válaszidőket feltételezve. Végül a modellezés rámutathat, hogy milyen változtatásokat érdemes végrehajtani a programban, hogy csökkentsük a válaszidőket.

A modellezés céljára a GPSS [3] szimulációs nyelvet választottuk. A modell felépítésénél a legfőbb szempont az volt, hogy minden végrehajtási időt a lemezhozzáférési időkre vezessünk vissza, mivel ezek befolyásolják döntően a válaszidőket. A modell kimenő adata egy sor táblázat és hisztogram, melyek a válaszidők eloszlásait mutatják utasításokra lebontva és összegezve.

A mérésekkel ellenőrzött modell felhasználható a rendszer továbbfejlesztésében. Az R40 gép bővítéseként nagyobb írássűrűségű és gyorsabb átlagos elérési idejű mágneslemezeket fogunk beszerezni. Feltehetjük a kérdést, milyen mértékben hatnak majd a nagyobb és gyorsabb lemezek a CEDRUS válaszidők alakulására. Ilyen módon megválaszolható a kérdés, hány terminált lehet még venni a CEDRUS-hoz, ha gyorsabb lemezekkel fog működni.

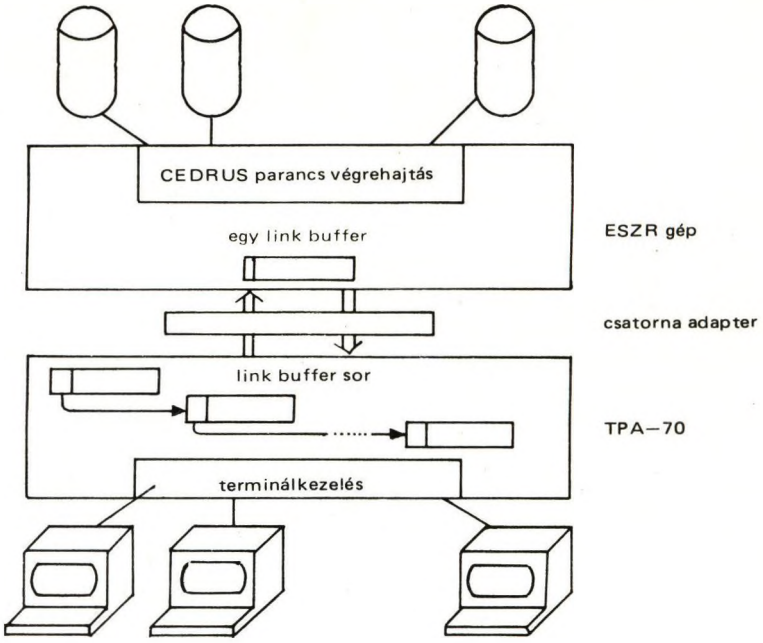
A modell felhasználható a rendszer jellemzőinek tisztán programozási eszközökkel való javítására is. Segítségével meghatározhatók azok a strukturális változtatások és paramétereik, amelyek segítségével csökkenthetők a válaszidők. A modell segít feltárni a rendszer szűk keresztmetszetét is.

Irodalomjegyzék

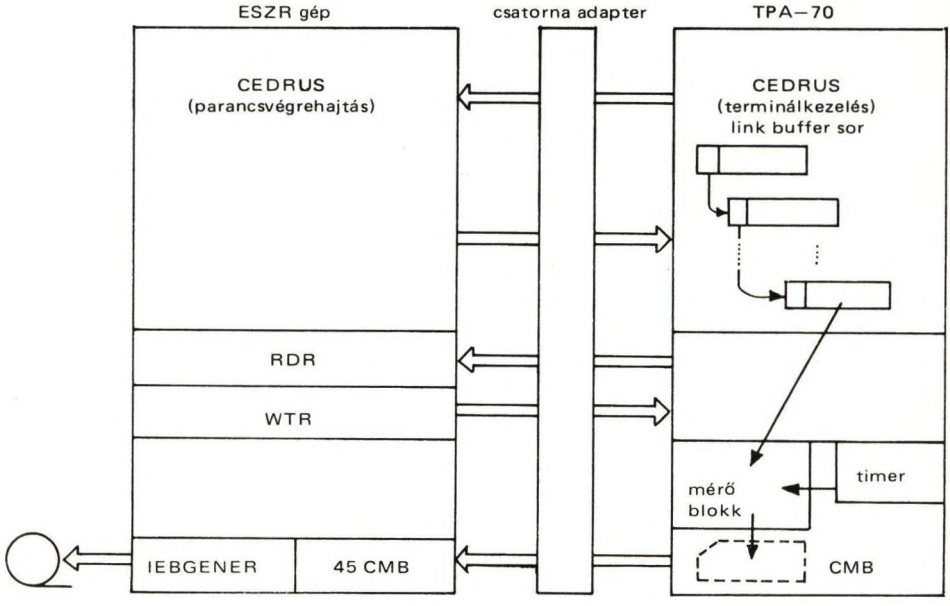
[1] A. Arató – I. Sarkadi-Nagy – F. Telbisz: A local network for the support of software development. Proc. of IFIP COMNET '77 vol. 1. p. 227–238. (1977)

[2] Arató A. – Sarkadi-Nagy I. – Telbisz F.: Feladatmegosztás ESZR gép és front-end processzor között a CEDRUS terminálhálózatban.
Programozási Rendszerek '78 konferencia, Szeged, I. 14–19 old. (1978)

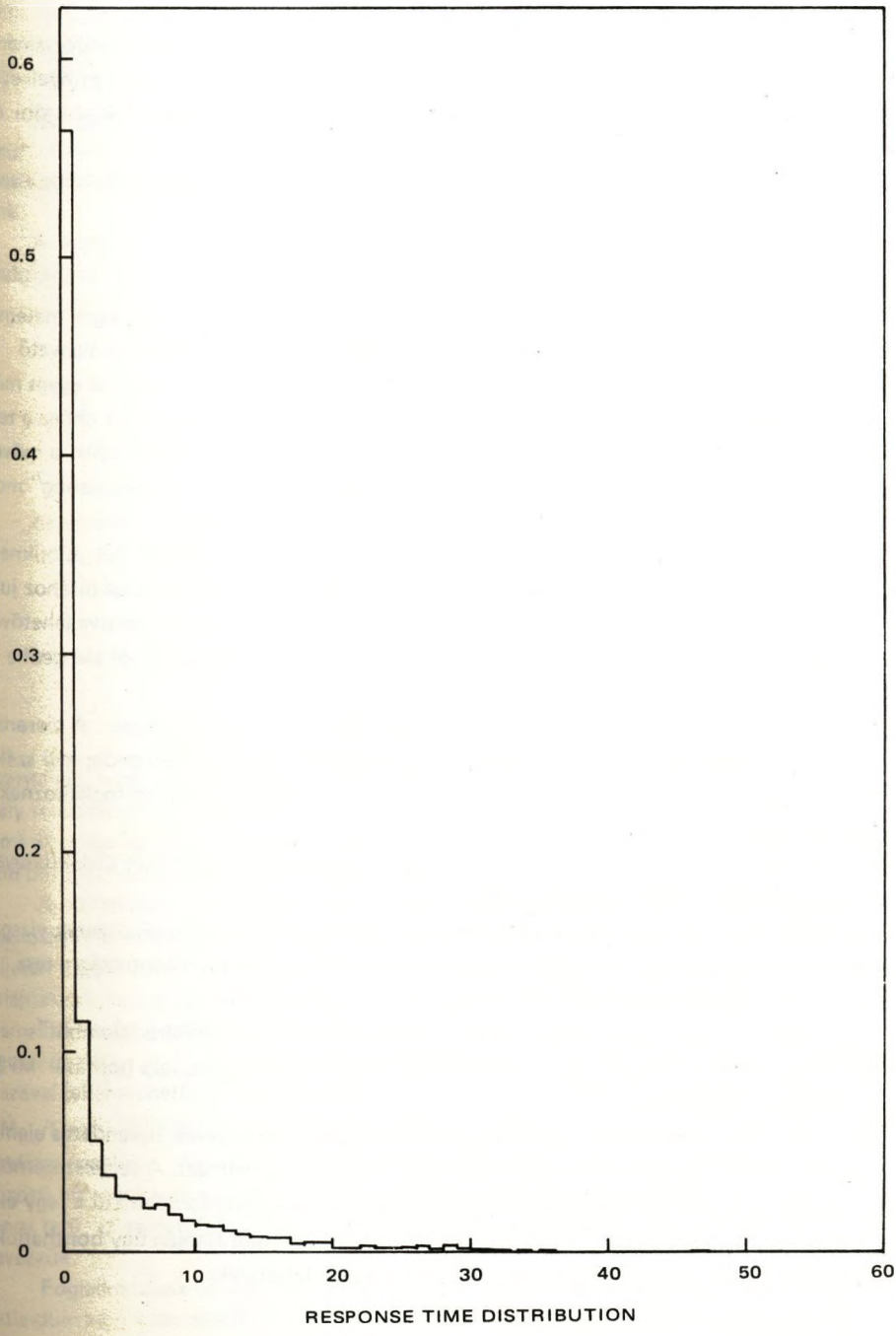
[3] General Purpose Simulation System/360. User's Manual IBM 1968.



1. ábra A CEDRUS mint kétgépes rendszer



2. ábra A mérésadatgyűjtés elvi felépítése



3. ábra A mért válaszidők eloszlása

VEGYIPARI RENDSZEREK SZIMULÁCIÓJA

Árva P.,—Bencze T.,—Szeifert F.
VESZPRÉMI VEGYIPARI EGYETEM

A számítógépes szimuláció a 70-es évektől kezdődően megszokott eszköz a vegyészmérnöki gyakorlatban. Hazánkban az elsők között dolgozták ki a SIMUL alkalmazói programnyelvet [E= Ennek ellenére a nagy szimulációs rendszerek kialakításának és elterjedésének csak az utóbi év ben érlelődtek meg a feltételei.

Mik ezek a feltételek? Ezt analizáljuk a továbbiakban, s arra is választ adunk, hogy a sok alternatív lehetőség közül saját munkáinkban mi melyik utat választottuk.

1. Matematikai modellezés

A számítógépes szimuláció alapja a matematikai modellezés. Műveleti egységek matematikai modellezésének fundamentumait hazánkban BENEDEK és LÁSZLÓ rakták le alapvető munkájukban [2] a 60-as évek elején. Ez termodinamikai alapokon lehetővé teszi az egyes műveleti egységek modelljeinek többnyire heurisztikus előállítását. Szorosan illeszkedik ehhez a termodinamikai tulajdonságok számítását elvégző összefüggés halmaz. Igy lehetővé válik a műveleti egységeknek, mint elemeknek rendszerré való szervezése. Ezek az ún. „flow-sheeting" modelek, amelyek, amelyek számos alkalmazói programnyelv alapjául szolgálnak.

A műveleti egységek — mint építő elemek — teljes technológiák leírásánál túl „kicsiknek" bizonyulnak. Ugyanis a rendszer nagy elemszáma miatt túlságosan bonyolult struktúrához jutunk. Másrészt a műveleti egységeket — mint rendszereket kisebb egységekre bontva lehetővé válik azok modelljeinek automatikus generálása. Ez a hatékonyság szempontjából elengedhetetlen követelmény.

A fenti problémákat a hierarchikus modellezés módszere hivatott megoldani. A hierarchikus modellezést a reaktor technikában kezdték el kialakítani [3], napjainkban pedig már széles körben elterjedt „gondolkodási módszer" [4]. Hazánkban is több kutatóhelyen foglalkoznak a hierarchikus modellezési módszer kialakításával [5, 6, 7].

Kollektívánk kb. 1973. óta foglalkozik tudatosan a hierarchikus szemlélet kialakításával ill. az oktatáson keresztül, annak elterjesztésével.

A hierarchikus modellezés igénye azzal kapcsolatos, hogy a bonyolult rendszerek vizsgálatánál a minőségi tulajdonságok szerepeltetése a modellekben ugyanolyan fontosságra tesz szert, mint a mennyiségi tulajdonságok jelenléte.

A különböző vegyipari objektumok összetettsége szembeötlő. A szakirodalomból ismeretes vegyipari rendszerek különböző szempont szerinti hierarchia szintekre való bontása. Mi az alábbi nomenklatúrát használjuk.

Tekintsük az $E' = \{$technológiai rendszer, műveleti egység, berendezés, berendezés elem, fázis, fázis elem, kémiai komponens, molekula, atom$\}$ tulajdonság-halmazt. A vegyészmérnöki tapasztalat alapján az esetek többségében av vizsgált objektumhoz hozzárendelhető E' egy és csak egy eleme. Az R_o objektumot a hozzárendelt $E_i' \in E'$ tulajdonság alapján úgy bonthatjuk fel $R_i / i \in P_n = \{1,2,\ldots n\}$ részekre, hogy teljesülnek az alábbi feltételek:

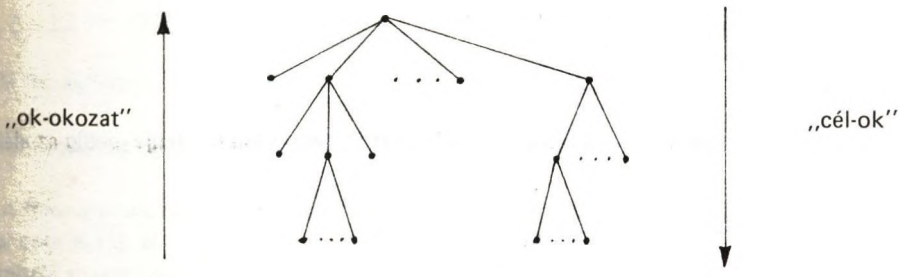
1. $\bigcup_{i \in P_n} R_i = R_o$ (az R_i-ik lefedik R_o-t)

2. $R_i \cap R_j = \emptyset / i, j \in P_n, i \neq j$ (az $R_i$ik diszjunktak).

Az $E'_i \in E'$ tulajdonságokon túl egyéb olyan tulajdonságok is definiálhatók, amelyek vegyipari objektumok halmazát ekvivalencia osztályokba sorolják (ez okozza a hierarchia- szintekre bontás sokféleségét a szakirodalomban). Legyen ezen újabb tulajdonságok halmaza T, továbbá $E''=E' \cup T$. Az E'' elemei között értelmezhető egy "< " rendezési reláció. Az $E''_i < E''_j$ reláció akkor igaz, ha az E''_j tulajdonságú objektum az E''_i tulajdonságú egyszeri vagy többszöri bontásával adódik. E'' tetszőleges két eleme összehasonlítható. Így értelmezhetjük a tulajdonságok $E=\{E_i / i \in P\}=<E''$ "< ">jól rendezett halmazát, amely szerint $E_i < E_{i+1}$ / $i \in P$ is felírható.

Nevezzük E-tulajdonságnak azt a tulajdonságot, amelyhez az E halmaz rendelhető „mértékül". A különböző vegyipari rendszerek hierarchia szintekre történő felbontását az E-tulajdonsággal tudjuk „mérni". Amennyiben $T=\emptyset$, a vegyipari rendszerek tipikus szintjeiről beszélünk.

A vegyipari objektumoknak az E-tulajdonság szerinti részekre bontásával egy fa rendezésű struktúrához jutunk (1. ábra). A fa bármelyik pontja egy-egy objektumot jelöl. Az élek azt rep-



1. ábra. Vegyipari rendszerek fa struktúrája

rezentálják, hogy az adott rendszernek mely rendszerek az elemei ill. azt, hogy az adott rendszer mely rendszernek az eleme. A fa szintjei a hierarchiaszintek. A fa max. eleme (a vizsgált objektum) áll a legmagasabb hierarchia szinten, míg a fa min. elemei (a vizsgált objektum tovább már nem bontott részei) pedig a legalacsonyabb szinten.

A hierarchia szintek relatív önállósággal, alá ill. fölérendeltséggel rendelkeznek. A relatív önállóságot az jelenti, hogy bármilyen vizsgálatot végzünk a rendszeren, az a hierarchia szinttől teljesen független. Az „ok-okozati" viszony az adott rendszert alárendeli az alacsonyabb hierarchiájúaknak, míg a „cél-oksági" viszony a magasabb hierarchiájú rendszereknek.

Tekintsük a tetszőleges hierarchia szinten álló $N\{0, i, \ldots\ldots, j\}$ nevű rendszert, amelynek az elemei az $N\{0, i, \ldots\ldots, j, k\}$ / $k \in P_n$ nevű rendszerek. A k-ik elem az extenzív mennyiségek M_k halmazával jellemezhető. Az elemek közötti kölcsönhatás extenzív mennyiségek áramainak $\{\dot{M}_{lm}/l, m \in \{0\} \cup P_n\}$ halmazával jellemezhető. Értelmezzük a nagy elemszámú homogén rendszer fogalmát. Az ilyen rendszerek elegendően sok izomorf elemet tartalmaznak, s az elemek közötti kölcsönhatás a térben invariáns. Az ilyen tulajdonsággal rendelkező rendszereket folytonos terű, az ezen tulajdonsággal nem rendelkező rendszereket pedig diszkrétterű rendszereknek nevezzük.

Foglalkozzunk ez utóbbiak modelljének a felírásával. Diszkrétterű rendszereknél az elemek individumként kezelendők. A rendszer modelljét visszavezetjük az elemek modelljeire, s az elemek közötti kapcsolatokra.

Valamely rendszer modellje, a környezetnek a rendszerre gyakorolt hatását egyértelműen leíró mennyiségek halmazát, a rendszernek a környezetre gyakorolt hatását egyértelműen leíró mennyiségek halmazára leképező független relációk halmaza. Az $N\{0, i, \ldots, j\}$ matematikai modellje:

$$\dot{M}\{0, i, \ldots, j\}: \{\dot{M}_{0l}, \dot{M}_{pq} \; / \; l, p, q, \in P\} \rightarrow \{\dot{M}_{pq} \; / \; p, q \in P\}$$

$$\{\dot{M}_{0l}, \dot{M}_{pq} \; / \; l, p, q \in P\} \rightarrow \{\dot{M}_{m0} / \; m \in P\} \; ,$$

ahol az $N\{0, i, \ldots, j, l\}$ rendszerekre hat az $N\{0, i, \ldots j\}$ környezete, az $N\{0, i, \ldots, j, m\}$ rendszerek pedig hatnak az $N\{0, i, \ldots, j\}$ környezetére.

Az $N\{0, i, \ldots, j\}$ és $N\{0, i, \ldots j, k\} / k \in P_n$ rendszerek áramai között az alábbi összefüggések állnak fent (az elemek közötti kapcsolatot fejezik ki):

$$L_k : \{\dot{M}_{pq} / p, q \in \{0\} \; UP\} = \{\dot{M}_{0l}, \dot{M}_{m0} / l, m \in P\}.$$

Az $N\{0, i, \ldots, j, k\}$ matematikai modellje.

$$M\{0, i, \ldots, j, k\} \cdot$$

A vizsgált rendszer elemei modelljére visszavezetett matematikai modellje pedig az alábbi lesz:

$$M\{0, i, \ldots, j\} = \{M\{0, i, \ldots, j, k\}, L_k \; / \; k \; P_n\}.$$

A legalacsonyabb hierarchia szinten levő rendszerek modelljei az alábbi formában adható meg:

$$M\{0, i, \ldots, j, \ldots\}: \{\dot{M}_{0l}, M\} \rightarrow \{M\}$$

$$\{\dot{M}_{0l}, M\} \rightarrow \{\dot{M}_{l0}\} \; .$$

Ezeknek ismertnek kell lenni.

Folytonos terű rendszereknél a térkoordináták halmaza legyen X, az ún. fázis téré pedig $y = X \cup M$. Bevezetjük az ún. Ψ_y elemszám sűrűség függvényt.

A $\Psi_y \cdot dy$ az $(y, y + dy)$ elemei fázistérben helyetfoglaló elemek számát adja meg.

Az elem-szám sűrűségre mérlegegyenletet írhatunk fel.

$$\frac{\partial \Psi_y}{\partial t} + \frac{\partial}{\partial y}(\Psi_y \cdot v_y) = q_y$$

ahol

q_y forrássűrűség.

$v_y = \frac{d_y}{dt}$ kinetikákák (az elemek modelljei).

Az $N\{0, i, \ldots, j\}$ és $N\{0, i, \ldots, j, k\}$ változói között az alábbi kapcsolat írható fel:

$$M = \int_{y' \subseteq y} M \cdot \Psi_y \, dy'$$

Ezt, továbbá az elemszám sűrűségre vonatkozó mérlegegyenletet felhasználva előállítható az $N\{0, i, \ldots, j\}$ rendszer M mennyiségeire vonatkozó differenciálegyenletrendszer:

$$\frac{\partial M}{\partial t} + \frac{\partial}{\partial (y \setminus y')} \left(\left\langle v_{(y \setminus y')} \right\rangle \cdot M \right) = q_M,$$

ahol

$\langle \ldots \rangle$ az átlagolás jele.

q_M M forrássűrűség az $(y \setminus y')$ térben.

Ez utóbbi egyenletrendszer függetleníthető az elemszám sűrűségére vonatkozó differenciálegyenlettől.

A fenti elveket figyelembe véve tetszőleges rendszer modellje az elemek modelljeinek készletének tekinthető, abból generálható. A rendszerek modelljeinek automatikus előállítása szükséges feltétele a nagy szimulációs rendszerek kiépítésének.

2. **Algoritmizálás**

A mérnöki feladatok során a rendszer egyes változóinak értéke ismert, más változók értékét viszont ki kell számolni.

Azt az utasítást, amely a rendszer modelljét felhasználva megadja azt az utat, melyen az ismert értékekből számítani tudjuk a kiszámítandó változók értékét, algoritmusnak nevezzük.

Egy sokváltozós bonyolult rendszernél számtalan variáció adódik a változóknak az ismert ill. a kiszámítandó osztályokba való besorolásánál. Olyan módszer, amely tetszőleges feladat automatikus algoritmus generálását elvégezné, még nem létezik. Emiatt a mérnöki tevékenységek alapján célszerű az algoritmusok osztályába sorolása. Üzemeltetési ill. üzemirányítási feladatok megoldásában jól használhatók az ún. szimulációs algoritmusok. Ennél ismert változók a környezet hatását leíró áramadatok, a rendszer geometriai méretei, a kinetikai összefüggések paraméterei. Kiszámítandó az összes többi áram adat. Tervezésnél viszont ismert az áram adatok nagy része, és ki kell számolni a geometriai változók értékét, különböző struktúra variációknál.

Dinamikus rendszerek szimulációs algoritmusainál a fő problémát a lassú számolás jelenti, míg stacioner állapotú rendszereknél a számolási idő-konvergencia konfliktus örökös problémát jelent. Ez utóbbihoz néhány megjegyzést fűzünk.

A modell egy sok változót tartalmazó bonyolult relációhalmaz. Ezt a relációhalmazt megközelíthetjük matematikus ill. mérnöki szemlélettel. Az első esetben az egyenlet orientált, a második esetben pedig blokkorientált módszerről beszélünk. Mérnöki körökben jelenleg érthető, az utóbbi a népszerűbb. Ez egyben általában mentesít a bonyolult egzisztencia vizsgálatoktól is.

Mindkét szemléletmódon belül a relációk felhasználása a számításhoz történhet egyszer (szimultán módszer) és egymásután (konszekutív módszer). Az előzőnél az iterációs ciklusba szereplő változók száma nagyobb, viszont a konvergenciafeltételek nagyobb biztonsággal teljesülnek. A szakirodalomban egyre gyakrabban alkalmazott módszer az, amelynél új paraméterek bevezetésével az egyes blokkok lineárissá tehetők. A lineáris rendszer számítása ezután már nem jelent gondot. Az így kiszámított változók ismeretében – konszekutív módon – a bevezetett paraméterek értékére újabb becslést kapunk. Az új paraméter értékekkel ismét a lineáris rendszer számítása következik, stb.

Rendszerünkben mind a blokkorientált konszekutív, mind ez utóbbi stratégia követésére lehetőség van. Sőt általában egy adott rendszer számításánál kombinálódnak. Egy adott alrendszer számításánál az első módszernek van prioritása. Abban az esetben, ha az iterálandó változók száma adott értéknél nagyobb, vagy a módszer divergens, automatikusan a második módszer aktivizálódik.

A számítási stratégiáknak részei a különböző matematikai blokkok (pl. differenciálegyenlet-rendszer megoldására Runge-Kutta módszerek, lineáris egyenletrendszer megoldó algoritmusok, közvetlen ill. Newton iteráció stb.).

3. Szimulációs program

A szimulációs feladatok elvégzésére számos programrendszert dolgoztak ki. Ezek a programrendszerek általában egy számítógépen vagy számítógép családnál használhatók. A programrendszerek egy része általános célú felhasználásra, más részük speciális feladatok szimulálására használható. E programrendszerek használatához különböző mértékű programozási tudás szükséges.

A vegyipari szimulációs problémák megoldására elterjedten alkalmazzák [8, 9, 10] : PACER, CONSEPT, SIMSCRIPT, GEMCS, MIMIC, DSL90, PROVES, PRIMER, SIMUL, SIMULA67 stb. nyelveket.

Az oktatási és kutatási munkák elvégzésére, a felsorolt szimulációs programok alkalmasak, de nem állnak rendelkezésünkre olyan programrendszerek és számítógép sem, amelyen bármelyiket is használni tudnánk. Igy az elmúlt évek során az ODRA–1204-es számítógépre a gépi lehetőségek figyelembe-vételével olyan programrendszert dolgoztunk ki, amely lehetővé teszi azt, hogy a minimális számítástechnikai ismeretekkel rendelkező szakemberek is hatékonyan használják a számítógépet a vegyipari rendszerekkel kapcsolatos szimulációs munkák jelentős részének elvégzésére.

A kidolgozott VETERESZ programrendszer lehetővé teszi, hogy a felhasználó szakemberek a szakterületükhöz közeli nyelven fogalmazzák meg szimulációs problémájukat. A szimulálandó rendszereknek, a rendszerek paramétereinek, a rendszerek áramainak, és az áramok változóinak nevet adhatnak, és a nevekre hivatkozva változtathatják meg a paraméterek és áramok értékeit. A programrendszer lehetővé teszi azt is, hogy egy alsó szintről kiindulva építsük fel a szimulálandó rendszert, és azt is, hogy a beolvasott rendszereken módosításokat hajtsunk végre, azaz a rendszerbe új rendszert vagy rendszereket építsünk be, töröljünk a rendszerből alrendszert vagy alrendszereket, és azt, hogy megváltoztassuk a rendszert felépítő alrendszerek kapcsolatát, azaz technológiai variánsokat vizsgáljunk.

A programrendszer kötegelt és dialógus feldolgozási üzemmódban dolgozhat. A programrendszer vezérlőprogramja a konzolon megadott utasításoknak megfelelő műveleteket hajtja végre. A megadott utasítások lehetővé teszik az alábbi műveletek elvégzését: – a rendszer –, azaz a rendszerben felhasznált paramétereknek, áramoknak és ezek változóinak, valamint ha nem automatikus számítástervezéssel dolgozunk, a rendszer számításának egyszerű programjának, (amely rendszerből és néhány vezérlőjelből áll) – beolvasását.

– a rendszerek bármelyik áramának és ha szükséges az áram bármelyik változója értékének a beolvasását;
– a rendszerek bármelyik paraméterének a beolvasását;
– a rendszerek bármelyik áramának és azok bármelyik változójának a kiírását;
– a rendszerek bármelyik paraméterének a kiírását;
– a rendszerek felépítésének módosítását, azaz a rendszerbe új alrendszert építhetünk, a rendszerből törölhetünk alrendszert, megváltoztathatjuk a rendszert felépítő elemek kapcsolatát, ha szükséges új áram beépítésével is, és ha nem automatikus számítástervezést végzünk, megváltoztathatjuk a rendszer számítási programját;
– a rendszer számítását a paraméterekkel és áramokkal meghatározott munkapontban;
– a rendszer vezérlésének szimulását az adott rendszer paramétereinek változtatásával;
– a rendszer paraméterekkel és áramokkal megadott munkapontjában a rendszer érzékenységének a vizsgálatát;
– a rendszer számításának automatikus tervezését, azaz a rendszerben levő recirkulációs körök felbontását, a felbontott rendszert felépítő alrendszerek számítási sorrendjének meghatározását, és az iterációs számítást vezérlő eljárás műveleti elem beépítését;
– a műveleti egységek számítási eljárásának generálását;
– dinamikus rendszerek szimulálását, és
– a rendszer működését leíró regressziós modellek előállítását a részletes modellek felhasználásával.

A programrendszer a műveleti egységek számítására ALGOL–1204 nyelven írt eljárásokat használ fel, ha a műveleti egység számító eljárást nem generáljuk. A számító eljárásainak felépítési szabályai paraméter megadása egyszerű, s így a számító eljárás készlet könnyen bővíthető. Jelenleg 40 számító eljárás áll rendelkezésünkre.

A programrendszer készítését is jelentősen nehezítette az elavult számítógép rendszer. Ez bizonyos esetekben körülményessé teszi a nagyobb, kb. 100 műveleti egységből álló rendszer szimulálását. Problémát okoz a számítógép háttérmemória kapacitása is, s a kis háttérmemória kapacitás lehetetlenné tette a szimuláláshoz nélkülözhetetlen fizikai-kémiai állandó adattár létrehozását és használatát. További problémát okoz a számítógép kis megbízhatósága, s ez már a kis rendszerek olyan vizsgálatánál is jelentkezik, pl. érzékenység vizsgálat amely hosszú számítási időt igényel.

A programrendszer használatával szerzett tapasztalatok alapján, már készülnek illetve elkészültek olyan résztervek, amelyek lehetővé teszik, hogy bármikor adaptáljuk a programrendszert egy korszerűbb, rugalmasabban használható és megbízhatóbb számítógépre.

A nagy szimulációs rendszer kiépítésének nem kevésbé lényeges feltétele megfelelő felkészültségű szakemberek léte. Ez elvezet a képzés területéhez. Erről a problémáról egy külön előadásban számolunk be.

Befejezésül tételesen felsorolunk néhány olyan szimulációs rendszert, amelyekkel a vegyészmérnöki feladatok megoldásában már tapasztalatot szereztünk:

Teljes ammóniagyári és timföldgyári programcsomag, amely stacioner szimulációra alkalmas.
Gáztisztító rendszer stacioner és dinamikus szimulációja;
Tetszőleges struktúrájú rektort generáló programcsomag;
Tetszőleges struktúrájú timföldgyári kikeverő rendszert generáló programcsomag.

A rendszer bővítésére és egyre inkább automatikussá tételére nagy erőfeszítéseket teszünk.

Irodalomjegyzék

[1] Benedek P. és mt.: Bonyolult műveleti egységek matematikai szimulációja, A kémia újabb eredményei 15., AK Bp. 1973.

[2] Benedek P., László A.: A vegyészmérnöki tudomány alapjai, MK. Bp. 1964.

[3] Boreszkov, G., K., Szlinko, M. G.: Vesztny. AN. SZU, 10, 29. 1961.

[4] Kafarov, V. V., Dorochov, I. N.: Vegyipari folyamatok rendszer analízise (oroszul) Nauka Moszkva 1976.

[5] Holderith J.: Magy. Kém. Folyóirat, **83**, 2, 69 1977.

[6] Blickle T.: Műszaki Kémiai Napok '77–78 kiadványai, Keszthely 1977, 1978., J. of. Ind. Chem (Veszprém) **6** (1978)

[7] Árva P.: Műszaki Kémiai Napok '75–78 kiadványai, Keszthely 1975, 1976, 1977, 1978., J. of. Ind.Chem. (Veszprém) **6** (1978)

[8] Peters, N., Barker, P. E.: An Appriasal of the use of PACER, GEMCS, CONSEPT for Chemical Plant, Simulation and Design, Chem. Eng. **9**, 149 (1974.)

[9] Wegstein, J. H. Com. A. C. M. **1**, No 6, 9 (1958.)

[10] Crowe, C., et al:: Chemical Plant Simulation (oroszul) Mir, Moszkva 1973.

AZ ORSZÁGOS JOGSZABÁLYNYILVÁNTARTÁS SZÁMÍTÓGÉPI MEGOLDÁSÁNAK KÉRDÉSEI

dr. Bacsó Jenő—Csáki Béla—dr. Dajka Miklós—Harza Lajos
IGAZSÁGÜGYI MINISZTÉRIUM – SZÁMKI

Bevezetés

A jogszabályok a társadalmi viszonyok egészét átfogják, rendezik és szabályozzák, meghatározott magatartást rendelnek el, tiltanak-, vagy engednek meg. Ily módon a jogszabály a társadalmi viszonyok külső burka, amelynek keretében az állampolgárokat és a szervezeteket jogok illetik meg, vagy kötelességek terhelik. Ezeknek az ismerete mindenki számára nélkülözhetetlen. A jogszabályt csak úgy lehet alkalmazni, ha létéről tudnak, ha köztudott, hogy az adott társadalmi viszonyt jogszabály rendezi és hogy ez a jogszabály hol található (milyen forrásból ismerhető meg) és mit tartalmaz.

Bár a jogszabályok ismerete mindenki számára szükséges, a vágyálmok birodalmába tartozik, hogy az egyénnek teljes, minden jogszabályra kiterjedő jogismerete legyen.

Hazánkban a társadalmi viszonyok fejlődése szükségessé teszi a jogszabályok felülvizsgálatát és az elavult, megváltozott, a társadalmi viszonyoknak meg nem felelő jogszabályok újabbakkal való felváltását. Ez a folyamat a korábbi jogszabályok módosításával, hatályon kívül való helyezésével, új jogszabályok kibocsátásával, néha egész jogterületek újjáalkotásával jár együtt, mindez újabb és újabb joganyag megismerését igényli. Ha még azt is hozzátesszük, hogy a jogszabályoknak és a hozzájuk kapcsolódó jogi alkalmazásoknak a száma igen nagy, akkor előttünk áll a joganyag szinte áttekinthetetlen szövevénye. Ezen a tényen az sem változtat, hogy rendszeres és következetes feladat a jogszabályok rendezése és — ott, ahol ez lehetséges — a jogszabályok számának a csökkentése.

Az átlagember a gyakorlatban — mint állampolgár — többnyire csak a polgári (adásvétel, csere stb.) — és az államigazgatási (lakás, adóügyek stb.) joggal, esetleg a büntetőjoggal kerül közelebbi kapcsolatba, és még ezeken — az egész jogterülethez képest szűkebb — területeken is igen nehéz eligazodni. Nehéz számontartani, vajon a régi és az újabb jogszabályok közül melyiket kell alkalmazni az adott esetben, mi a jogszabály hatályos szövege stb. Nincsenek tehát könnyű helyzetben az állampolgárok, gyakorlatilag azonban ugyanez érvényes a szervezetekre is az intézményekre is, amelyeknek ismerniük kell az általuk alkalmazandó jogterületet.

Ez a jogterület azonban nem azonos az alkalmazandó jogszabályokkal, hanem magába foglalja az alkalmazások során kialakult gyakorlatot, sőt a tudományos állásfoglalásokat is, amelyek ismerete már nélkülözhetetlen a korszerű jogalkalmazásban.

Mindezek alapján nyilvánvaló, hogy a jogi tájékoztatással kapcsolatos igényeket a hagyományos, kézi (kartotékrendszerű) nyilvántartás már nem képes kielégíteni, a korszerűsítés megköveteli a számítógépes adatfeldolgozási módszerek alkalmazását.

Az alábbiakban röviden ismertetjük az országos számítógép jogszabálynyilvántartás kialakításának folyamatát és lépcsőfokait.

Előzmények, célok

A bevezetőben ismertetett megfontolások alapján az Igazságügyi Minisztérium tervbe vette, hogy a VI. ötéves terv időszakában általános, minden jogszabályra kiterjedő (jogi) adatbankot hoz létre. A jogszabálynyilvántartás tehát a hazánkban megjelenő összes jogszabályt és annak gyakorlatát magába foglalná.

Ezt követően az Igazságügyi Minisztérium – nem lévén számítástechnikai szakembergárdája és információfeldolgozási tapasztalata – felvette a kapcsolatot a Számítógépalkalmazási Kutató Intézettel (SZÁMKI), majd a felek szerződést kötöttek a jogi adatbank kialakítására és működtetésére.

A Minisztérium és a SZÁMKI munkacsoportja közösen kidolgozta a jogi adatbank szervezési koncepcióját, amelynek alapján megkezdődtek a munkálatok.

A SZÁMKI elemezte (és ma is elemzi) a jogszabálynyilvántartás számítógépi megoldását befolyásoló felhasználói tényezőket, majd ennek alapján feladatokat határozott meg, amelyek átfogó számítógépi kísérletsorozat kiindulópontjai. A kísérletek alapvető célja az *első lépcsőben* a jogi információk tárolására, feldolgozására és visszakeresésére alkalmas programrendszerek kutatása, a *továbbiakban* pedig a számítógépes rendszer kialakítása.

Az adatbank – az adatvégállomásokon keresztül – egyidejűleg több felhasználó számára adnak hozzáférési lehetőséget, Az elsődleges felhasználó az Igazságügyi Minisztérium lenne (1–2 terminál már a kísérleti szakaszban is működne), ezen kívül adatvégállomások működnének néhány intézményben (Legfelsőbb Bíróság, Legfőbb Ügyészség, Minisztertanács Tanácsi Hivatala. Minisztertanács Titkársága); továbbá a minisztériumokban és az országos hatáskörű szervekben, végül a fővárosi- és megyei bíróságokon és ügyészségeken, esetleg a megyei tanácsokon is. A jogszabálynyilvántartási rendszernek végső formájában ki kell elégítenie az állampolgárok igényeit is. A rendszer kiépítésében az *első* szakaszt az Igazságügyi Minisztérium igényeinek kielégítése jelenti, bővítésre az itteni tapasztalatok birtokában kerülhet sor.

A munkálatok az Államigazgatási Számítógépes Szolgálat (ÁSZSZ) Honeywell Bull 66 típusú nagyszámítógépén folynak.

. . . . . – – – . . . . .

Az alábbiakban kissé részletesebben ismertetjük a jogszabálynyilvántartás felépítését és a jogi adatbank működésének néhány jellemzőjét.

1. Az adatok köre

A jogszabálynyilvántartás *három* részből tevődik össze:

- a jogszabályokból,
- a bíróságok elvi jelentőségű állásfoglalásából, illetve az országos hatáskörű szervek által kiadott elvi jellegű iránymutatásokból (jogi iránymutatások) és
- a jogi szakirodalom fontosabb megállapításaiból.

Az adatbank tehát nemcsak a jogi rendelkezésekről, hanem az azok alkalmazásával kapcsolatban felgyülemlett tapasztalatokról, továbbá a kapcsolódó tudományos művek és szakcikkek megállapításairól is tájékoztatni tud majd.

Az információs egység a *jogszabály,* vagy *jogszabályhely*, változó hosszúságú rekordok.

Az adatbank ún. referencia-bázisú rendszer lesz, vagyis a jogszabályok szövegét egyelőre nem tartalmazza. Kivételt képeznek a jogi iránymutatások és a jogirodalmi megállapítások, az adatbank ugyanis ezekben az esetekben a referencia-adatok mellett a vonatkozó megállapítások (az ún. elvi magok) szövegét (vagy szövegkivonatát) is tartalmazni fogja. Ezt a megoldási forma célszerűségi – elsősorban a kutatásra vonatkozó – szempontok indokolták.

A jogszabályokat illetően az adatbank *teljeskörű* lesz, tehát a törvényeken, az országgyűlési határozatokon, a törvényerejű rendeleteken és az elnöki tanács határozatokon (ez a nem jogszabály típus jelenti az ún. törvényszintet) túlmenően tartalmazni fogja a Minisztertanács, az államtitkárok és az országos hatáskörű szervek vezetői által kiadott utasítások és rendeletek adatait is.

Egy-egy joganyagról az adatbank *egységesen* az alábbi adatbank fogja tárolni:

- azonosító adatok: a jogszabály (vagy iránymutatás) számjele (jogirodalmi mű esetén a szerző neve), a kibocsátó szerv megnevezése, a kibocsátás ideje, a jogszabály típusa stb.;
- a joganyag alkalmazásához szükséges adatok: hatályba lépés, hatályon kívül való helyezés, forráshely, a végrehajtásra és a módosításra vonatkozó adatok stb;
- a joganyag tartalmára vonatkozó tárgyszavak (deszkriptorok), a joganyag szerkezetének megfelelő bontásban;
- egyéb adatok: eljáró szervek, jogterület stb;
- a statisztikai adatszolgáltatás céljaira szükséges alapadatok.

A jogi iránymutatásokra és a jogi szakirodalmi megállapításokra vonatkozó adatok köre némileg szűkebb a jogszabályokénál.

2. A jogszabályok hierarchikus szerkezete és ennek számítógépi vetületei

Valamennyi jogszabály hierarchikus szerkezetű, azaz fejezetekre, azokon belül pedig szakaszokra, majd pontokra, bekezdésekre stb. bomlik. Ebből logikusan következik, hogy a jogszabályok teljes mélységű számítógépi feldolgozása is hierarchikus nyilvántartási szerkezetet (adatszerkezetet) „állít elő".

Természetesen másféle hierarchiák is vannak (pl. a jogszabályszintek közötti összefüggések), amelyeket folyamatosan vizsgálunk; a számítógépes feldolgozás során ezeket az összefüggéseket is figyelembe kell vennünk.

A jogszabályok feldolgozása éppen azért algoritmizálható, mert a jogszabályokról, illetve azok szerkezeti egységeiről — minden szinten — ugyanazon típusú adatokat kell nyilvántartani, tehát összeállítható az algoritmizálás alapját képező adattípus-lista. Minden szerkezeti egységre vonatkozóan *csak* azon adatokat kell megadnunk, amelyek ebben a listában szerepelnek és adott esetben értelmezhetőek is; (miután az adatlistát a nyilvántartásban szereplő *összes* jogszabály jellemzőinek figyelembevételével dolgoztuk ki, az *egyes* jogszabályok esetében az adatlista nem minden adata lesz értelmezhető).

A hierarchikus szerkezet számítástechnikailag azt jelenti, hogy a jogszabály szerkezeti egységeit képviselő adattámbök fa-struktúra szerint helyezkednek el.

Az egymás mellett és az egymás fölött elhelyezkedő adattömbök ugyanazon adatainak értékei részben megegyeznek, részben pedig különböznek egymástól, ezért a hierarchia magasabb szintjein levő tömbök adatait *általában* az alacsonyabb szintű tömbökre nézve is érvényeseknek kell tekintenünk. Ez más szavakkal azt jelenti, hogy a magasabb szinten egyszer már szerepeltetett adatot a hierarchia alacsonyabb szintjein *csak* akkor kell újból megadnunk, ha az adat értéke eltér a magasabb szinten megadott értéktől.

Formai szempontból a következő hierarchikus érvényességi körök képzelhetőek el:

- a magasabb szinten megadott érték általánosan érvényes (*általános* érvényesség).
- a magasabb szinten megadott érték néhány hierarchikus szinten át érvényes, majd adott szinten más értéket vesz fel, a továbbiakban pedig ez az érték lesz mérvadó (*váltó* érvényesség),
- vannak olyan adatok, amelyek csak a saját adattömbjükre érvényesek (*saját* érvényesség).

A nyilvántartásban szereplő jogszabályi szerkezet el fog térni a jogszabályok tényleges szerkezetétől. Ezt egyrészt az indokolja, hogy nem mindegyik jogszabály tartalmazza az összes hierarchikus szintet (de pl. paragrafus minden jogszabályban van), másrészt: ha a jogszabály adott szerkezeti egysége több önálló témakört is felölel (pl. 2 paragrafus a házassággal, 3 pedig a válással foglalkozik) ez nem mindig derül ki a jogszabály tényleges szerkezetéből.

A jogszabálynyilvántartás karbantartása nagyrészt az új logikai rekordok elhelyezéséből, illetve a meglevő rekordokat érintő kiegészítések/módosítások átvezetéséből fog állni. Az adatbázis feltöltése kötegelt üzemmódban történik majd.

Adatmódosításokra ritkán – elsősorban technikai okokból – lesz szükség, különleges adatbiztonsági intézkedések mellett. A tervek szerint terminálról is lehet módosítani.

A meglevő rekordokat nem fogjuk törölni, technikai okokból azonban szükség lehet bizonyos logikai rekordok archív állományba való helyezésére. Az archív állományban meg kell tartani az eredeti hierarchikus szerkezetet, maga az archiválás pedig kötegelt üzemmódban zajlik le.

5. Eddigi eredmények, további feladatok

A jogszabálynyilvántartás kidolgozásának sarkalatos pontja olyan programtermék kidolgozása, vagy felkutatása, amely alkalmas változó hosszúságú logikai rekordok és nagytömegű információ kezelésére, valamint az előzőekben említett visszakeresési szempontok kielégítésére.

Gyakorlatilag három út kínálkozott a software kiválasztására:

- az IDS adatbáziskezelő programcsomag és az MDQS visszakereső programrendszer, továbbá
- az ún. HIR programrendszer kipróbálása, valamint
- ismeretlen programtermék felkutatása.

Mindhárom úton elindultunk.

a) Az első esetben azt vizsgáltuk, hogy az ÁSZSZ HwB 66/60 típusú számítógéphez rendelkezésre álló programcsomagok milyen mértékben alkalmasak jogszabálynyilvántartási feladatok megoldására. Az eddigi kísérletek azt mutatták, hogy a programcsomagok elvileg alkalmasak a kívánt célra, de a számítógépi háttér *csak* más felhasználók *rovására* teszi lehetővé a kívánatosnak tartott 1–2 perces válaszadási határidő *folyamatos* betartását.

A kísérletekből arra lehet következtetni, hogy a számítógépi eszközöknek ez a hátrányos vonása független a kidolgozott megoldás színvonalától, illetve az adatok tömegétől, és kizárólag a választott paraméterek – elsősorban a visszakereséshez szükséges DAC üzemmód – alkalmazásának következménye.

b) A HIR programrendszert a SZÁMKI dolgozta ki, kifejezetten szöveges információ kezelésére. A rendszer alapján az elmúlt évben bemutatóra is sor került, amelynek során korlátozott számú jogszabály adatait lehetett visszakeresni néhány másodperc alatt.

c) Harmadik lehetőségként különböző (általunk) kevéssé, vagy egyáltalán nem ismert programtermékeket vizsgálunk meg abból a szempontból, milyen mértékben alkalmasak a fentiekben körvonalazott feladatok megoldására (a DATORG, a PMSZK, az ÉVM Számgép rendszerei, JOG–DOK stb., valamint külföldi programtermékek és rendszerek).

A vizsgálatoktól azt várjuk, hogy egy-egy programtermék meghatározott tulajdonságai megfelelnek céljainknak, s esetleg ezen az úton is továbbléphetünk.

Ez év októberében a HIR alapján újabb bemutatót tartunk az Igazságügyi Minisztérium képviselői előtt. A bemutató a családjogi törvényre és a hozzá kapcsolódó jogszabályokra (illetve a jogszabályok adataira) épül, s teljesítenie kell a visszakeresésekre megállapított válaszadási időket.

Ilyen esetekben ún. pszeudo-szinteket építünk be a hierarchikus szerkezetbe, amelyek segítségével egyrészt „ki lehet igazítani" a hierarchiában meglevő réseket (s egyúttal jelentősen megkönnyítjük a visszakeresés folyamatát is), másrészt pedig egyértelművé lehet tenni az adott hierarchikus szint tartalmát (ebben az esetben pszeudo-szintek közbeiktatásával jelezzük, hogy pl. bizonyos paragrafusok „ezt", más paragrafusok pedig „azt" a témakört érintik).

A jelenlegi ismereteink szerint az adatbázisstruktúra kialakításához elegendő lesz 10—15 hierarchikus szint figyelembevétele.

3. Visszakeresési szempontok, adatszolgáltatás

A visszakeresés szempontjainak kialakításakor figyelembe kellett vennünk: az adatbankot *elsősorban* jogászok fogják használni, a használati/kezelési utasításoknak tehát olyanoknak kell lenniök, hogy a számítástechnikában járatlan egyének is könnyen elsajátíthassák és alkalmazhassák őket. További lényeges jellemző: *jelenlegi ismereteink* szerint a nyilvántartás *bármelyik adata* visszakeresési szempont lehet.

Ilyen meggondolások alapján az ember-gép kapcsolatot a természetes nyelvhez közelálló nyelvűre (és természetesen párbeszédes formájúra) terveztük.

Arra is tekintettel kellett lennünk, hogy becslések szerint az információigények döntő többsége (legalább 60—70 százaléka) olyan jellegű kérdésekben ölt testet, amelyek azt tudakolják: valamely jogviszonyt milyen jogszabály rendez. Ezekben az esetekben tehát a tárgyszó a kiindulási pont, az eredmény pedig a jogszabály számjele lesz. Előreláthatólag több ezer tárgyszó kialakítására lesz szükség, s a választott feldolgozási módból adódóan a tárgyszavak maguk is hierarchikus felépítésűek lesznek.

Ha elfogadjuk az előző feltételezést (kiindulópont a tárgyszó), akkor a következő lépésben a jogszabály számjelének ismeretében az adott jogszabály *bármely* adata meghatározható (visszakereshető) és egymással kapcsolatba hozható. Ilyen típusú visszakeresés esetén tehát a kérdésben a jogszabály számjelén kívül a jogszabály valamelyik adata(i) is szerepel(nek), a válaszban pedig mindazon adatok megjelennek, amelyek az adott jogszabályhelyre — a hierarchikus adatösszefüggések alapján és az érvényesség különböző típusait is figyelembe véve — érvényesek.

A kérdések harmadik csoportja a statisztikai célokra szolgáló adatok visszakeresésére irányul.;

Minden olyan válasz statisztikai célra kell, amelyikben a jogszabályok darabszámai jelennek meg (kibocsátó szervenként, vagy jogszabálytípusonként részletezve). Meghatározott esetekben arra is szükség van, hogy a jogszabályt együttesen előterjesztő-, vagy a jogszabállyal egyetértő szerveken belül mód legyen az egyes előterjesztők, illetve egyetértők szerinti visszakeresésre is (lehetővé váljon, hogy pl. ha a KSH—PM—OT az együttes előterjesztő, a KSH) vagy a PM, vagy az OT (neve alapján is visszakereshetőek legyenek a jogszabályok).

A tervezett válaszadási idő a kérdés jellegéhez igazodik: az első két típusú kérdés esetén max. 3 perc, míg a statisztikai jellegű kérdések esetén megelégedhetünk azzal is, ha a választ néhány napon belül adjuk meg.

4. Karbantartási kérdések

A kísérletek eddig az adatbázis kialakítására és a visszakeresési módszerek feltárására irányultak, az adatbázis karbantartási folyamatára vonatkozóan csak az alapelveket dolgoztuk ki.

Irodalomjegyzék

[1] Dr. Bacsó Jenő: Az Igazságügyi Minisztérium számítógépes információrendszerének elő tes fejlesztési elgondolásai, Budapest, 1978. (kézirat)

[2] Csáki Béla–dr. Dajka Miklós–Harza Lajos–dr. Palicz Mihályné: Az országos jogszabály nyilvántartási számítógépi megoldásának kérdései. Budapest, 1979. február (tanulmány kézirat)

[3] Csáki Béla: Az országos jogszabálynyilvántartási rendszer egyes kérdései. Az államigazg tási információrendszerek számítógépi fejlesztésével kapcsolatos kutatási munkák 4. sz melléklet. SZÁMKI 1931/II., 1977. Budapest.

[4] Csáki Béla: A HIR demonstrációs rendszer Honeywell Bull számítógépen. SZÁMKI Tan mányok, 1979/3. sz. Budapest

[5] Csáki Béla–Csorba Zsuzsa: Modellkísérlet a számítógépes jogszabálynyilvántartás probl máinak vizsgálatára. Magyar Jogász Szövetség, 1977. (pályázat)

A LINEÁRIS NYELVEK ÉS A PETRI HÁLÓK KAPCSOLATA

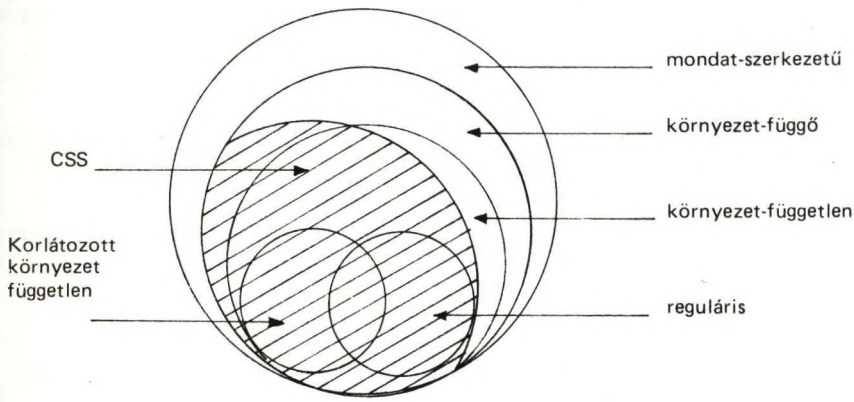
Bagyinszkiné Orosz Anna
ELTE

Vizsgálatainkhoz a J. L. Peterson [1] által bevezetett Petri-háló fogalmat használjuk, amely a következőkben foglalható össze.

Egy PH = (P, T, Σ, S, F) rendszert címkézett Petri hálónak nevezünk, ahol P a helyek nem üres véges halmaza, T az átmenetek nem üres véges halmaza, $S \subseteq P$ kezdő halmaza, $F \subseteq P$ végső helyek halmaza, a Σ véges halmaz egy ábécé, amellyel T elemeit címkézzük. Tetszőleges $t_j \in T$ $t_j = (\sigma_j, I_j, O_j)$, ahol $\sigma_j \in \Sigma$, I_j és O_j a P halmazon értelmezett listák (azaz egy hely többször is előfordulhat a felsorolásban). I_i és O_i jelenti a t_i átmenethez kapcsolódó bemenő, illetve kimenő helyeket. (A szokásos gráf-ábrázolásnál a helyeket körökkel, az átmeneteket téglalapokkal, vagy egyenes szakaszokkal reprezentáljuk, az I_j, O_j listában feltüntetett kapcsolatokat a gráfban nyilakkal jelöljük.(Azt, hogy a listában egy helyen többször szerepel, a gráfban többszörös nyillal jelöljük. A Petri háló helyeihez nem negatív egész értékeket rendelünk, s ezt súlyozásnak nevezük. (Gráf-ábrázolásnál a súlyokat a körökbe helyezett pontokkal reprezentáljuk – az i egész számnak i db pont felel meg.) A súlyozást az átmenetek ún. akciói megváltoztathatják. Egy t_j minden bemenő helyén legalább annyi súly van, mint ahányszor a hely az I_j listán szerepel. Az akció hatására az I_j listán szereplő helyeken eggyel csökken a súlyok száma, az O_j listán szereplő helyeken eggyel nő.

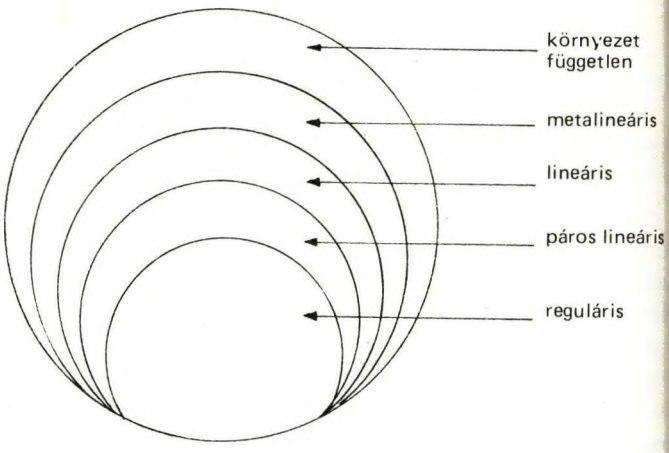
A Petri háló szimulációját egy kezdő helyre helyezett egyetlen súllyal indítjuk. A szimulációban egymás után akciót lebonyolító átmenetekből alkotott $t_{j_1}, t_{j_2}, \ldots, t_{j_n}$ sorozatokat teljes átmenet-sorozatnak nevezzük, ha a sorozatban utolsó helyen álló t_{j_n} átmenet akciója hatására valamely végső helyen egyetlen súly van, a háló többi helyén pedig nincs súly. Az ilyen teljes átmenetsorozatokhoz rendelt $\sigma_{j_1}, \sigma_{j_2}, \ldots, \sigma_{j_n}$ címkesorozatot számítási sorozatnak nevezzük, s a Petri háló által elfogadott (vagy generált) szónak tekintjük. Adott Petri hálóra vonatkozóan az összes ilyen szó halmazát a Petri háló nyelvének (vagy számítási sorozat-halmaznak) nevezzük és CSS-sel jelöljük.

Peterson megvizsgálta a CSS és Chomsky-féle nyelvosztályok, valamint a CSS és a korlátozott környezet-független nyelvek viszonyát. Eredményeit az 1. ábrán látható diagramban folaltuk össze.



1. ábra

Célul tűztük ki a környezet-független nyelvek és a CSS nyelvosztály közös részének to vizsgálatát. Ehhez a 2. ábrán szemléltetett tartalmazási viszonyokból indulunk ki, amelynek b tosítása Salomaa [2] könyvében található.



2. ábra

T-típusnak nevezünk egy lineáris nyelvtant, ha helyettesítési szabályainak halmaza nem tartal

$$\left\{B_1 \to U_1 B_2 W_1, \ldots, B_S \to U_S B_{S+1} W_S, \ldots, \quad B_r \to U_r B_1 W_r, B_S \to U'_S B_{S+1} W'_S\right\}$$

alakú részhalmazt, ahol a B_i-k nem-terminális jelek, U_i, W_i terminális sorozatok, $W_S \neq W_S$

T-típusú a nyelv, ha a generálható T-típusú nyelvtannal.
Bebizonyítjuk, hogy a T-típusú nyelvek osztályát tartalmazza a CSS.

Tekintsük a $G = (V_N, V_T, A_1, H)$ T-típusú nyelvtant, ahol

$$V_N = \left\{A_1, A_2, \ldots, A_n\right\}, \qquad V_T = \left\{a_1, a_2, \ldots, a_l\right\},$$

$$H = \left\{A_{j_1} \to P_1 A_{k_1} R_1, A_{j_2} \to P_2 A_{k_2} R_2, \ldots, A_{j_r} \to P_r A_{k_r} R_r, A_{j_{r+1}} \to P_{r+1}, \ldots, A_{j_m} \to P_m\right\}.$$

$$P_i, R_j \in V_T^* \qquad i = 1, 2, \ldots, m; \qquad j = 1, 2, \ldots, r$$

Állítás: Minden T-típusú lineáris nyelvtannal generálható nyelvhez megadható olyan 2n helyből és m+r átmenetből álló Petri háló, amely a nyelvet elfogadja.

Bizonyítás:

I. algoritmus:

1. Vegyünk fel 2n+r helyet. Címkézzük meg az $A_1, A_2, \ldots, A_n, Q_1, Q_2, \ldots, Q_n, W_1, W_2, \ldots, W_r$ szimbólumokkal. Legyen $q = 1$.

2. Tekintsük a q-adik helyettesítési szabályt:
$A_j \to P_q A_l R_q$ vagy $A_j \to P_q$ alakú. Csatlakoztassunk az A_j címkéjű helyhez egy kimenő átmenetet, amelyet a P_q címkével címkéztünk meg.
3. Ha a q-adik szabályban nincs nem-terminális, a P_q címkéjű átmenet kimenő helye Q_j lesz. Folytassuk 6./-tól.
4. Ha a q-adik szabályban van nem-terminális, a P_q címkéjű átmenetnek két kimenő helye lesz: A_l és W_q.
5. Csatlakoztassunk W_q-hoz egy kimenő áetmenetet, amelyet R_q-val címkéztünk. Ennek az átmenetnek még egy bemeneti helye lesz: Q_l; kimenő helye pedig Q_j.
6. Növeljük q-t 1-gyel. Ha $q \leqslant m$, folytassuk 2./-től, egyébként álljunk meg.

Az I. algoritmus minden $A_j \to P_q$ szabályhoz generál egy $t_q = (P_q, \{A_j\}, \{O_j\})$ átmenetet, amely a súlyok számát nem változtatja meg a hálóban. Minden $A_j \to P_q A_l R_q$ szabályhoz egy átmenet-párt generál az algoritmus:

$$t_q = (P_q, \{A_j\}, \{A_l, W_q\}), \qquad t'_q = (R_q, \{W_q, Q_l\}, \{Q_j\}).$$

Az itt generált P_q címkéjű átmenetek akciói eggyel növelik a hálóban levő súlyok számát, az R_q címkéjűeké pedig eggyel csökkentik, ugyanazon a helyen, mint amelyiken a P_q címkéjű növelte. W_q biztosítja, hogy a szabály jobb oldalának P_q és R_q terminális-sorozata ugyanannyiszor kerüljön a szóba. Azt, hogy a megfelelő helyre kerüljenek, a Q_i-ik biztosítják. Tehát a háló szimulációja során csak akkor jutunk teljes sorozathoz, ha az összetartozó P_q és R_q átmenetek azonos számú akciót bonyolítanak le.

Az I. algoritmus által generált Petri háló kezdő helye A_1, végső helye Q_1 lesz.

Most megmutatjuk, hogy a Petri háló által elfogadott minden p szóhoz tartozik egy G-beli levezetés.

Tekintsük a $\sigma = t_{i_1}, t_{i_2} \ldots t_{i_n}$ teljes átmenetsorozatot (a hozzá tartozó szimuláció az A_1 kezdő helyről indult és a Q_1 végső helyen ért véget). Legyen $p = U_1 U_2 \ldots U_n$ a σ-hoz tartozó számítási sorozat.

Az A_1 helyen megjelenő súly hatására aktivizálható állapotba kerül az összes olyan átmenet, amelynek A_1 az egyetlen bemenő helye. Ezek között van az $U_1 = P_{i_1}$ címkéjű t_{i_1} átmenet, amelyhez az $A_1 \to P_{i_1} A_{l_1} R_{i_1}$ helyettesítési szabály tartozik. A t_{i_1} átmenet akciója egy súlyt visz A_{l_1}-re és W_{i_1}-re, miközben A_1-ről egy súlyt elvisz. A következő lépésben csak azok az átmenetek kerülhetnek aktivizálható állapotba, amelyeknek A_{l_1} az egyetlen bemenő helyük, mert a W_{i_1}-hez csatlakozó R_{i_1} címkéjű átmenetek még egy bemenő helye van: Q_{l_1}, amelyre még nem került súly. A k-adik lépésben az U_k címkéjű t_{i_k} átmenet bonyolít le akciót. Ekkor három eset állhat fenn:

a) $U_k = P_{i_k}$, valamelyik $A_{j_k} \to P_{i_k}$ szabályra,

b) $U_k = P_{i_k}$, valamelyik $A_{j_k} \to P_{i_k} A_{l_k} R_{i_k}$ szabályra,

c) $U_k = R_{i_k}$, valamelyik $A_{j_k} \to P_{i_k} A_{l_k} R_{i_k}$ szabályra.

a) esetben az akció egy súlyt elvisz A_{j_k} -ról, s egy súlyt helyez Q_{j_k} -ra. Ezáltal az

$U_{k+1} = R_{i_{k-1}}$ címkéjű $t_{i_{k+1}}$ $(= t'_{i_{k-1}})$átmenetneksúlyozottá válik a második bemer

helye is. A $t_{i_{k+1}}$ átmenet $W_{i_{k-1}}$ bemeneti helye a $P_{i_{k-1}}$ címkéjű $t_{i_{k-1}}$ akciója ha

ra már súlyozottá vált.

b) esetben az akció egy súlyt elvisz A_{j_k} -ról, s egy súlyt helyez A_{l_k} -ra és W_{i_k} -ra. Ekkolye

$U_{k+1} = P_{i_{k+1}}$ címkéjű $t_{i_{k+1}}$ átmenet kerül aktivizálható állapotba.

c) esetben az akció egy súlyt elvisz W_{i_k} -ról és Q_{l_k} -ról, és egy súlyt visz Q_{j_k} -ra. Ezáltal

vizálhatóvá válik az $U_{k+1} = R_{i_s}$ címkéjű $t_{i_{k+1}}$ $(= t'_{i_s})$ átmenet, amelynek W_{i_s} és Q_{j_k}

bemenő helyei $(s < k-1)$.

Világos, hogy a c) esetben újabb átírási szabályokat nem alkalmazunk, csak a korábba alkalmazott szabályok jobb szélét vesszük figyelembe.

Az U_n címkéjű t_{i_n} átmenet akciója után Q_1-n van egy súly, s a hálóban nincs több akti

ható átmenet. $U_n = R_{i_1}$ szükségképpen teljesül (az elsőként akciót lebonyolító t_{i_1} átmenethe

rendelt $A_1 \rightarrow P_{i_1} A_{l_1} R_{i_1}$ szabályra vonatkozóan).

Látható, hogy a σ teljes sorozathoz rendelt p szó előállításában n db átmenet szerepelt, amelyek között megegyezőek is lehetnek, s az átmenetekhez (n+1) /2 db helyettesítési szabál rendelt az algoritmus.

Tehát minden σ átmenetsorozathoz tartozik egy

$$(A_1 =) A_{j_1} \rightarrow P_{i_1} A_{l_1} R_{i_1}, (A_{l_1} =) A_{j_2} \rightarrow P_{i_2} A_{l_2} R_{i_2}, \ldots$$

$$\ldots, (A_{l_{\frac{n-1}{2}}} =) A_{j_{\frac{n+1}{2}}} \rightarrow P_{i_{\frac{n+1}{2}}}$$

levezetés.

Megfordítva is igaz, minden $p \in L(G)$ szóhoz tartozik egy σ teljes átmenetsorozat.

Legyen $p = P_{i_1} P_{i_2} \ldots P_{i_{n-1}} P_{i_n} R_{i_{n-1}} \ldots R_{i_1}$. A p szó pontosan akkor tartozik L(G)-h

ha megadható egy G-beli levezetés. Tegyük fel, hogy a levezetésben a következő átírás szabályok vesznek részt:

$$(A_1 =) A_{j_1} \rightarrow P_{i_1} A_{l_1} R_{i_1}, (A_{l_1} =) A_{j_2} \rightarrow P_{i_2} A_{l_2} R_{i_2}, \ldots,$$

$\ldots, (A_{l_{n-2}} =) A_{j_{n-1}} \rightarrow P_{i_{n-1}} A_{l_{n-1}} R_{i_{n-1}}$,

$(A_{l_{n-1}} =) A_{j_n} \rightarrow P_{i_n}$.

Az első szabály alkalmazásakor egy súlyt helyezünk a Petri háló A_1 kezdő helyére. Az aktivizálható átmenetek között van a P_{i_1} címkéjű t_{i_1} átmenet is. t_{i_1} akciója után csak A_{l_1} és W_{i_1} helyek lesznek súlyozottak. Csak azok az átmenetek kerülnek aktivizálható állapotba, amelyeknek A_{l_1} az egyetlen bemenő helyük. Ezek között van a P_{i_2} címkéjű t_{i_2} átmenet. Az (n−1)-edik szabály alkalmazásakor egy súly van az $A_{j_{n-1}}$ címkéjű helyen, s ezáltal aktivizálhatóvá válik a $P_{i_{n-1}}$ címkéjű $t_{i_{n-1}}$ átmenet, a $t_{i_{n-1}}$ akciója hatására súlyozottá válik az $A_{l_{n-1}} = A_{j_n}$ és $W_{i_{n-1}}$ címkéjű j hely. Ekkor viszont az aktivizálható átmenetek között van a P_{i_n} címkéjű t_{i_n} átmenet, amelynek egyetlen kimenő helye a Q_{j_n} címkéjű hely. Amint Q_{j_n} súlyozottá vált, aktivizálható állapotba került az $R_{i_{n-1}}$ címkéjű $t'_{i_{n-1}}$ átmenet, amelynek $W_{i_{n-1}}$ bemenő helye az (n−1)-edik lépésben már súlyozottá vált. A $t'_{i_{n-1}}$ akciója súlyt visz a $Q_{j_{n-1}}$ helyre, s ezáltal az $R_{i_{n-2}}$ címkéjű átmenet aktivizálhatóvá válik. Végül a t'_{i_1} átmenet akciója a hálóban levő utolsó súlyt Q_1 -re viszi.

Tehát a $p = P_{i_1} P_{i_2} \ldots P_{i_{n-1}} P_{i_n} R_{i_{n-1}} \ldots R_{i_1}$ L (G) -beli szóhoz a Petri háló

$\sigma = t_{i_1} t_{i_2} \ldots t_{i_{n-1}} t_{i_n} t'_{i_{n-1}} \ldots t'_{i_1}$

szimulációja tartozik.

Ezzel az állítás bizonyítását befejeztük.

Metalineáris nyelvtanokra nem alkalmazható ebben a formában az algoritmus, mert a helyettesítési szabályok jobb oldalán egymás mellett fellépő nem-terminálisokat ezzel a módszerrel nem tudjuk kezelni. Könnyű belátni, hogy a T-típusú metalineáris nyelvek is a CSS részhalmazát alkotják, mivel Peterson megmutatta, hogy a CSS zárt az egyesítés és az összefűzés műveletére nézve, s az is ismeretes, hogy a metalineáris nyelvek a lineáris nyelvekből az említett két művelet segítségével előállíthatók.

Reguláris nyelveket elfogadó Petri háló konstruálására egyszerűbb algoritmust lehet megadni.

Ezúton szeretnék köszönetet mondani Makay Árpádnak a lektorálás során tett értékes észrevételéért.

Irodalom

[1] Peterson, J. L.
Computation Sequence Sets.
Journal of Computer and System Sciences 13, 1–24 (1976)

[2] Salomaa, A.
Formal languages
Academic Press New York and London 1973.

PROGRAMTERVEZÉS LOGIKAI ALAPOKON

Balogh Kálmán—Sántáné Tóth Edit—Szeredi Péter
NIM IGÜSZI SZKI SZKI

Az itt közölt ismertetés egy kidolgozás alatt lévő, VDM-szerű tervezési rendszer ala it mutatja be. A fejlesztést az SZKI és a NIM—IGÜSZI munkatársai végzik, az OMFB ál mogatott Tudományos Kutatási Munkák 3. fejezete keretében, a SZÁMKI megrendelésé Részletesebb ismertetést ld. a [D 33] anyagban.

1. Bevezetés

Nagyobb software rendszerek készítése bonyolult, fáradságos, nehezen átlátható fo mat. Problémái megoldására a feladatok növekedésével párhuzamosan körülhatárolódott témakör a számítástechnikán belül, amely a rendszertervezés és -készítés technológiájával módszereivel és eszközeivel — foglalkozik.

Igen sok, a tervezést lehetővé tevő, ill. segítő módszer, ajánlás látott napvilágot az években; ezeket elsősorban a gyakorlat szakemberei fejlesztették ki. Az 1978-ban írott [tanulmány 6. fejezete 30 ismertebb tervezési módszert hasonlít össze, feladatkörönként készítés egyes fázisaiban való alkalmazhatóságuk, automatizáltságuk, gépfüggetlenségük, e tőségük és a keletkezett terv (ill. program) ellenőrizhetősége alapján. A vizsgált módszere a *hagyományos* felfogást követik, amely szerint a software-előállítás folyamata több egy épülő, de célkitűzéseiben és módszerében egymástól igen eltérő szakaszra, fázisra bomlik talában a következő fázisokat különböztetik meg: követelménydefiniálás vagy specifikálás vezés, kivitelezés, tesztelés, üzemelés. Az egyes fázisok eredményeként előálló terméktől strukturáltság mellett általában azt is megkívánják, hogy minél jobban ellenőrzött és dok tált legyen.

Egy-egy vizsgált módszer a fázisok közül egyet vagy néhány egymás utánit támoga módszer mellé adott leíró nyelv absztrakciós szintje és rugalmassága szerint.

E módszerek a magasabbszintű nyelvekből, blokkdiagramnyelvekből, olykor a mate kai logika nyelvéből vett eszközöket nyújtanak felhasználóiknak. Legtöbbjük elterjedten nálatos, igen sok közülük számítógéppel is támogatott. Egyik sem lép fel azonban olyan nyel, hogy a software-előállítás során egyetemlegesen használható, magasfokúan automati technológiai rendszere legyen.

Napjainkban tanúi lehetünk annak a törekvésnek, hogy a kutatók — a struktúrált ramozás elvét továbbra is követve — elméleti diszciplinák (elsősorban a matematikai logi az algebra) felé fordulnak azt remélve, hogy olyan *korszerű* és egzakt módszereket és es zöket tudnak kimunkálni, melyek a software-előállítás sürgető problémáit megoldanák. I nek előfutáraiként tekinthetjük például:

— az algebrai közelítésű CLU [Li, 75] és CLEAR [Bu, 77] nyelveket; előbbi klass példája absztrakt adattípusok korszerű megadását és kezelését biztosító programozási ny utóbbi eszközöket kíván nyújtani algebrai elméletek struktúrált leírására, módot nyújtva a felhasználói problémák egzakt specifikálására — még a problémák megoldását jelentő p ram kifejlesztése előtt;

— az algebrai és logikai megalapozású IOTA [Na, 78] nyelvet és rendszert, amely adat- és eljárásabsztrakciós lehetőségeket nyújt specifikációk és programok jólstruktúrált, rarchikusan kapcsolódó, verifikált modulokból való felépítéséhez. Hasonló e nyelvhez a

gikai eszköztár felhasználásával kifejlesztett, korszerű ALPHARD [Wu, 76] programozási yelv is;

– a logika egy résznyelvét magában foglaló, sokat ígérő PROLOG [Sz, 77] nyelvet, amely a matematikai formalizmushoz közelálló nyelven egyszerre segíti a programelőállítás pecifikálási és kivitelezési fázisait;

– a CIP rendszer [Ba, 78] ún. széles spektrumú nyelvét, amely lehetővé teszi a felhasználónak, hogy a programelőállítás specifikálási és kivitelezési fázisaiban ugyanazon a nyelven rja le a feladatot. Az egyes tervezési lépések során születő feladatleírások a nyelv által nyújtott helyességmegőrző transzformációk alkalmazásával vihető át egymásba;

– a denotációs szemantikának a felhasználóhoz közelített első átfogalmazásaként a VDM Bj, 78] nyelvet, amelynek formalizmusa a VDL szemantikaleíró nyelvből lett kifejlesztve.

Itt kell megemlítenünk egy hazai kísérletet, a SAM módszert és eszközkészletet [Dö, 76], mely a VDL szemantikaleíró nyelv és egy természetes nyelv együttesével, mint eszközkészlettel kívánja programcsaládok absztrakt, jólstruktúrált, hierarchikusan kapcsolódó rétegekből álló, étegenként verifikált leírását elősegíteni. (Logikai alapú változatát ld. [Dö, 77]-ben.)

Egy gyakorlatban is használható, logikai alapú, a VDM filozófiáját és jelölésmódját követő tervezési nyelv kifejlesztése már az elmúlt évben megindult az ANSWER rendszer [Be, 78] eretében; ennek eredményéről számolnak be a [D 27] és [D 29] munkák. A tavalyi tapasztalatokat – és a közben megismert néhány korszerű nyelv megoldásait – figyelembevéve dolgozunk tovább a tervezési nyelv továbbfejlesztésén, az ezt támogató rendszer körvonalazásán s a hozzájuk kapcsolódó módszertan kimunkálásán. Mindezeket együttesen LDM-nek (Logic-ased Development Method) nevezzük a továbbiakban.

Ismertetésünk további részében foglalkozni kívánunk a programelőállító rendszerek néhány alapkérdésével, megvilágítva az általunk kifejlesztendő rendszer elvi alapjait. Ezután bemutatjuk tervezési nyelvünk néhány lényegesebb vonását, majd – illusztrációként – megadjuk egy egyszerű assembler legmagasabb szintű tervváltozatának leírását.

2. Programelőállító rendszerek néhány alapkérdése

2.1 A hagyományos és a korszerű programelőállítás fő célkitűzései

Egy software-termék időben egymás után hozott tervdöntések során, *lépésenként* készül l. A Bevezetőben említett, *hagyományos* gyakorlat a software-előállítást több egymásra épülő, e célkitűzéseiben és módszereiben egymástól igen eltérő szakaszra, fázisra bontja (ezek közül legutolsó, az üzemelés már nem tartozik szorosan a programelőállításhoz).

Ennek megfelelően az egyes fázisok számára más-más eszközkészlet került kifejlesztésre. gy fontos kivétel az SADT (Structured Analysis and Design Technique, lásd a [Ro, 77] leírást), amely – a funkcionális dekompozíció elvét követve – sajátos diagramjelöléseivel egyszerre segíti a követelménydefiniálási és tervezési fázisokat – informális verifikálási lehetőséget nyújtva. Ismeretes azonban, hogy e módszerek alkalmazása esetén mennyi kényelmetlen, ehezen feldolgozható visszacsatolásra van szükség egy-egy bonyolultabb software-termék előállítása során. Amíg a visszacsatolás egyetlen fázison belül történik, általában nincs nagyobb robléma. Az esetben azonban, ha a hiba nem hárítható el a fázison belül, a hiba tényleges kának – és helyének megtalálása, majd a hibás döntést hozó fázistól kezdve szükséges korekciók végrehajtása igen megnöveli a készítésre fordított munkát, időt, költségeket. Súlyosbítja a helyzetet, ha – mint ahogy az eléggé elterjedt a gyakorlatban – fázisonként más-más szakember (-csoport) dolgozik.

Kívánatosnak látszik e hagyományos, fázis-szemléletet úgy *korszerűsíteni,* hogy az előbbi, kényelmetlenül kezelhető visszacsatolások jórésze kivédhető legyen. Lehet olyannak is tekinteni a software-előállítás folyamatát, amelynek során a feladatmegadást jelentő valamely leírás-

ból kiindulva, *„hasonló" lépések sorozatával* jutunk el a feladat egy lehetséges megoldását lentő programhoz. A részünkről korszerűnek nevezett nyelvek ill. rendszerek az egyes elő állítási lépéseken belül szándékoznak minél több fázist megvalósítani (a tesztelés helyett, ve mellett a verifikálást is).. A cél itt az, hogy egy-egy lépés megtétele után előálló „félké termék-elem" önmagával — és környezetével — a lehető legnagyobb mértékig összehangolt zonyított, esetlegesen próbafutásoknak már alávetett, jól dokumentált produktum legyen. igény vezérelt minket is az LDM fejlesztése során.

2.2 A korszerű programelőállítási módszerek és az LDM

Ebben a pontban vizsgálat tárgyává tesszük a már korábbiakban felsorolt korszerű szereket, ill. rendszereket, majd megkíséreljük ezekkel összevetni az LDM kifejlesztésénél nyesülő fontosabb alapelveket.

A programelőállítási rendszereket először abból a szempontból fogjuk vizsgálni, hogy specifikálási, tervezési és kivitelezési fázisok közül melyeket milyen mértékben támogatják. Megállapíthatjuk, hogy míg a CLEAR elsősorban a specifikálási, addig a CLU a kivitelezési fázist segíti. Az ALPHARD és IOTA rendszerek a specifikálási és kivitelezési fázist egyaránt támogatják, és — ami a legfontosabb — ezek összhangját is biztosítani kívánják. Az eddig említett négy rendszerben közös az, hogy a tervezési fázist nem támogatják, bár az általuk biztosított hierarchikus struktúrálási és modularizálási lehetőségek egyes tervezési módszerek nél elengedhetetlenek lehetnek. Véleményünk szerint azonban a tényleges tervezési fázist az előbbi lehetőségek nem pótolják, ezek annak csak szükséges feltételei. Hangsúlyozni szeretnénk még, hogy az előbbi rendszerek a specifikálási és kivitelezési szintre külön-külön nyelvet vezetnek be.

A Bevezetésben említett módszerek másik fontos csoportja — a VDL, VDM és részben a SAM — a programkészítési folyamatot absztrakt matematikai objektumok világában vetíti ki, és egységes matematikai eszköztárat használ e folyamat egyes lépéseinek megfogalmazására. Így egyetlen, közös formalizmus segítségével lehet bennük megfogalmazni a software-termék specifikációját, absztrakt kivitelezését, és a két végpont közötti átmenet biztosítására — célszerűen több lépésben — a termék tervváltozatait. Ezek a rendszerek nem foglalkoznak a gépi támogatáshoz szükséges formalizmussal, sem a modularizálás, hierarchikus struktúrálás formális eszközökkel való támogatásával.

Az LDM esetében az előbb ismertetett közelítésmódok előnyös tulajdonságait kívánjuk egyesíteni. Ez a célkitűzés két ponton is követel alapvetően újszerű megoldásokat. Egyrészről a tervezési nyelv olyan *konstruktív formalizálását,* amely már egy tervszinten (sőt akár már a specifikálási szinten is) biztosítja a gépi végrehajtás (futtatás) lehetőségét. Ez — véleményünk szerint — nagymértékben hozzájárulhat a hibás specifikáció vagy helytelen tervezési döntés korai felismeréséhez. Másrészről ismeretes, hogy míg az IOTA, ALPHARD-szerű rendszerek a funkcionális dekompozíció irányában struktúrálják a leírást, addig a CIP-, VDM-szerű közelítésben a realizációhoz való közeledés, algoritmizálás, konkretizálás szintje szerint tagolódik a leírás. Tervező rendszerünkön belül mindkét felbontási irányt támogatjuk, ami leírások összetettebb struktúrálását igényli (ennek megoldásáról a 3. pontban szólunk).

Tervező rendszerünk célkitűzéseit tehát a következőkben foglalhatjuk össze:

— a programelőállítási folyamat *mindegyik fázisát* — a specifikálási fázistól kezdődően támogatni kívánjuk, a hangsúlyt a több lépésben való tervezésre helyezve (mint pl. a CIP-ben);

— matematikai alapokra épülő *egységes formalizmust* (nyelvet) kívánunk használni minden egyik fázisban (mint pl. a VDM-ben);

— nyelvünk legyen teljesen formalizált, biztosítva a tervek *gépi kezelését* — mechanikus verifikálását (mint pl. IOTA-ban), valamint gépi futtathatóságát;

– a nyelv és a rendszer támogassa a tervek *hierarchikus struktúrálását* mind a *funkcionális dekompozíció* irányában (ld. pl. IOTA), mind pedig a *realizálás* (konkretizálás, implementálás) irányában (mint pl. a VDM-ben).

Befejezésül foglalkoznunk kell az LDM matematikai alapjául szolgáló formalizmus kiválasztásával. A matematikai logikát (általában a klasszikus elsőrendű predikátumkalkulust) széles körben használják specifikációs nyelvként. Ugyanakkor többéves tapasztalataink igazolják, hogy ugyanez a nyelv (illetve ennek valamely megszorítása, pl. a PROLOG) jól használható nagyon magas szintű programozási nyelvként is, tehát alkalmas a kivitelezési fázis leírására is. Ezek indokolják, hogy a rendszer alapjaként az elsőrendű predikátumkalkulust, pontosabban ennek egy konstruktív résznyelvét válasszuk.

3. A tervezési nyelv bemutatása

Tervezési rendszerünk alapját képező tervezési nyelv előzetes leírását a [D 33] tanulmányban adtuk meg. A következőkben – a rendelkezésünkre álló kereteken belül – röviden, informális ismertetését adjuk a nyelvnek. Először ismertetjük a tervek szerkezetét meghatározó nagyobb nyelvi egységeket (3.1 pont), majd vázoljuk, milyen lehetőségeket nyújt a nyelv a legkisebb nyelvi egységen, az ún. tervrészen belül (3.2 pont).

3.1 A tervek szerkezete

Tervezési módszerünk alapgondolata az, hogy a feladat elsődleges megfogalmazásául szolgáló specifikáló világot fokról-fokra, szintenként közelíthessük a feladat megoldási teréül szolgáló, implementáló, algoritmikus világhoz. E realizálási lépések során keletkező, egymással kapcsolódó szintek (*level*-ek) sorozata alkotja egy feladat tervét. Minden szinten egy teljes tervváltozatot kell megadnunk. Egy szint a felülről lefelé finomító (funkcionálisan dekomponáló) szemléletben született tervdöntések szerint, egymással hierarchikusan kapcsolódó csoportokra (*group*-okra) bomlik. Minden egyes csoport tehát a demompozíció egy lépése során születik. Az egyes csoportok – egymáshoz már szabadabban kapcsolódó – tervrészekre (*part*-okra) tagolódnak.

Mind a szintek, mind a csoportok a megfelelő alapszóval (*level* ill. *group*) kezdődnek, amelyet a szóbanforgó egység által vállalt funkcióra utaló név követ. Mindkét egység azután „bejelenti" az általa definiálásra (*defines*) vállalt tartományokat és/vagy relációkat; ezek típuselőírása az egységen belüli definiáló előfordulásuk után található meg. Ezután az egységet alkotó elemek (szinten belül a csoportok ill. csoportokon belül a tervrészek) felsorolása következik. Ha egy alkotóelem leírása egy – az előző szinten megadott – elem változatlan átmásolását jelentené, elég csak utalni *(ref)* annak korábbi deviníciójára.

Az eredeti egységen belül nem definiált, csak igényelt *(needs)* tartományok és/vagy relációk – elvárt típuselőírással ellátott – felsorolása zárja le a szintek, ill. csoportok érdemi mondanivalóját.

Az egy csoporthoz tartozó tervrészekben szereplő nevek hatásköre az őket tartalmazó csoport – kivéve a definiálásra vállalt és igényelt neveket, amelyek a csoporton kívül is használhatók. Ezek a nevek határozzák meg a csoportnak, mint fekete doboznak a szintet alkotó többi csoporttal való érintkezési pontjait. Ezek alapján egy alárendeltségi reláció deviniálható a csoportok között: egy csoportnak alárendeltje egy másik, ha az előbbi csoport által igényelt nevek közül az utóbbi legalább egy név definiálását vállalja. Egy szint hierarchikus struktúráját ezek után a szintet alkotó csoportok közötti alárendeltségi reláció határozza meg. Megjegyezzük, hogy a korszerű nyelvek modul-fogalmának tervezési nyelvünkön belül a csoport fogalma felel meg.

3.2 Tervek építőelemei, a tervrészek

A tervrészek egy-egy összetartozó részfeladatot írnak le – szabványos vagy más te szekben definiált fogalmakra visszavezetve. Minden egyes tervrész tartományokat és/vagy tományokra vonatkozó műveleteket definiál. Egy-egy *tartománydefiníció* objektumok egy mazát, vagyis típust definiál, a *műveletek* megadása pedig a megfelelő tartományok fele lációk és függvények definiálását jelenti.

A *tartomány*ok szintaktikus szabályok formájában adhatók meg. A szintaktikus sz jobb oldalán alaptartományokat, már definiált tartományokat és összetett tartományoka ző operátorokat használhatnak.

Az alaptartományok a következők:

number – az egész számok tartománya;

basic – szerkezet nélküli objektumok végtelen tartománya (olyankor használjuk, h terv adott szintjén az objektumok szerkezete érdektelen számunkra);

nil – az egyetlen, „üres" objektumból álló tartomány.

Az összetett tartományok a következők (jelöljenek t, t_1, ... tn tetszőleges tervtartom kat, „a" tetszőleges nevet, p egyargumentumú relációnevet):

list t – t-beli objektumok véges listáiból álló tartomány

set t – t-beli objektumok véges halmazaiból álló tartomány

struct (t_1,...t_n) – olyan struktúrák (rekordok) tartománya, amelyek i-edik komponens beli objektum;

map (t_1, t_2) – olyan leképezések tartománya, amelyek t_1 egy véges részhalmazából be képeznek:

t *named* a – olyan tartomány, amely „a" névvel ellátott t-beli objektumokból áll

t_1; t_2 – az a tartomány, melynek elemei t_1-beli vagy t_2-beli objektumok;

t *suchthat* p – a t tartomány azon objektumaiból álló tartomány, amelyek a p relá nak eleget tesznek.

Például az alábbi tartománydefiníció

s-prog::= *list* s-stmt *suchthat* wf-s-prog

azt jelenti, hogy s-prog olyan s-stmt típusú objektumok listája, amely wf-s-prog tulajdon vagyis jólformált forrásprogram.

A tartomány-képző operátoroknak megfelelően a nyelv tartalmaz kiválasztó, felépítő és tettobjektum-bejáró alapműveleteket, relációkat ill. függvényeket. Pl. ha L valamilyen ob tumok listája vagy halmaza, akkor az „E *elem* L" reláció pontosan akkor igaz, ha az E jektum eleme az L-nek. A „*hd* L" függvény eredményül az L lista fejét adja ki.

Struktúrák komponenseit a struktúranevekkel elnevezett függvényekkel választhatjuk ki.

s-stmt::= *struct* (labels, mnem, addr), és S „s-stmt" típusú, akkor „labels (S)" S el komponensét adja.

Összetett objektumok bejárását beépített műveletek könnyítik meg. Az SL lista be sával fogalmazható meg pl. a következő jólformáltságot leíró logikai kifejezés

forall S *elem* SL *holds* wf (S, SL), azaz:

minden S eleme SL-re fennáll, hogy S jólformált SL-re nézve.

Hasonlóan, ha SL forrásutasítások listája, és a „tr-stmt (S)" függvény felelteti meg forrásutasításoknak a tárgyutasításokat, akkor a tárgyprogramot, mint tárgyutasítások lis

list (tr-stmt (S) *forall* S *elem* SL)

implicit listamegadással írhatjuk le.

A *műveletdefiníciók* (reláció- illetve függvénydefiníciók) logikai állítások formájába hatók meg. A relációdefiníció baloldalán a definiálni kívánt reláció szerepel (a relációné „formális paraméterekkel" ellátva), jobboldalán pedig szabványos- és felhasználói reláció

felépített szokásos logikai kifejezés áll. A két oldalt az *iff* (akkor és csak akkor) logikai művelettel kell összekapcsolni. Példa relációdefinícióra:

wf-s-prog (SL) *iff*
forall S *elem* SL *holds* wf (S,SL).

A függvénydefiníciókat egyenlőség alakjában kell megadni, amelynek baloldalán a definiálni kívánt függvény, jobboldalán pedig a függvény értékét definiáló kifejezés áll.

Mind a reláció – mind a függvénydefiníciókat egy típuszáradékkal kell ellátni, amely a paraméterek illetve az eredmény tartományát írja elő. Példa típuselőírással ellátott függvénydefinícióra:

tr-prog (SL) =
list (tr-stmt (S) *forall* S *elem* SL)
type total fct s-prog → t-prog.

Az LDM célja az is, hogy támogassa az implementáláshoz közelítő tervezési szintek megfogalmazását. Ehhez szükséges az algoritmikus programozás elemeinek bevezetése. Ezt úgy kívánjuk biztosítani, hogy bizonyos, kötöttebb alakú műveletdefiníciónál megengedjük a végrehajtás algoritmusát előíró alak használatát. Az ilyen műveletdefiníciókban a hagyományos globális változók is használhatók.

A végrehajtás algoritmizálása mellett az alacsonyabb szintek feladata, hogy az adatstruktúrákat konkretizálja, közelebb hozza a hatékony realizációhoz. Ezt a konkrétabb tartományfajták használatával érhetjük el, pl. ha egy fogalom leírására a felső tervszinteken halmaztartományt használunk, középső szinteken áttérhetünk a listával való reprezentálásra, alsó szinteken pedig algoritmizálhatjuk a lista előállítását úgy, hogy ciklusban alkalmazzuk a megfelelő listafelépítő műveletet.

4. Az LDM alkalmazása egy egyszerű példán

Az LDM előző pontban adott rövid áttekintése után vizsgáljuk meg egy egyszerű assembler LDM-tervét. A közölt ábra a terv első három szintjének szerkezetét mutatja be.

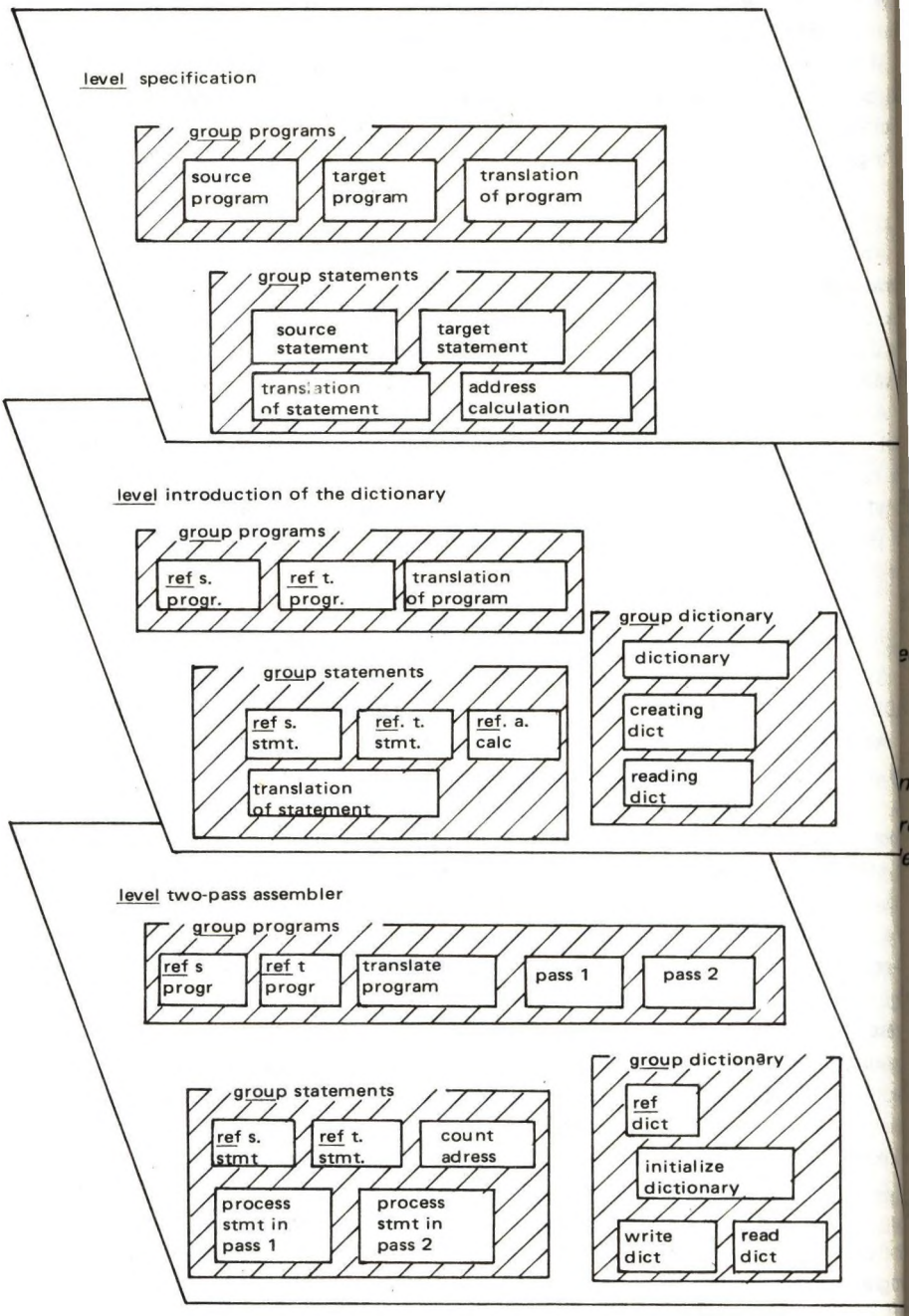
Az első szint a feladat specifikációját adja meg (innen a szint neve: „specification"): A „programs" nevű csoport elmondja, hogy egy jólformált forrásprogram jólformált utasítások listája, a tárgyprogram pedig tárgyutasítások listája; kijelenti továbbá, hogy egy jólformált forrásprogram lefordításának eredménye nem más, mint a megfelelő forrásutasítások lefordítása során előálló tárgyutasításlista. A szint „statements" nevű csoportja megadja a forrás- és tárgyutasítások szerkezetét, továbbá – egy megfelelő címkiszámító függvényre támaszkodva – definiálja, miben áll egy utasítás lefordítása (a forrásprogram miatt „környezetre" nézve). E szint LDM-programja az ábra után található. (A megértés megkönnyítése céljából néhány kulcsfontosságú vagy bonyolultabb egység mellett megtalálható annak „kiolvasott" formája). A program átnézése után látható, hogy a terv ezen első szintje meghatározza ugyan a fordítás eredményét, és tartalmaz algoritmikus elemeket (egy forrásprogram fordításának eredményét a lefordított forrásutasításokból építi fel), a terv mégsem hatékony. Vegyük észre ugyanis, hogy a címkék minden egyes alkalmazó előfordulásánál a címkiszámító függvény a forrásprogramból újra és újra kikeresi az egyetlen („uniquely-defined") definiáló előfordulást.

A hatékonyabb címkiszámítás végett a második szinten bevezetjük a szótár segédfogalmat a „dictionary" csoportban – még csak deklaratívan előírva, mit értünk szótárkezelésen. A „statements" csoport még szintén csak deklaratívan rendelkezik egy utasítás lefordításáról. A programok és utasítások struktúrája ugyanaz marad, mint az első szinten; az emiatt hivatkozott (*ref*) tervrészek nevét – helyhiány miatt – az ábrán rövidítettük.

A harmadik szint a második szinten előírt szótárazó fordítás egy alternatíváját, a kétmenetes fordítás tervét írja le. Ebben mind a tárgyprogram, mind a szótár felépítését algorit-

mikusabbá tesszük megfelelő ciklusutasításokkal váltva ki az implicit lista- illetve leké felépítő műveleteket, a címszámítást pedig egy címszámláló bevezetésével algoritmizálj

A következőkben bemutatjuk az ismertetett három szint felépítését mutató ábr követően pedig az első szint részletes LDM-leírását.



Ábra: Az egyszerű assembler első három tervszintjének szerkezete

```
level specification.
defines tr-prog.

    group programs.
    defines s-prog, t-prog, tr-prog.
      part source-program.
        s-prog::= list s-stmt
                  suchthat wf-s-prog.
        wf-s-prog (SP) iff
          forall S elem SP holds
                  wf-s-stmt-in-s-list(S,SP).
    endpart.

    part target-program.
      t-prog::= list t-stmt.

    endpart.
    part translation-of-program.
      tr-prog(SP)=list(tr-stmt(S,SP)
      forall S elem SP)

      type total fct s-prog → t-prog.

    endpart.
needs s-stmt: domain, t-stmt: domain,
    tr-stmt: type total fct (s-stmt,
    s-prog) →t-stmt, wf-s-stmt-in-s-list:
    type pred (s-stmt, list s-stmt).
endgroup.

group statements.
defines s-stmt, t-stmt,
    tr-stmt, wf-s-stmt-in-s-list.
    part source-statement.
      s-stmt::= struct (labels, mnem, addr).

      labels ::= set name.
      addr ::=name.
      wf-s-stmt-in-s-list (S,SL) iff
        forall NMelem (labels(S)
        union set (addr(S)))
        holds uniquely-defined-in-s-list
        (NM,SL)
      type pred (s-stmt, list s-stmt).
```

Az „assembler" definíciójának feladatspecifikáló változata (szintje) következik, amelynek lényege a tr-prog függvény meghatározása.

A forrásprogram forrásutasítások listája: olyan, amely jólformált forrásprogram.
Egy SP jólformált forgásprogram, ha minden SP-beli S utasításra fennáll, hogy S jólformált utasítás az SP listára nézve.

Egy SF program lefordított alakja az a lista, amelynek elemeit úgy kapjuk, hogy az SP-beli S utasításokat az SP „környezet" figyelembevételével lefordítjuk.

tr-prog minden forrásprogramhoz egyetlen tárgyprogramot rendel.

Egy forrásutasítás olyan struktúra, amelynek a részei: cimkék, mnemonik, címrész.

A cimkék: nevek véges halmaza.
A címrész: egy név.
Egy S utasítás akkor és csak akkor jólformált az SL környezetben, ha az S cimkéiként és címrészeként szereplő összes NM név egyértelműen van definiálva SL-ben.

```
        uniquely-defined-in-s-list
        (NM,SL) iff
          exists! S elem SL
              suchthat NM elem labels (S)
        type pred (name, list s-stmt).
    endpart.
    part target-statement.
        t-stmt::=struct (instr, num-addr).

        num-addr::= number.
    endpart.

    part translation-of-statement.
        tr-stmt (struct (LS,M,NM),SP)=
          struct (tr-mnem(M), tr-name(NM,SP))
        type total fct (s-stmt, s-prog)→t-stmt.

    endpart.
    part address-calculation.
        tr-name (NM,SP)=
          the I elem ind SP
              suchthat NM elem labels (SP[I])
        type total fct (name, s-prog) →num-addr.
    endpart.
  needs mnem: domain, instr: domain,
    tr-mnem : type total fct mnem → instr.
    name : domain.
  endgroup.
endlevel.
```

Egy NM név egyértelműen van definiálva ben akkor és csak akkor, ha pontosan eg olyan S eleme létezik az SL-nek, amelyr eleme S cimkéinek.

Egy tárgyutasítás olyan struktúra, amely sításkódból és numerikus címrészből áll.

A numerikus címrész: egy szám.

Egy M mnemonikú, NM címrészű utasítá fordított alakja SP-ben egy olyan tárgyut tás, amelynek utasításkódja M lefordított ja, címrésze pedig NM-nek az SP környez lefordított alakja.

Egy NM név lefordított alakja SP-re nézv latív címe SP-ben): az az I egész szám az elemeinek sorszámai (indexei) által alkot tából, amely SP azon elemének sorszáma amelynek cimkéi között szerepel NM.

3a, 78] F.L. Bauer et al.: Towards a Wide Spectrum Language to Support Program Specification and Program Development. SIGPLAN NOTICES, 1978. dec. No. 12. pp. 15-24.

Ba,78] Balogh Kálmán: Egy logikai módszer programok szemantikus tulajdonságainak bizonyítására. Egyetemi doktori disszertáció, 1979.

Be, 78] Bedő Árpád, Langer Tamás: A programozás technológiája az ANSWER operációs rendszerben. Szeged, 1978. november. Programozási Rendszerek '78 konferencia kiadványa, I. kötet. 84-93 old.

Bj, 78] Bjorner D., Jones, C.B. (editors): The Vienna Development Method: The Meta-Language. Lecture Notes in Computer Science (ed. by G.Goos and J.Hartmanis) 61. Springer-Verlag, Berlin Heidelberg New York 1978.

Bu, 77] Burstall, R.M. Goguen,J.A.: Putting Theories together to make Specifications. Proceedings of 5th International Joint Conference on Artificial Intelligence, Cambridge, MASS. 1977. pp. 1045-1058.

D 27] Balogh Kálmán, Szeredi Péter: A PROLOG nyelv alkalmazása software és hardware objektumok tervezésére (Esettanulmány) VI. kötet Tervezés meta-nyelven. SOFFTECH D27, 1978. augusztus. Készült a NIM–IGÜSZI-ben, a KSH–OSZI megrendelésére, az SZKCP CF–31 célfeladat keretében.

D 29] Balogh Kálmán, Sántáné-Tóth Edit, Szeredi Péter: Software tervező rendszerrel kapcsolatos feladatok összefoglalása. SOFFTECH D29, 1978. november, Készült a NIM IGÜSZI-ben, a KSH-OSZI megrendelésére, az SZKCP CF–31 célfeladat keretében.

D 33] Szeredi Péter, Sántáné-Tóth Edit, Balogh Kálmán: Logikai alapú tervező nyelv előzetes specifikációja. SOFFTECH D33 1979. június. Készült az SZKI-ban és a NIM–IGÜSZI-ben, a SZÁMKI megrendelésére.

Dö, 76] Dömölki Bálint, Sántáné-Tóth Edit: Formal Description of Software Components by Structured Abstract Models. Computational Linguistics and Computer Languages XI. k. 1976. pp. 31–72. (Magyar változata megjelent az Információ Elektronika 1977. XII. évf. 4. számának 203–211 oldalain.)

Dö, 77] Dömölki Bálint, Farkas Zsuzsa, Sántáné-Tóth Edit: On the Formal Description of Software Objects. Preprints of Second Hungarian Computer Science Conference, Budapest, 1977. pp. 338–361.

Li, 75] Liskov, B., Zilles, S.: Specification Techniques for Data Abstraction. IEEE Trans. on Software Engineering, Vol. 1. 1975. pp. 7–19.

Na, 78] Nakajima, R., Honda, M., Nakahara, H.: Describing and Verifying Programs with Abstract Data Types. Formal Description of Programming Concepts, edited by E.J. Neuhold. North-Holland Publishing Company, 1978. pp. 527–556.

Ro, 77] Dickover, M.E., McGowan, C.L., Ross, D.T.: Software Desing Using SADT. Softech, Inc. August 1977. TP 061.

Sz, 77] Szeredi Péter, Futó Iván: PROLOG kézikönyv. SZÁMOLÓGÉP VII. évf. 3–4. szám. 1977. 7–130 old.

T 17] A rendszertervezés korszerű módszereinek áttekintése. SOFFTECH T17, 1978. március. Készült az SZKI-ban, a KSH–OSZI megrendelésére, az SZKCP CF–31 célfeladat keretében.

Wu, 76] Wulf, W., London, R., Shaw, M.: An Introduction to the Construction and Verification of Alphard Programs. IEEE Trans. on Software Engineering, Vol. 2. 1976. pp. 253-364.

A STRUKTÚRÁLT PROGRAMOZÁS ESZKÖZEI

Bedő Árpád
SZÁMKI

Immár Magyarországon is elmondhatjuk, hogy a struktúrált programozásnak van ho férhető irodalma, és azt is megállapíthatjuk, hogy mi sem tudtuk meghatározni azt, hogy is a struktúrált programozás.

Ezért én is veszem a bátorságot, hogy egy eléggé általános meghatározást adjak: a el túrált programozás olyan programkészítési *módszer,* amelynek követésével valamilyen szer toknak megfelelő szerkezetű programok készülnek.

Ez a meghatározás újabb kérdéseket vet fel: a programoknak van-e szerkezetük, és gyan határozhatjuk meg? Tudunk-e előre megadott szerkezetű programokat készíteni? R sorolhatjuk-e a különféle programszerkezeteket úgy, hogy pl. az egyik jobb (megfelelőbb) másiknál? Mik lehetnek a szerkezet megfelelősége megítélésének a szempontjai?

Ezekre a kérdésekre a számítástudomány még nem adott elfogadható választ; a szakirodalomban csak részeredményeket találhatunk.

Megállapíthatjuk azt, hogy *struktúrált programozási szaktudással programozóink nem rendelkeznek* (mert nem tudnak mit és kitől megtanulni).

Ez a jól ismert eset: mielőtt a szaktudás megvan, a szakember a tudás helyett *célo* követ.

Mi a jó szerkezetű programok írásának a célja?

Használható programok írása. Ez nyilvánvaló volt a struktúrált programozás gondolatának tűnése előtt is. Amit az új irányzat világossá tett az az, hogy a programok használhatóságuk legjobban a *módosíthatóságukkal* függ össze. Egyszerűen fogalmazva: amelyik program nem zen módosítható, azt az élet használhatatlannak minősíti.

A struktúrált programozás fő célja: a körülményekhez szabható, módosítható progr készítése. A struktúrált programozás elvi kérdéseivel foglalkozó kollégáink által elért eredmények így értékelhetjük:

— felhívták a figyelmet ezekre a problémákra
— bizonyos részkérdések megoldását mind elméleti mind kísérleti úton végzik.

A struktúrált programozás számomra a következő szemléletet adja:

1. A program legfontosabb minőségi jellemzője a módosíthatóság (alkalmazhatóságának mértéke.

2. A programozás közösségi tevékenység. (A programot embertársaim számára, az att mazás céljából írom.)

Miért nem programozunk struktúráltan?

A struktúrált programozás alapvető ellentmondása a következő: *a programszerkezet alakításában az emberi és a gépi szempontok egymással éles ellentmondásban vannak.*

A mindennapi tapasztalat azt mutatja, hogy a módosítható, olvasható, alkalmazható programok nem hatékonyak, nehezen illeszkednek a jelenlegi operációs rendszerekhez. Ezt dítva: az „igazán hatékony" programok szerkezete és működése olyan, hogy módosítás *emberi* használatuk általában a lehetetlenséggel egyenértékű.

A struktúrált programozás módszerének önmagában is van gyengesége: ez egy olyan módszer, amelynek *a követéséhez szükséges eszközök* hiányoznak, (túlnyomó többségük még ki sincs találva).

Ilyen körülmények között kik programoznak struktúráltan? A fantaszták. (Vagy rossz nyelvek szerint: akiknek nem igazi feladatokat kell megoldaniuk.)

Milyen eszközök vannak?

Szeretnénk a struktúrált programozás módszerét tömegesen alkalmazni. Ezen cél elérése érdekében a tömegméretekben használt nyelvek (COBOL, PL/I) lehetőségeit

vagy *korlátozzuk:* megállapodásokat, kódolási szabványokat dolgozunk ki és kötelezően alkalmazhatjuk;

vagy *bővítjük:* olyan előfeldolgozó programokat teszünk az igazi fordítóprogramjaink elé, amelyek „kijavítják", illetve kiegészítik az eredeti nyelv gyenge pontjait.

Ezek nyilvánvalóan *kényszermegoldások,* bár hátrányaikat csak az igazi megoldások megjelenésével fogjuk belátni.

Milyen eszközök hiányoznak?

A programozás legfőbb eszköze a programozási nyelv. Megvizsgálandó, hogy a használatban lévő programozási nyelvek megfelelően segítik-e (kényszerítik-e) a programozót arra, hogy megfelelő szerkezetű programot írjon? Ahhoz, hogy ilyen vizsgálatot végezzünk, tudnunk kell előre azt, hogy mit is keresünk, különben nem fogunk ráismerni, ha meg is találjuk!

A programok jó szerkezete kialakításában a legfontosabb eszközünk az *absztrakció.* Kétféle absztrakciót különböztetünk meg: algoritmus- és adatabsztrakciót.

Algoritmus-absztrakciós lehetőség szinte minden nyelvben van: ez a szubrutin vagy az eljárás. Adatabsztrakcióval már ritkán találkozunk. A programozási nyelvek elméletével foglalkozó kutatások ezt a területet még nem térképezték fel eléggé, s ezért a jelenleg széleskörűen használt nyelvek ilyen lehetőséggel nem rendelkeznek. (Adatabsztrakciós lehetőséggel rendelkező nyelv például a CLU és az ALGOL 68.)

Az absztrakción kívül még két fontos eszközt említek. Az egyik a fogalmak kialakításának a lehetősége, s ezen fogalmakkal való programozás (lásd SIMULA 67). A másik terület – sajnos ez még csak nem is eszköz – a programok vezérlési szerkezetének a „programozása". (Itt a soros, kvázi-párhuzamos, párhuzamos futási szerkezetű programokra gondolok.)

Az eddig említett eszközökön kívül (ezek felett) még meg kell valahogy birkóznunk a programrendszerek, illetve a nagyméretű programok szerkezetének kialakításával is.

Ebben a munkában az operációs rendszereket, illetve a moduláris programozást lehetővé tevő nyelvi (fordítási) rendszereket kell eszközként felhasználnunk. Itt az a kérdés, hogy ezek alkalmasak-e a struktúrált programozás segítésére.

A továbbiakban két részletkérdéssel kívánok foglalkozni:

– az algoritmus-absztrakcióval és
– a moduláris programozással.

Algoritmus-absztrakció

Az eljárásdeklarációnak van két olyan esete, amely kizárólagosan a struktúrált progr zási módszer követésekor fordul elő:

– a finomítás és

– az átnevezés.

Finomításnak nevezzük az olyan eljárást, amelyet csak egyszer hívunk meg. Az átn zés pedig olyan eljárás,amelynek törzse csupán egyetlen hívásból áll. Nyilvánvaló, hogy a ram futási minősége szempontjából az átnevezéseknek és finomításoknak eljáráshívásokba ló jelentkezése teljesen felesleges: jelenlétük egyedüli oka a struktúrálás.

A struktúrált programozás alapvető ellentmondásából egyenesen következik, hogy p ramunkba a jó szerkezet kedvéért olyan dolgokat is bele kell írnunk, amelyek jelenléte a tás idején nyilvánvalóan csökkenti a hatékonyságot. Ezért – ha a struktúrált programozá módszerét követni akarjuk – olyan fordítóprogramokra van szükségünk, amelyek a szerk okokból bekerült felesleges eljáráshívásokat elhagyják.

Ezt csak olyan fordítóprogramok képesek megtenni, amelyek alaposan elemzik (és esetleg alakítják) a forrásprogram szerkezetét.

Moduláris programozás

A moduláris programozás mindennapjaiban két fontos folyamat érdemel figyelmet: program modulokra való felbontása, illetve adott modulokból egy program összeállítása. felbontás tervezési kérdés: ebben az előadásban ezt nem kívánom érinteni. Az összerakási azonban már most is használunk eszközöket, az úgynevezett kapcsolóprogramokat (linkage editor, composer). Az összekapcsoláson felül segítik-e ezek a programok a moduláris progr ramozást? Egyetlen segítséget adnak: a névütközéseket kimutatják, azaz figyelmeztetik a programozót, hogy vagy valamit nem definiált, vagy pedig rossz névvel kíván valamely defi niált dolgot használni. Azonban nem ellenőrzik sem a paraméterek számát, sem a paramé sem pedig az eljárások típusát. Ezeken felül még az is baj, hogy a program olyan formá elő a modulok összekapcsolása után, hogy már lehetetlenség az eredeti szerkezetét elemei átalakítani. (Pedig láttuk, hogy erre bizony szükség van.) Ezt úgy összesíthetjük, hogy a ismert összekapcsoló-programok alkalmatlanok arra, hogy a struktúrált programozás módsz segítő eszközök legyenek.

Összefoglalva:

Előadásommal arra kívánom a figyelmet felhívni, hogy a struktúrált programozás mter szerének a követéséhez új fajta szoftver-eszközökre van szükségünk.

Két ilyen eszközt megemlítettem:

– a programszerkezet-átalakító (szerkezeti optimalizáló) programot és

– az úgynevezett magasszintű összekapcsoló-programot, amely a modulok közötti k csolatok ellenőrzését elvégzi, és lehetőséget ad az előzőekben említett szerkezeti optima lás *egész programra* kiterjedő elvégzéséhez.

Intézetünk Programozási Rendszerek Főosztályán az ANSWER programfejlesztést s operációs rendszerben megvalósított CDL2 fordítóprogramban meg van oldva a fenti szer ti optimalizálás, valamint az ANSWER úgynevezett nyelvi rendszere a modulokból magas tű kapcsolóprogrammal készít programot.

Ajánlott irodalom:

H.M. Stahl: Portability and Efficiency in Using an Open-Ended Language, Angewandte In matik 5/79.

A GIN–S SZÁMÍTÓGÉPES FEKVŐBETEG–NYILVÁNTARTÓ RENDSZER–MODELL FELÉPÍTÉSE, KÉSZÜLTSÉGI FOKA ÉS MŰKÖDÉSÉVEL SZERZETT TAPASZTALATOK*

Benedek Szabolcs, Dr. Nagy Ferenc
SZOTE

A Szegedi Orvostudományi Egyetem Számítástechnikai Központjában 1973. óta foglalkozunk a betegellátást és a kutatást segítő feladatok számítógépes megoldásával [11]. Munkacsoportunk programokat dolgozott ki az izotóp rutin diagnosztikai eljárások leleteinek automatikus előállítására [7], a szcintigráfiás [8] és a gastroenterológiai endoscopos leletek készítésére [10], valamint az anamnesztikus és betegvizsgálati adatok gyűjtésére [14, 15, 17].

A kidolgozott egyedi megoldások számos hátránnyal rendelkeztek:

– lassú volt az átfutási idő,
– adatközlési hibák miatt zavarok következtek be,
– a különböző témák nem egységes szemlélet alapján kerültek kidolgozásra,
– a feldolgozott adatok archiválása és későbbi értékelése nehezen megoldható volt.

Az egyedi megoldások előnyének tartjuk, hogy a lehetőségünk volt a különböző szemléletmóddal készült eljárások közül kiválasztani a legmegfelelőbbet, továbbá szert tettünk a kórházi információrendszer megvalósításához szükséges kezdeti ismeretanyagra. A valós igények és eredmények alapján felmerült a számítógépes információrendszer létrehozásának igénye is, amelyet az akkor rendelkezésre álló hardware lehetőségek (CIII–10010) nem engedhettek meg.

Az előzmények alapján 1976-ban a Szegedi Orvostudományi Egyetem Számítástechnikai Központja egy fekvőbetegnyilvántartó rendszer-modell kialakítását kapta feladatul az Országos Műszaki Fejlesztési Bizottság és az Egészségügyi Minisztérium támogatásával munkába állított R–10-es számítógépre.

Ezt a feladatot az OMFB-vel kötött szerződésnek megfelelően 1979. február 28-ig elkészítettük, majd 5 héten keresztül éles környezetben (I. sz. Belgyógyászati Klinika Gastroenterológiai Osztályán(is kipróbáltuk.

A szervezési és programozási munkák megkezdése előtt meg kellett ismernünk a jelentősebb külföldi és belföldi eredményeket [5, 6, 18] valamint azt a hagyományos információrendszert, amelyet a későbbiekben egy számítógépes információrendszerrel szeretnénk helyettesíteni [1, 4].

Felmérések [1] alapján meghatároztuk az ápolási feladatok általános és speciális részeit. Vizsgálatainkat a legrészletesebben a gastroenterológiai osztályon végeztük el, mivel a kitűzött feladat ennek a speciális osztálynak az információrendszerére vonatkozott.

Számbavéve a felhasználók igényeit, a várható hardware és szellemi erőforrásokat, a rendszerrel szemben a következő elvárásaink voltak:

1. A hagyományos információrendszert felváltó számítógépes információrendszer [13] a megszokott ápolási-gyógyítási folyamatot csak a lehető legkisebb mértékben módosítsa. Nem törekedhettünk olyan, – számítástechnikai szempontból egyszerűbb – megoldási módokra,

---

* Az előadás anyaga az Eü.Min. 3-23-0201-03-0/GY számú „Számítástechnikai módszerek, rendszerek, berendezések adaptálása az orvostudományban és az egészségügyben" c. tárcaszintű kutatási főirányhoz minisztériumi szinten kiemelten elfogadott „Számítástechnikai és matematikai módszerek alkalmazása az orvostudományban és az egészségügyben" c. témában végzett kutatómunka alapján készült.

amelyek a klinika megszokott életritmusát és munkamenetét már a kezdeti időszakban is nyegesen módosítanák, ez ugyanis a rendszerünket már a bevezetés fázisában idegenné ten a felhasználók számára. Ezt a döntésünket indokolta továbbá az is, hogy rendszerünket fo zatosan szeretnénk bevezetni a klinikára, és így az ideiglenesen párhuzamosan működő ké ta információrendszernek a csatlakozási pontokon történő szinkronizációja megoldhatatlan lenne.

2. A kiválasztott osztály információrendszerét úgy fedje le, hogy a későbbiek során sik osztályra, majd az egész klinikára is bevezethető legyen.

3. Alkalmazkodni tudjon a klinika várható átszervezéseihez.

4. A kifejlesztésre kerülő információrendszer alapvetően on-line adatfelvételi móddal delkezzen.

5. A kezelők számára a rendszer használata könnyen elsajátítható legyen.

6. A szükséges outputok a megfelelő időpontra elkészíthetők legyenek.

7. A folyamatos működtetés alatt a Számítástechnikai Központ egyéb feladatainak r része elvégezhető legyen.

A folyamatos fejlesztés során figyelembe kellett venni, hogy:

1. a számítógépes programokat és orvosszervezési feladatokat olyan összhangban kell fejleszteni, hogy a kipróbálásuk minél előbb megkezdődhessen s az esetleges tervezési és p ramozási hibák minél előbb feltárhatók és kijavíthatók legyenek.

2. A kifejlesztett alrendszerek és modulok kipróbálása a Számítástechnikai Központ programfejlesztői és kutatási feladatait lényegesen ne zavarja, valamint a kezdeti teszteket tétlenül a rendszerfejlesztők végezzék el.

3. A rendszer elkészülte után huzamosabb időn keresztül rutin alkalmazásra kerüljön eközben fejlesztés folyamatosan történhessen.

Az előzőekben ismertetett fő követelmények figyelembevételével, valamint a felmérés a korábbi eredményeink felhasználásával rendszertervet készítettünk. Ebben rögzítettük (kb 150 oldal és 60 ábra) a hagyományos információrendszert, meghatároztuk a kiépítésre ker teljes számítógépes klinikai információrendszerhez szükséges fejlesztéseket, és a feladatok r oldásának ütemét.

Az on-line adatfelvétel megvalósításához alapvető input perifériának VT–340 típusú display-eket választottunk.

Első és legfontosabb feladatként létre kellett hoznunk egy olyan programrendszert, (alaprendszer) [2, 16], amely a következő feladatoknak és elvárásoknak tesz eleget:

– folyamatosan bővíthető újabb feladatokat ellátó modulokkal és újabb display-ek beállítását is lehetővé teszi,

– az R–10 számítógép központi memóriájának a monitorterületen kívül maradó rés csak fele részben használja,

– az on-line információrendszerektől elvárt – 4 sec-on belüli – válaszidő a későbbie ben rendelkezésre álló újabb erőforrások birtokában (központi memória, diszk, gyors adat viteli vonalak, stb.) elérhető legyen,

– a felhasználók jogosultságának ellenőrzését a rendszer használata során.

A titkosítás hatékonyságához az alaprendszernek biztosítania kell:

– a hozzáférést biztosító titkor kódok cseréjét a kezelő kérésére bármikor,

– a titkos kód védelmét más kezelő elől,

– a titkosítás durva megsértése esetén a megfelelő display automatikus kikapcsolás

– a jogosultak számára a kikapcsolt display-ek újbóli aktiválását.

Az alaprendszernek biztosítania kell még:

– a software vagy hardware hibák miatt történő leállások utáni gyors újraindíthatóságot;

– a későbbi tárgyalásra kerülő feldolgozó programok indítását és a végrehajtásának felü-
letét;
– 16–20 display egyidejű szimultán működtetését,
– a kezelők és a számítógép közötti dialógusok [3] (utasítások) végrehajtását,
– az utasítások végrehajtása alatt adott válaszok színtaktikus és szemantikus ellenőrzését,
– a hibás válaszok esetén a megfelelő hibajelzéseket és a továbbfolytatás biztosítását;
– az utasítások végrehajtásának bármikori felfüggesztését,
– újraszerkesztéssel alkalmazkodni tudjon az esetleges kedvezőbb erőforrás-feltételekhez
módon, hogy egyidejűleg hatékonysága is megnőjön,
– segítse a rendszerfejlesztést, programfejlesztést, és rendszer-tesztelést.

Az alaprendszer létrehozása után az ún. feldolgozó programokat kellett elkészítenünk.
k legfontosabb feladatai a következők:
– leletek előállítása [12]
– adminisztratív és vezérlő listák előállítása.

A feldolgozó programokon kívül szükség van olyan programokra, amelyek elvégzik a kö-
kező feladatokat:
– napi és időszakos adatkarbantartást,
– a rendszer fejlesztésének segítését és a tesztelés megkönnyítését,
– az időszakosan állandó adatok rendszerbe történő bevitelét (központi input),
– a rendszer file-ok allokálását és kezdeti értékellel való feltöltését (rendszer generálás) [9].
– az éppen működő rendszer dokumentációjának automatikus előállítását.

Jelenleg a következő alrendszerek és modulok készültek el:

1. Betegfelvételi és elbocsájtási alrendszer.

a) Input display-ről:
– betegfelvétel legszükségesebb adminisztratív adatai,
– betegfelvétel orvosi adminisztratív adatai,
– betegfelvétel bővebb adminisztratív adatainak felvétele, illetve a korábbiak javítása,
– beteg elbocsájtáshoz szükséges orvosi adatok,
– beteg elbocsájtásának adminisztratív adatai.

b) Output sornyomtatóra:
– adminisztratív listák felvételi iroda számára,
– vezérlő listák RTG, EKG laboratóriumok és kórlapszoba számára, valamint az ügyele-
orvos számára.

c) Output display-re:
– napi és aktuális létszám és ágyfoglaltság professzor, felvételi iroda, osztályvezető or-
számára,
– betegek információrendszerbeli állapotai (adminisztratív adatok felvétele megtörtént,
tusz felvétele megtörtént, stb.) az adatközlők számára.

2. Izotópdiagnosztikai alrendszer.

a) Input display-ről:
– időszakosan állandó standard, mérési és méréstechnikai adatok felvétele,
– pajzsmirigy scintigráphiás leletek adatainak felvétele,
– ambuláns betegek adminisztratív adatainak felvétele.

b) Output sornyomtatóra:
– 11 féle lelet a laboratórium orvosa számára.

3. Ápolási alrendszer

a) Input display-ről:

– beteg ágyra helyezése a munkadiagnózis és a kezelőorvos közlése

– státusz fölvétele

– kezelőorvos és ágycsere közlése.

b) Output sornyomtatóra:

– státusz és rectoscopos leletek előállítása

– öntapadós etikettek előállítása vérminták azonosításához.

A kipróbálás időszakában folyamatosan üzemelő programrendszerünkkel a következő tosabb tapasztalatokat szereztük:

– az eddig elkészült software – az adott hardware feltételek mellett – mintegy 1 ágyas fekvőbetegellátó intézet kórházi információrendszerének kiszolgálására alkalmas. To alrendszerek és modulok bevezetése után is – amivel a szükséges diszk és központi mem terület igénye és a számítógéppel történő párbeszédek száma is megnő – várhatólag 100 kiszolgálására alkalmas marad.

– a későbbiekben várható folyamatos üzemeltetéshez 8–18 óráig terjedő időszakba legalább 7–8 órán keresztül kell biztosítani a gépidőt,

– a napi adatkarbantartásra, a leletek és listák előállítására az esti vagy éjszakai m ban kb. egy óra gépidőt kell biztosítani;

– igen munkaigényes feladatnak bizonyult az „üres rendszer feltöltése", amit a fo tos üzemet megelőző napokban és éjszakákon végeztünk el.

Ez a tapasztalat mutatja, hogy az esetleges hardware hibák komoly gondot okozha ugyanis az egy napnál hosszabb gépkiesés alatt a klinikán „történteket" utólagosan kell mítógéppel közölni. Ezt a munkát az éjszakai órákban csupán a hagyományos kórlapok ján – a kezelőorvos kikérdezése nélkül – sokszor lehetetlen a számítógép által megköve pontossággal és teljességgel elvégezni.

Sűrűn történő gép-kiesések esetén pedig – mivel a rendszer sohasem lesz időazonos felhasználók számára használhatatlanná válik.

A mintegy 170 ember/hónapi munkával elkészített rendszerünket a klinikán dolgozó (nem rendszerfejlesztésben résztvevő) orvosok is alkalmazhatónak tartják és bevezetését k 3 éves időtartamra célszerűnek látnák.

A hosszabb távú üzemeltetés lebonyolítására a számítóközpont és az I. sz. Belgyógy ti Klinika vállalkozik, amennyiben az üzemeltetéshez szükséges erőforrásokat biztosítani t

Az előzőekben mintegy 4 éves időszak munkáját ismertettük. A továbblépés egy le séges formáját a napi rutin munkába történő bevezetésében és a vele párhuzamosan törté újabb alrendszerek és modulok fejlesztésében látjuk.

Irodalom

1] Benedek Sz.: A hagyományos információáramlás elemzésének módszertana és problémái egy számítógépes betegnyilvántartási rendszer létrehozása kapcsán. NJSZT 8. Kollokvium, Kerekasztal, Szeged, 1977.

2] Benedek Sz., Nagy F.: Betegellátást segítő adatkommunikációs (input-output) egy lehetséges megvalósítása R–10 számítógépre. NJSZT 7. Kollokvium, Szeged, 1976. 115–121. old.

3] Benedek Sz., Nagy F., Lehoczky A.: Felhasználható képtípusok a GIN–S utasításainak felépítésével. NJSZT 8. Kollokvium, Szeged, 1977.

4] Benedek Sz., Nagy F., Dr. Győri I., Dr. Csernay L.: A számítógép alkalmazásának lehetőségei a betegellátásban III. A kórházi információrendszer alapelvei, a tervezés főbb szempontjai. 1978. Bp. XVI. évf. 14–16. old. Orvos és Technika

5] Collen M.F.: General Requirements for a Medical Information System (MIS) Comp. and Biomed. Res. 3. 393–406. (1970.)

6] Collen M.M.: Hospital Computer Systems. Wiley and Sons New York, 1974.

7] Csernay L., Benedek Sz.: Pajzsmirigy rádiójód diagnosztikai vizsgálatok automatikus értékelése számítógéppel. Magyar Radiológia 26. 369–376. old. Bp. 1974.

8] Csernay L., Csirik J.: Computer scintigraphia. Orvosi Hetilap, 111. 13–14. old. 1971.

9] Dr. Csernayné Somogyi K., Benedek Sz., Pasek B., Lehoczky A.: A GIN–S generálása R–10 számítógépen

10] Döbrönte Z., Náfrádi J., Varró V., Benedek Sz.: Computer Aided Data Processing for Digestive Endoscopy with the Help of a Coda-System Endoscopy 7. 1–5. 1975.

11] Győri I.: A Szegedi Orvostudományi Egyetem Számítástechnikai Központjának tevékenysége. Egészségügyi Gazdasági Szemle, 12. 112–116. 1974.

12] Lehoczky A., Benedek Sz., Nagy F.: Szöveges leletek előállítása a GIN–S-ben. NJSZT 8. Kollokvium, Szeged, 1977.

13] Dr. Nagy F., Benedek Sz., Dr. Varró V., A számítógép alkalmazásának lehetőségei a betegellátásban. IV., A kórházi információrendszer kiépítésének főbb szempontjai klinikánkon. Orvos és Technika XVI. évf. 2. 1978. 33–37. old.

14] Nagy F., Benedek Sz., Győri I.: A számítógép alkalmazásának lehetőségei a betegellátásban. Kórlapdokumentáció, az anamnesztikus adatok gyűjtése. Orvos és Technika, 6. 185–187. old. 1977.

15] Nagy F., Benedek Sz., Győri I.: A számítógép alkalmazásának lehetőségei a betegellátásban II., Az anamnesztikus adatokat gyűjtő kérdőívvel szerzett tapasztalataink. Orvos és Technika, 1, 9–13. old. Bp. 1978.

16] Pasek B., Benedek Sz.: Egy kórházi információrendszer input tevékenységeit megvalósító programrendszer jellemzői. Programozási Rendszerek, 1978. Szeged, 445. old.

17] Pollák Richárd, Lehoczky András: Betegvizsgálati adatok rögzítése számítógépes feldolgozásra. NJSZT Kollokvium, Szeged, 1976. 107–113. old.

18] Pótzy P., Kovács F., Naszlady A., Széphalmi G.: Kórházi Információrendszerek létesítésének néhány kérdése az Országos Korányi TBC és Pulmonológiai Intézetben folyó fejlesztői munka tükrében NJSZT 5. Kollokvium, Szeged, 1974. 367–377. old.

A NYOMDAIPARI TERMELÉSIRÁNYÍTÁS SZÁMÍTÓGÉPES MEGOLDÁSI LEHETŐSÉGE

Bernát Iván
Kossuth Nyomda

A nyomdaipar egyes nagy vállalatai létszámot és termelési volument tekintve, valam termelés vertikalitásának vonatkozásában kinőtték a hagyományos és átlagos nyomdaipari lati nagyságrendet.

Ez a dinamikus fejlődés természetszerűleg igényli a mind nagyobb munkamegosztás alapuló tevékenységi színvonal növekedését, ami a nagyüzemi termelés előfeltétele.

A nyomdaipari sajátosságok sok vonatkozásban eltérnek az általános ipari gyakorlat ezért nehéz más iparágakban található megoldásokat a nyomdaiparban maradéktalanul érv síteni.

1. A nyomdaipari sajátosságokról

A nyomdaipari sajátosságok két fő területre csoportosíthatók: Először is sajátos a rendelő és a nyomda kapcsolata, mivel a megrendelő az igényeit csak bizonyos elvégzett melő fázis, korrektúra fordulók, imprimatúra leadásakor pontosítja. Ekkor – pl. egy kön gyártása esetén – a gyártás időbeli lefutásának mintegy 80%-a már megtörtént. A nyomd iparban a gyártáselőkészítés fő tevékenysége a dokumentáció elkészítése. Az előkészítési matokban résztvevők tevékenysége nem rendezhető az általános ipari kapcsolat szerint. E gyakorlat hosszú évtizedek alatt alakult ki és a rugalmas menetközbeni változtatásokra túl érzékeny, kisipari adottságokra támaszkodik. Ez a gyakorlat megrögzött, részben szok jogon alapuló, részben adminisztratíve is szabályozott kapcsolattartási rendszerré vált. Ez kialakult gyakorlaton egy vállalati, de még egy iparági szervezés keretében sem lehet vált ni. Ugyanis a megrendelők többségét képviselő könyvkiadók a Kulturális Minisztérium, a galmazók a Belkereskedelmi Minisztérium irányítása alá tartoznak, a nyomdák pedig a K ipari tárcához. Így a gyakorlattól való eltérés a más-más tárcához tartozó kiadók és a for mazók rendszerének racionalizálását is jelentené. Csupán egyetlen nyomda kapcsolattartás rendjének megváltoztatása az üzleti versenyben nem lenne kedvező kihatású a vállalatra. elmondottakból következően a megrendelő-nyomda kapcsolatot, mint meglevő adottságot figyelembe venni, mivel az e téren kívánatos korszerűsítés meghaladja egy szervezés keret lehetőségeit.

A másik nyomdaipari sajátosságokkal rendelkező terület a vállalaton belüli irányítá rület. Ezzel viszont, mint nyilvánvalóan korszerűsíthető területtel kell foglalkozni.

2. A gyártási programról általában

A nyomdaipari vállalatok megnövekedett feladatai szükségessé tették a gyártásszerv rendszerének felülvizsgálatát és korszerűsítését. Ennek elvégzésére egyre sürgetőbbé vált, vel a meglévő rendszer nem adott megfelelő áttekintést a munkák állapotáról és így a dö sekhez szükséges információk nem álltak rendelkezésre a kellő időben.

Jelenleg a gyártási program a legtöbb esetben nem más mint egy elvárási lista, mely egy-egy üzem vagy munkahely részére közli, hogy mit, mikorra készítsen el. Nemigen tesznek mégcsak utalást sem arra, hogy az egyes műveleteket mikor kell elkezdeni, illetve befejezni ahhoz, hogy a következő munkahely is menetrendszerű kezdési és befejezési határidőt tudjon tartani. A határidők rendszerint nem differenciáltak, hanem egybeesnek többségükben a programozott időszak záró, forduló napjával.

Az operatív gyártási programok felépítése és teljesítésükről történő visszajelentések a legtöbb esetben olyan szerkezetűek, melyek a termelés irányítását végzők részére legfeljebb nagyvonalú tájékoztatást adnak és főleg a gépkihasználási, statisztikai, az üzemgazdasági elemzések részére szolgáltatnak értékelhető adatokat.

A gyártási programkészítést a felügyeleti és irányítási szervek nem kívánják meg, nem írják elő. A gyártási program és megvalósítása viszont a vállalatok jól felfogott érdeke.

A gyártásprogramozás nem csupán egy tevékenység a vállalati működés más tevékenységei között, hanem a vállalati mechanizmus jó működésének nagyon fontos része. A jó gyártási program biztonságos termelésirányítás megvalósításán túl további jelentős vállalati gondok elhárítását, illetve megoldását is segíti:

– *A munkaerő-biztosítás vonatkozásában* a termelés területén segíti a termelékenység növekedését, és ezáltal lehetővé teszi, hogy a termelő létszámmal nagyobb eredményt lehessen elérni. Csökkenti az adminisztratív munkaigényt, amennyiben a határidő módosítási és reklamációs ügyintézés lényegesen csökken.

– *Az anyagbiztosítás vonatkozásában* segíti a vállalati anyaggazdálkodás hatékonyságát azáltal, hogy megfelelő beszerzési ütemezést tud meghatározni, és megakadályozza az anyagkészletek fölösleges növekedését.

– *Pénzügyi vonatkozásban* a következő kihatásokat vehetjük figyelembe:
  - a gyártási átfutási idő csökkenése csökkenti az eszközök lekötöttségének idejét, növeli a forgó eszközök forgási sebességét;
  - csökkenthető az anyagkészlet-állomány;
  - csökkennek a késedelmes szállítások miatt felmerülő többlet-költségek;
  - csökkenthetők a gyártásközi fölöslegesen megmozgatott anyagok és a félkésztárolás miatt felmerülő többlet-költségek;
  - megbízhatóbbá teszi a vállalat pénzügyi tervezését.

– *A munkafegyelem vonatkozásában* nagyban elősegíti a munkafegyelem megszilárdítását. Talán nem szorul bővebb magyarázatra, hogy nehéz a munkafegyelmet olyan helyen megkövetelni, ahol az egyes időszakokban fokozottan sürgetik a munkát, majd a hajrát követően akadozik a munkaellátottság. Természetes követelmény, hogy ahol az irányítás nem tudja biztosítani az ütemességet és a megfelelő feltételeket, arra a termelő munkahely a saját területén a munkafegyelem lazításával reagál. Ez bekövetkezhet, mivel az irányításnak a saját gyengesége miatt nincs erkölcsi alapja a számonkérésre.

A nyomdaipari sajátosságokból adódik, hogy nagyszámú, egyedi megoldást igénylő, viszonylag bonyolult megrendelés teljesítését kell szolgáltatnia. Ezért, a programkészítés bonyolultsága itt sokkal nagyobb, mint azoknál az ipari vállalatoknál, ahol kisebb számú megrendelésen belül termékenként is kevesebb feltételnek kell eleget tenni.

A programkészítés szempontjából egyetlen megrendelés csupán egyetlen tételt jelent. Nem hagyhatjuk figyelmen kívül ugyanis, hogy minden egyes megrendelés minimálisan legalább öt technológiai fázison megy keresztül – ami meglehetősen ritka dolog könyv főprofil esetén – és az egyes műveletek kezdési és elkészülési időpont meghatározásánál ez már tíz határidő megállapítást jelent. Ezekhez még figyelembe kell vennünk az anyagbiztosítási, a korrektúra forduló, az anyagmozgatás és az egyéb program-megbízhatóságot befolyásoló határidőtényezőket. Tehát csupán egy mégrendeléssel kapcsolatban a programozónak minimá-

lisan 8–10 szempontot, határidőt kell figyelembe venni. Tovább bonyolítja a nehézségeket, hogy az egyes technológiai helyeken az ott készülő, megrendelésekre olyan sorrendet kell állapítani, mely a következő technológiai helyekre megállapított sorrendekkel is összhangban van.

A gyakorlatban a programozó rutinból minimálisra redukálja a lehetséges kombinációk számát, és ennek eredményeként már eleve csak redukált lehetőségeket veszi figyelembe. Természetesen minél nagyobb mérvű a rendelés, annál több lehetőség van váratlan akadályok a gyártás irányításának során szükségszerűen felmerülő problémák jelentkezésére.

3. A termelés számítógépes irányításának szükségessége

A gyártásprogramozás számítógép segítségével történő megvalósításakor reálisan számolni kell a következőket:

– A számítógépek egy kötött árakkal dolgozó ipar számára drágák, és csak gazdaságos üzemeltetésük mellett térülnek meg, illetve biztosítják a kívánt gazdasági eredményt.

– A számítógép alkalmazás miatt felmerülő többlet-költséget meg kell termelni, és nem lehet azt a vevőre áthárítani.

– A gyártásprogramozás számítógép alkalmazásával történő megoldása csak egy építő eleme lehet egy integrált vállalati adatfeldolgozási rendszernek. Ugyanis a termelési programkészítés olyan adatok feldolgozása, amelyek eredetileg a vállalat más szervezeti egységeiben képződnek, más vállalati részfeladatok megoldásának eredményeként. A gyártás programozása csakis akkor lehet célszerű és gazdaságos, ha jól szervezett csatornákon jövő szekunder adatokkal valósítják meg.

Ezeknek a megvalósítása érdekében a könnyűipari ágazat számítástechnikai kutatás-fejlesztési célprogram (KIM–13) szellemében a könnyűipari V. ötéves számítástechnikai alkalmazás-fejlesztési tervének előírása szerint a nyomdaipari szakágazatra vonatkozóan az a döntés született, hogy ki kell dolgozni

– a szakágazati számítástechnikai alkalmazásának Automatikus Irányítási Rendszerét
– a vállalati termelésirányítási rendszerhez kapcsolóló alrendszereket, modulokat az átláthatóság messzemenő figyelembevételével.

Így a nyomdaipari szakágazati Automatikus Irányítási Rendszer az alábbi alrendszereket tartalmazza:

– a termelés műszaki előkészítése;
– a termelés irányítása;
– az anyaggazdálkodás irányítása;
– a munkaerőgazdálkodás irányítása;
– az állóeszköz gazdálkodás irányítása;
– a segédtevékenység irányítása;
– az értékesítési;
– az önköltség és jövedelmezőség;
– a pénzügyi alrendszerek;
– az adatbank alrendszer.

4. A fő célkitűzések

A rendszer elé támasztott követelmény az, hogy mind leterhelésben, mind gazdaságilag megközelítsen egy optimális programot, illetve segítse ennek megvalósítását.

Abból az információ igényből indulunk ki, mely a nyomdaipari sajátosságok mellett a ogramcsoporttól elvárható:

– A gyártó üzemek részére egyenletes leterhelést biztosító gyártási, továbbá a kötészet, pedíció (szállítás), pénzügy részére havi kiszállítási (esetleg dekád) programok legyenek ké-thetők.

– A megrendelések visszaigazolásához szállítási határidők legyenek megállapíthatók.

– A vállalati kereskedelmi részlegét kellő időben lehessen tájékoztatni, hogy milyen emrészekben, mely programhelyeken, milyen időpontban milyen nagy szabad kapacitás mu-kozik, amire munkát illetőleg kooperációs szolgáltatást lehet vállalni alulterheltség, illetve dni túlterheltség esetén.

– Az anyaggazdálkodás pontosan tájékoztatható legyen, hogy mely munkához, milyen pontig kell (a kiemelt) anyagokat biztosítania.

– Egy-egy munka gyártási, átfutási ideje csökkenthető legyen.

– A termelésirányítás folyamatosan tájékozódhassék a terhelésről, hogy idejében megte-sse az operatív intézkedéseket.

– A gyártáselőkészítés – melynek a nyomdaiparban más feladata van mint általában a obi iparágban – folyamatosan tájékozódhassék, hogy a konvertálható technológiai megoldá-k közül, mikor, melyik alternatíva segíti elő a termelést.

– A megrendelőknek szükség esetén felvilágosítást lehessen adni az egyes munkák állásá-, állapotáról.

– A TMK nagyjavítások ütemezhetők, illetve figyelembe vehetők legyenek a termelés mpontjából.

– A tervezett és tényleges ráfordítási idők ösztönzően hassanak a munkanormák karban-tására, és automatikusan elvégezhetők végezhetők legyenek.

– A túlóra igények előrejelezhetőek legyenek.

– Hatékonyabb legyen a külső kooperáció igénybevétele és annak programozhatósága.

5. A számítógépes információrendszer működési köre

Egy teljes elektronikus adatfeldolgozási rendszerterv a Kossuth Nyomda teljes termelési, vezési és gazdasági rendszereinek átfogását tervezi. Ennek egyik alrendszere a termelésirá-ítás alrendszer. A jelen szervezés csak a termelésirányítás területére, illetve kibővítése a ter-elés által szolgáltatott a termelésirányítás szervezéséhez szükséges adatok feldolgozására ter-ki. Ugyanakkor a termelésirányítás alrendszer információs köre kiterjed a vállalat valamennyi nkcionális területére.

Az alrendszer bevezetése után adatokat kap a termelés programozásához.

– a gyártáselőkészítéstől (Terhelési lap, TRADAT, munkánként),

– az anyag és áruforgalmi osztálytól (anyagbeszerzési jegyzék, AGBEER, beérkezési té-lenként);

– az üzemgazdasági osztálytól (személyi és gépkapacitási adatok, KAPACT, programozott üveletenként);

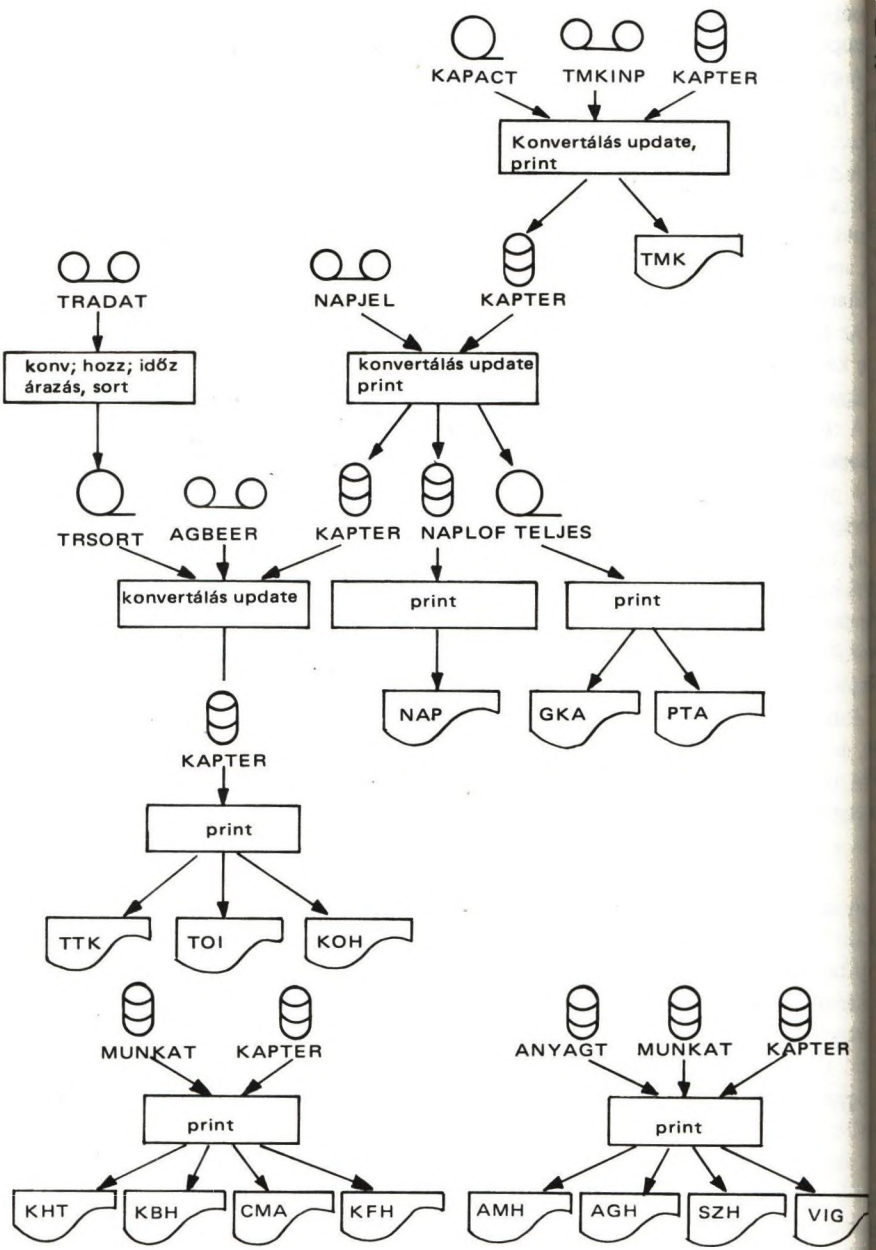
– a munkaügyi osztálytól (programozási normaidőadatok, NORMAT, programozott mű-letenként);

– az üzemektől (napijelentések, NAPJEL, a termelés napi alakulásáról, programozott üveletenként);

– a gazdasági igazgatótól (műveletenkénti átlagos árbevételek, MUVELT);

– a termelési igazgatótól (hozzálék mennyiségek, HOZZAT, műveletenként);

– az üzemfenntartás vezetőjétől (termelőberendezések nagyjavítási terve, TMKINP, prog-mozott gépcsoportonként);



1. ábra A nyomdaipari termelésirányítás általános folyamatábrája könyv főprofil ese

6. Kapacitás leterhelés vizsgálat

A rendelkezésre álló kapacitás felméréséhez néhány fontos dolgot kell tudni. N
tesen:

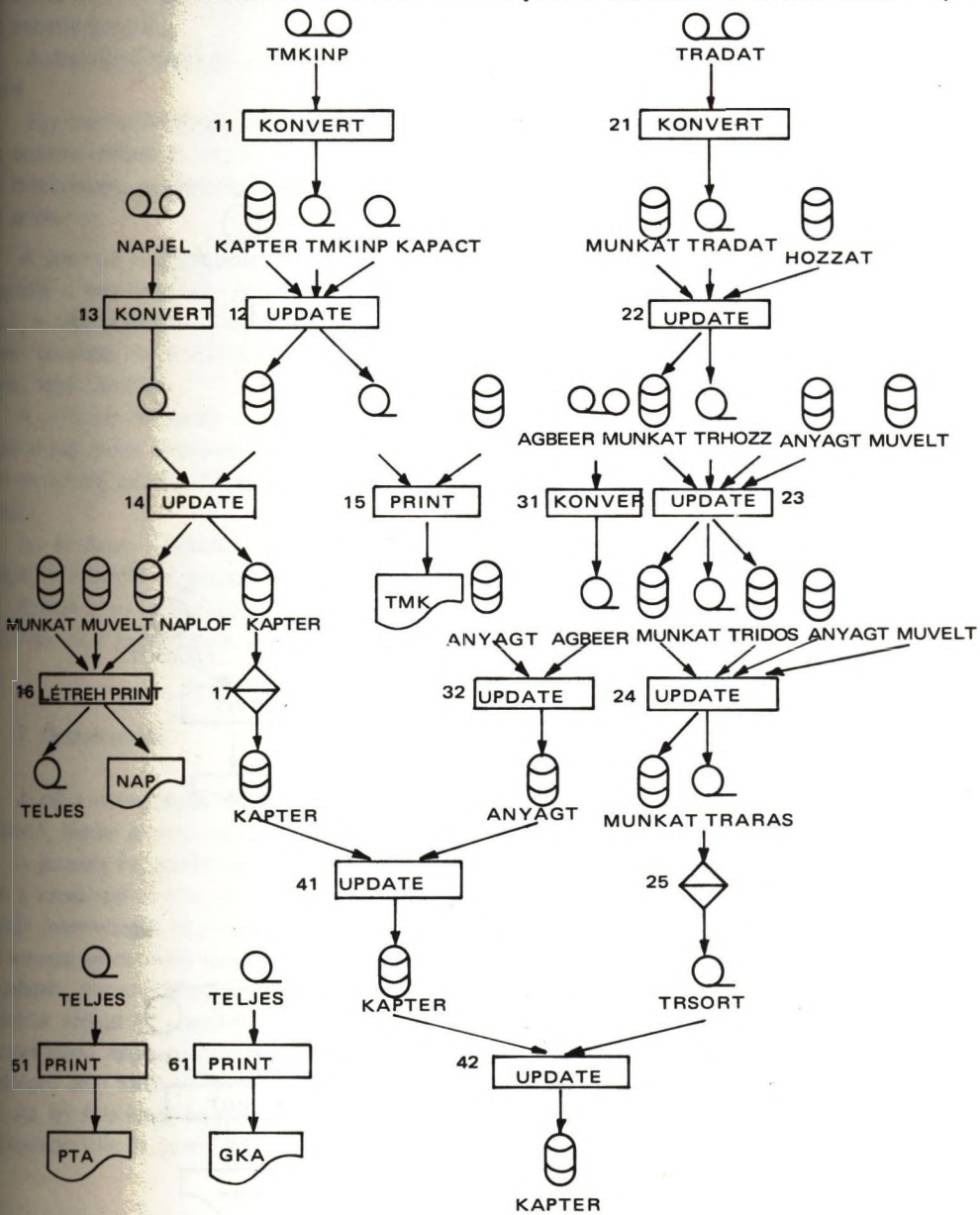
- van-e megfelelő gépi kapacitás, amelyen a feladatot el lehet végezni,
- rendelkezésre áll-e megfelelő anyag, amelyet a megrendelő kívánsága szerint,

használásnak megfelelő mennyiségben, minőségben és a feldolgozásnak megfelelő időben hozat be a vállalat,

– tudni kell, hogy mely munkák készültek már el, illetve vannak gyártás alatt, vagy hogy ezek a munkák milyen mértékben készültek el, ti. hol tartanak.

Az egész kapacitás leterhelés vizsgálathoz – mely egyébként a teljes munkát és problémakört magában foglalja – generálhatunk egy ún. KAPTER – kapacitás terhelési – file-t. Ez a file, mint egy programhelyenkénti naptár hordozza a legfontosabb adatokat, valamint azokat, amelyek a többi, alfile-hoz a feldolgozás érdekében kapcsolódnak.

E KAPTER file-időadataiból le kell vonni a javítási műveletekre fordítandó időket, a

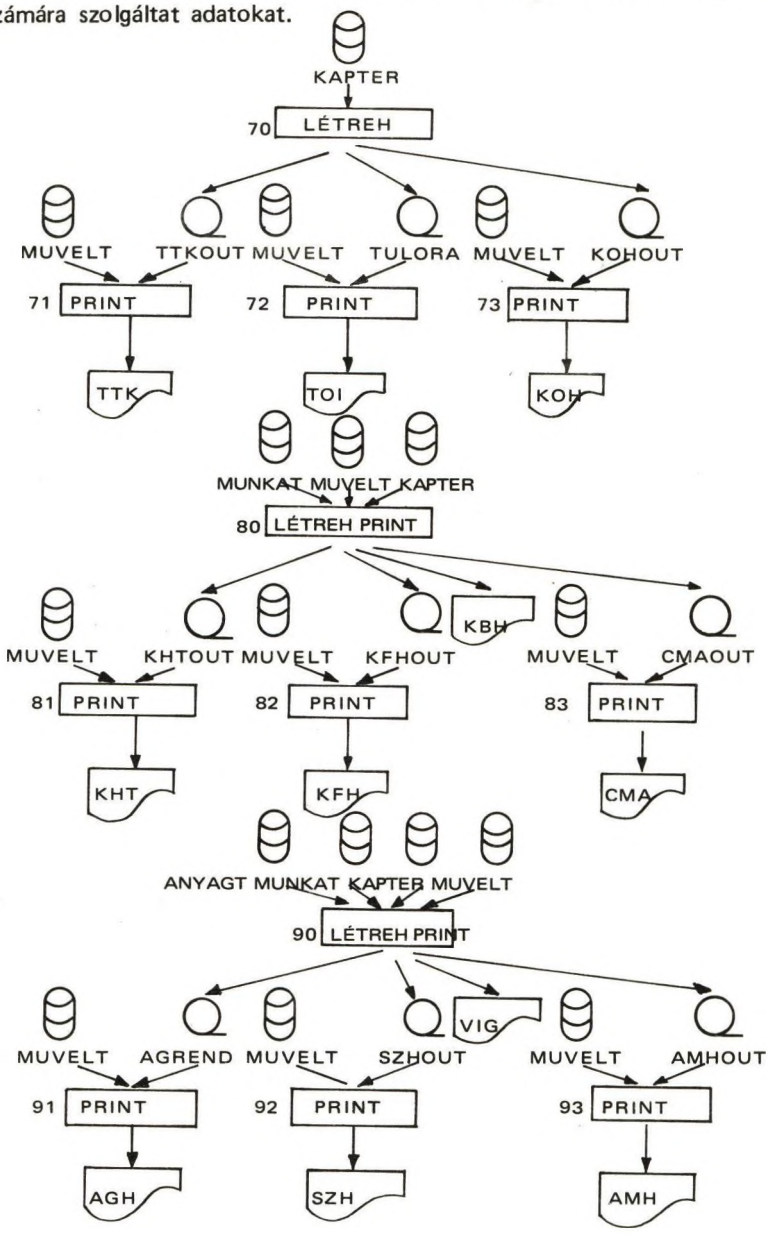


2. ábra A feldolgozás folyamata: az aktuális kapacitásterhelés file kialakítása

napijelentések alapján az elkészült munkák idejeit, valamint meg kell itt jelölni, hogy me munkák, mely műveleteihez áll rendelkezésre az előírt anyag mind minőségben, mint me nyiségben.

Az új, még be nem programozott munkákról az ún. tranzakciós adatokat a megren lőktől kapott, (a Kiadói Főigazgatóság által egységesített megrendelő lapról nyert) adatok alapján a gyártáselőkészítő egy terhelési lapot állít ki.

Ez a terhelési lap tartalmazza a munka alapadatait, valamint az egyes programkeresz metszetekre jutó mennyiségi adatokat, az ott felhasználandó anyagok leírásával. Ez az alap zonylat a feldolgozás első lépésében csak az ún. MUNKAT – a munkák regisztrálására szo ló – file számára szolgáltat adatokat.



3. ábra A feldolgozás folyamata: az aktuális kapter file-ból az outputok kialakítása

Iparágunkban egy országos normatáblázat szerint van egy ún. hozzálék rendszer, egy technológiához tartozó rontási százalék, hivatalos selejt, amely nagysága végül is az egyes dolgozótól függ. (Hogy mennyire tud, vagy tudnak ezen belül maradni, ez a nyomda számára hozhat és vihet is bizonyos összegeket.) Ezzel a hozzálékkal azonban mind az anyagok, mind a programhelyek technológiai mennyiségi adatait bővíteni kell.

Csak ezután a bővített anyagtöbblet után állnak az ún. ANYAGT — anyag törzs — rendelkezésére a feldolgozható anyagok.

A különféle programhelyek más és más természetes mértékegységben dolgoznak, amiket közös nevezőre csak az idővel lehet hozni. Megbízható normaidők alapján ez nem jelenthet komoly gondot.

A szóbajövő munkák programozása ezután a következő prioritás rendszer alapján történne.

Egy munka fontosságának egyik szempontja a várható árbevétel. Minden programhelyhez hozzárendelünk az előző időszakok adataiból nyert átlagok alapján egy várható árbevételi mennyiséget, egységenként. Ezért a tranzakciós adatainkat kibővítjük a munkák árbevételi értékeivel.

A prioritás érdekében a feldolgozás menetébe bele kell vinnünk a szubjektumunkat is. Végpedig el kell határozni minden egyes munkáról előre, hogy mennyire fontos az a nyomdának, a népgazdaságnak, milyen fontos politikailag. Ezt egy számsorral lehet modellezni. A jobban szeretem munkákat kisebb, a kevésbé szeretem munkákat nagyobb számmal jellemezzük, vagy fordítva.

A prioritás harmadik összetevője az anyag. Teljesen természetes dolog, hogy a feldolgozandó anyag minél kevesebb időt töltsön az üzem, a vállalat anyagi — pénzügyi — és topográfiai területén, mivel a forgó készletekbe beleölt pénzösszegeket minél hamarabb fel kell szabadítani.

Így áll össze a prioritás egy háromjegyű számmá, amely első számjegye a munkák fontosságát, a második az értékét és a harmadik az alapanyagok elérhetőségét jelöli.

Ezután a sorbarendezésnek már semmilyen akadálya sincs. A feldolgozó program, a programhelyek napi kapacitásait a sorbarendezett munkákkal folyamatosan feltölti.

7. Összefoglalás

A cél a nyomdaipari sajátosságoknak megfelelő viszonylag egyszerű és a vállalati AIR-rá bővíthető, illetve annak alapjául szolgáló kapacitás terhelési rendszert felállítása volt.

A javasolt kapacitás terhelő rendszer egy mérleghez hasonlítható. Ennek egyik serpenyőjében a kapacitást tettük, levonva belőle a nagy- és a tervezett javítási időket, az elvégzett munkák mennyiségét és a beérkezett anyagokat mint egy jelölést, hogy mely munkákat milyen keresztmetszeteken kezdhetjük el. A mérleg másik serpenyőjébe, pedig a beérkező munkák adatait, illetve a programhelyenként leterhelhető mennyiségeket helyeztük el úgy, hogy bővítettük azokat az engedélyezett hozzálékokkal, konvertáltuk a normaidejük alapján a természetes alapú mennyiségeket idő mennyiségekre, majd sorbarendezzük az általunk illetve a számítógép által közösen generált prioritás indexek alapján.

Az így felállított kapacitás terhelési mérleggel illetve az ebből adódó táblázatokkal a különböző funkcionális és üzemi részlegek vizsgálhatók, irányíthatók.

A TPA/I KISSZÁMÍTÓGÉP SOFTWARE ESZKÖZEI KARDIOLÓGIAI DIAGNOSZTIKAI CÉLÚ FEJLESZTÉSÉNEK ÁTTEKINTÉSE

Bolyky János–Szabó András–Simonkay Sándor
KFKI, KFKI, BME

Bevezetés

A számítógéppel segített diagnózis megvalósításának sikeressége egy adott diagnosztikai területen nagymértékben függ a felhasználásra kerülő hardware, software eszközök használhatóságától.

A KFKI–OTKI együttműködés keretében létrehozott kisszámítógépes interaktív EKG diagnosztikai rendszer (1) fejlesztése során a TPA/i kisszámítógép standard alapsoftware eszközeinek orvosbiológiai mérésadatgyűjtési, adatkezelési, adatfeldolgozási irányú bővítésére is sor került.

Mivel az így létrehozott alapsoftware bázis (OS)i általános célú, RTS/i real-time operációs rendszerek, REAL–Time BASIC programozási nyelv) szolgáltatásai unikálisnak tekinthetők az orvosbiológiai számítógép alkalmazások feladatkörében, az előadásban a rendszerszolgáltatások felhasználói oldalról történő bemutatására kerül sor.

Az alapsoftware-bázis részletezése előtt röviden áttekintjük a kisszámítógépes interaktív EKG-diagnosztikai rendszer működését.

1. A rendszer működése

Felvételezés: a hitelesített, páciens azonosító kóddal ellátott EKG regisztrátumok a Frank elvezetési rendszer előírásainak megfelelően kerülnek felvételezésre. A felvételezés automatizált, mind a páciens azonosító kód, mind a kalibrációs és az EKG jelek egy négy csatornás analóg/FM magnetofonon kerülnek tárolásra (off-line működés).

Adatelőkészítés: az adatelőkészítési eljárás célja, hogy a lényegkiemeléshez felhasználható EKG regisztrátumot előállítsa. Ennek során a rendszer egy CAMAC interface-on keresztül fogadja a páciens azonosító kódot, és 500 Hz-es frekvenciával mintavételezi az EKG jeleket. Ezután ellenőrzi, hogy az adott EKG rekordban megtalálhatók-e a páciens azonosító kód, a kalibrációs, és Frank rendszernek megfelelően rögzített X, Y, Z EKG jelek elemei. Majd annak ellenőrzésére kerül sor, hogy az EKG jeleket tartalmazó rekord szakaszon a jel-zaj viszony nem kisebb-e egy empirikus küszöbértéknél. A továbbiakban a rendszer ellenőrzi, hogy elsősorban impulzus zaj jellegű műtermektől mentes, felhasználható EKG regisztrátum szakasz van-e az adott rekordb

Mindezek után a rendszer a mintavételezett EKG jeleket egy digitális, rekurzív software szűrő segítségével szűri. A szűrőkarakterisztika 50 Hz-nél levő nulla helye a hálózati brumm k szűrését biztosítja, egyébként a karakterisztika nem torzítja el az EKG jel spektrális tartalmát. A zajszűrés után csatornánkénti kalibráció következik, az 1 mV-nak megfelelő kalibrációs jel litudóra normálva.

Az alapvonalak visszaállítása érdekében a rendszer csatornánként, az egyes szívrevolúci $P_{end} - Q_{onset}$ szakaszaira kiszámítja a térbeli sebesség (SV) vektor szakaszokat. A térbeli sebesség vektor szakaszok minimum helyeire vonatkoztatva, csatornánként elvégzi az alapvonalak visszaállítását.

Lényegkiemelés: Az eljárás első lépéseként a tipikus szívciklus kiválasztására kerül sor, amely 10 egymást követő szívciklus magnitudó (M) vektorainak QRS maximumaira fittelt átlagától való eltérés minimumának megkeresése alapján történik. Az ezt követő karakterisztikus pont kiemelés során az EKG görbét a rendszer a jelek morfológiailag meghatározható, karakterisztikus pontjait jellemző idő és amplitudó adatokkal írja le. Az automatikus karakterisztikus pont felismerő algoritmus a térbeli nagyság és sebesség vektorok, valamint az X, Y, Z elvezetések vizsgálatával P_{onset}, P_{peak}, Q_{onset}, Q_{peak}, Q_{end}, QRS_{max}, S_{end}, ST, T_{peak}, T_{end} karakterisztikus pontokat keresi meg. Ezután a rendszer az EKG jel idő és amplitudó paramétereit egy rendezett minta-készletben tárolja, az így összeállított, komprimált minta-készlet közvetlenül felhasználható diagnózis-számításhoz.

Diagnózis-számítás: A diagnózis-számítás első részében a rendszer az ORS szakaszt analizálja, referencia adatbázis felhasználásával, az etalonhoz való hasonlítás alapján, a legközelebbi szomszéd keresésének módszerét felhasználva.

Az így megtalált ORS diagnózist az algoritmus több eljárással valószínűsíti. Ha pl. az ORS diagnózis infarktusta utal, akkor a ORS szakasz első 4 – nagy információ értékű – momentán vektorának hasonló vizsgálatával dönt a rendszer az infarktus lokalizációjáról, de figyelembe veszi a tradicionális diagnózis-alkotás döntési szabályainak egyes elemeit is.

A ORS-re vonatkozó diagnózistól függően az ST, T hullámok analízisét a tradicionális orvosi döntési logikán alapuló fa struktúrájú eljárással végzi el az algoritmus, amire objektív lehetőséget az ST–T szakasz ORS-nél lényegesen egyszerűbb morfológiája ad. Végül aritmia analízis következik, az EKG rekord értékelhető revolúciói alapján.

A rendszer által választható diagnózis string-készlet a ORS szakaszra 41, az ST szakaszra 25, a T hullámra 35, az aritmiára vonatkozóan 28 állítást tartalmaz.

Eredményközlés: a megengedett diagnózis-kombinációk kiválasztása után a rendszer könnyen interpretálható formában közli az eredményeket.

2. Az alapsoftware eszközök működése

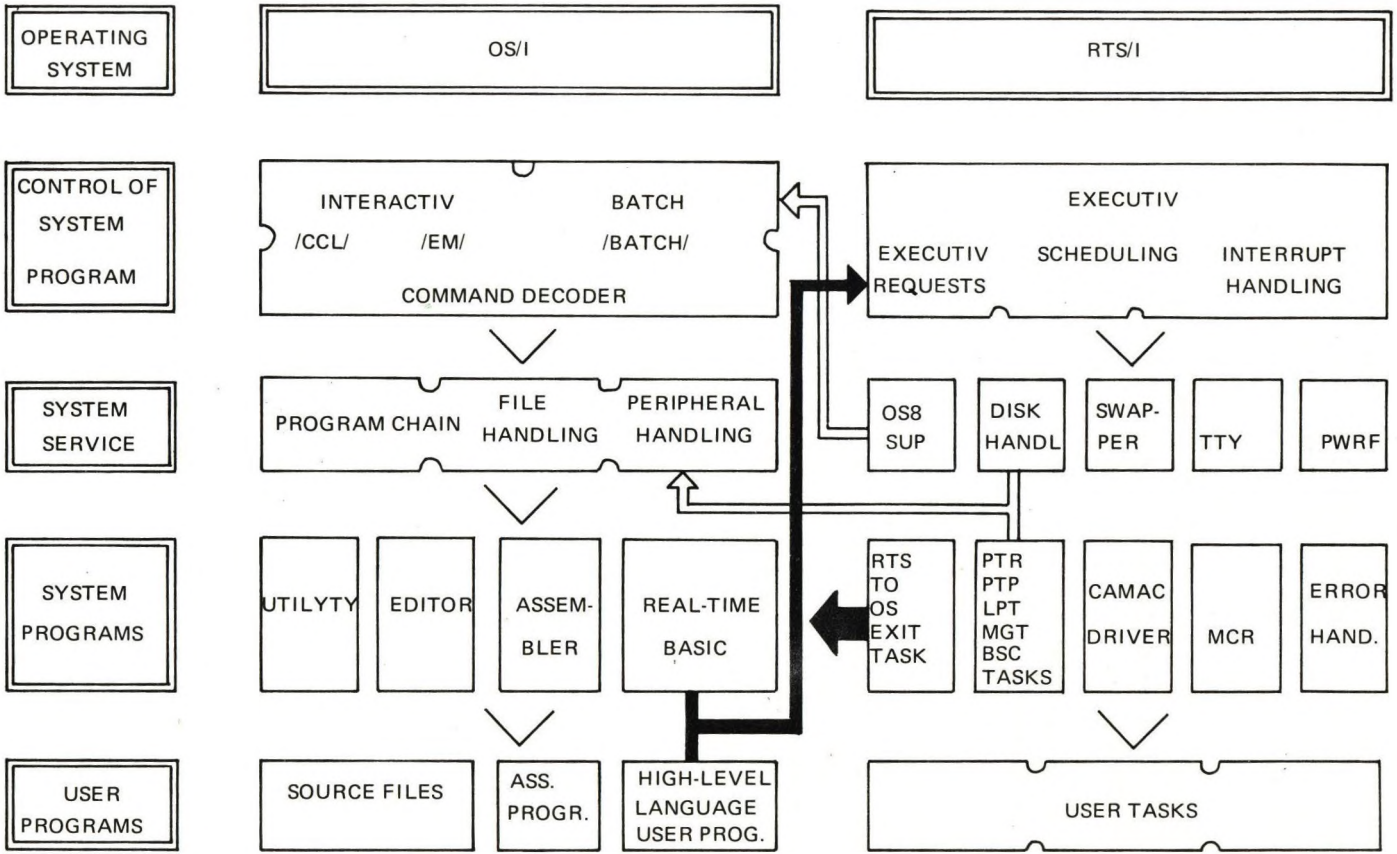
2.1. *Mérésadatgyűjtés*

A real-time mérésadatgyűjtés az RTS/i–OS)i operációs rendszer (1. ábra) együttes felügyelete alatt történik. A mérés vezérlésének eszköze a rendszer real-time magasszintű nyelve, az RT–BASIC. Az RT–BASIC sajátossága, hogy mind az RTS/i realtime rendszer, mind az OS/i általános célú operációs rendszer szolgáltatásait képes kihasználni.

Az RTS/i real-time rendszer révén lehetőség nyílik real-time mérésadatgyűjtési feladatok elvégzésére. A real-time jellegű CAMAC kezelés, amely az RTS/i eszközeivel valósul meg, teljes mélységig használható az RT–BASIC magasszintű nyelvből.

Egy adott feladat megvalósítása során kikristályosodó feladatrészek, amelyek megoldásában változás nem várható, továbbá amelyek elvégzése jelentősen befolyásolná a program futásának össz-időtartamát, átültethetők magasszintű nyelvű programokból assembler nyelvre.

Az assembler programok az RTS/i real-time rendszer taskjaiként kerülnek implementálására. Ez lehetővé teszi a feladatmegoldás real-time jellegének továbbfejlesztését, párhuzamos mérésadatgyűjtés megvalósítását és a számítógép sebességének teljes kihasználását. Ezen RTS/i tasakok azonban továbbra is az RT–BASIC felügyelete alatt, a BASIC és az RTS/i összekapcsolása révén, mintegy BASIC alprogramokként működnek.


OPERATING SYSTEM
OS/I
RTS/I
CONTROL OF SYSTEM PROGRAM
INTERACTIV
/CCL/
/EM/
BATCH
/BATCH/
COMMAND DECODER
EXECUTIV
EXECUTIV REQUESTS
SCHEDULING
INTERRUPT HANDLING
SYSTEM SERVICE
PROGRAM CHAIN
FILE HANDLING
PERIPHERAL HANDLING
OS8 SUP
DISK HANDL
SWAP-PER
TTY
PWRF
SYSTEM PROGRAMS
UTILYTY
EDITOR
ASSEM-BLER
REAL-TIME BASIC
RTS TO OS EXIT TASK
PTR PTP LPT MGT BSC TASKS
CAMAC DRIVER
MCR
ERROR HAND.
USER PROGRAMS
SOURCE FILES
ASS. PROGR.
HIGH-LEVEL LANGUAGE USER PROG.
USER TASKS

2.2. *Adatfeldolgozás*

Az adatfeldolgozás, ellentétben a real-time jellegű mérésadatgyűjtéssel, nem igényli a CAMAC kezelését és nem tartalmaz real-time operációs rendszert feltételező párhuzamos műveleteket. Számításigényessége viszont feltételezi az OS/i operációs rendszer szolgáltatásait. Ennek megfelelően feldolgozás alatt csak az OS/i működik, mint támogató operációs rendszer.

A feldolgozás a mért adatok alapján teljes mértékben az RT–BASIC magasszintű nyelv segítségével történik. A magasszintű nyelv lehetővé teszi a program egyszerű, kényelmes és gyors változtatását, megengedi, hogy az algoritmus optimális változatát kísérletezhesse ki a felhasználó. Ugyanakkor az RT–BASIC nyelv könnyen elsajátítható, így a feladat megoldását nem feltétlenül szükséges számítástechnikai szakemberre bízni.

2.3. *Az OS/i operációs rendszer által nyújtott támogatás*

Az OS/i operációs rendszer háttértár szervezési és adatkezelési rutinjain kívül még néhány olyan alap-software támogatást is nyújt, amelyek célja a programozás kapcsán felmerülő számítógépes problémák kiküszöbölése. Ezek közé tartozik például a rendszer memória-kezelése is. A rendszerbe épített programláncolási opció segítségével lehetőség van magasszintű nyelven olyan programok készítésére, amelyek helyigénye meghaladja a rendelkezésre álló memória kapacitását.

Az operációs rendszer másik fontos szolgáltatása, a perifériakezelés segítségével lehetséges az összes, a számítógéphez csatlakoztatott periféria kezelése a magasszintű nyelvből, továbbá a periféria konfiguráció bővítése a felhasználói programja módosítása nélkül. Végül mód nyílik speciális perifériák, így például grafikus displayek kezelésére is, ami az orvosbiológiai célú feldolgozások közben megvalósítandó interaktivitást teszi elérhetővé.

3. A software eszközök használata

Az operációs rendszerek működtetése, összekapcsolása, a BASIC nyelvű felhasználói programok futtatása az OS/i operációs rendszer BATCH monitorja segítségével történik. Ez a felügyelő program lehetővé teszi az operációs rendszerek, rendszerprogramok és a felhasználói programok programozott működtetését. Ennek révén összetett feladatmegoldások is megvalósíthatók, operátori vagy felhasználói vezérlés mellett, vagy felügyelet nélkül. A BATCH processzor vezérlő információinak előállítása után a rendszer alkalmas az önálló működtetésre, így az RTS/i, az OS/i, vagy a két rendszer összekapcsolt együttesének működtetésére is.

Végül összefoglalva azokat a szolgáltatásokat, melyek a bemutatott alap-software eszközöket jellemzik:

- Kapcsolat az általános célú és a real-time operációs rendszer között
- Real-time periféria kezelés (CAMAC)
- Távadatközlés
- Nagyméretű, assembler nyelvű felhasználói taskok implementálhatósága szabványos RTS-i task-ként
- Virtuális file hozzáférés
- Grafikus input-output funkciók
- BATCH feldolgozás.

4. Irodalom

1. Bolyky J.—KozmannGy.—Szlávik F.—Wolf T.—Antalóczy Z.—Regős L.—Bukosza I.—Bíró S.: A KFKI—OTKI EKG diagnosztikai rendszer fejlesztés tapasztalatainak áttekintése. 9. Neumann Koll., Szeged, 1978.
2. Bolyky J.—Szabó A.: Design principles of a new computer aided ECG analysis program. 14 th DECUS Europe Symp., Copenhagen, 1978.
3. Szabó A.: Some enhancements of RTS/i. 13t DECUS Europe Symp., London, 1977.

A VEIKI R–40 SZÁMÍTÓGÉPÉHEZ CSATLAKOZÓ R–10 ALAPÚ FRONT–END

Böszörményi László
VEIKI

A VT–55000 software rendszere

A VT–55000 software rendszere az EMSV szupervizor programból, valamint a vezérlő illesztő logikát realizáló emulátor programból, illetve programokból áll.
rendszer jelenleg is fejlesztés alatt áll, az első változata 1976 szeptembere óta üzemel.
Az EMSV szupervizor jelenlegi, első változata (EMSV–I) látja el a perifériális berendesek (CCA, CLA, real-time óra, helyi perifériák) fizikai szintű kezelését, valamint multitask örnyezetet biztosít az emulátor programok számára.
A rendszer másik fő része az emulátor programok készlete. E készletből jelenleg az első rogram van működő állapotban (EM 1), amely a 2740 basic és 2741 típusú berendezésnek egfelelő interface-t biztosítja a központi ESZR gép felé, és a VT–340 képernyő, valamint VTS–56100 mikroprocesszoros képernyős terminál AP–70 típusú változatának megfelelő goritmust valósít meg a vonalak felé.

A VT–55000 szupervizora (EMSV–I)

Az EMSV–I a VT–55000 szupervizor jellegű feladatait végzi el. Ezek közül a legfontosabbak a következők:

A CCA fizikai kezelése:

A CCA három megszakítási szintet foglal le. A legmagasabb prioritású az ún. EXTREM szint, amelyen a SYSTEM RESET, a SELECTIVE RESET és a HALT I/O parancsok lekezelése történik.
A legalacsonyabb prioritású az üzemmódváltáshoz tartozó megszakítási szint (UMV IT), melyet a VT–55000 előlapján található csatorna átkapcsolók aktivizálhatnak.
A középső, az ún. NORMAL IT szintet az átviteli parancsok és a tényleges adatátvitel aktivizálja.
CCA-ra a központi ESZR gép felől érkező eseményeket a CCA-mikroprogram állítja sorba, hogy az EMSV–I feldolgozhassa és szükség esetén a megfelelő emulátor programnak továbbíthassa.
CCA-ra a VT–55000 emulátor programjai felől (bizonyos esetekben egyéb forrásból) érkező szolgálás-kérdéseket az EMSV–I állítja sorba a CCA-mikroprogram számára.
CCA-n mindkét irányban blokkos átvitel lehetséges, amelyet a CCA – mikroprogramban és CCA kezelő programban definiált vezérlő karakterek valamelyike, vagy valamelyik számítógép (a központi ESZR gép vagy a VT–55000) byte-számlálójának a lejárta állíthat le.

A CLA fizikai kezelése:

A CLA a modemek illesztését végzi, és – a CCA-hoz hasonlóan – mikroprogram vezérlésű berendezés. Ez mindkét irányban blokkos átvitelt tesz lehetővé, amelyet a CLA kezelő

programban definiált vezérlő karakterek valamelyike, vagy a VT–55000 byte-számlálój lejárta állíthat le. A CLA kezelő-programja segítségével a CLA működésmódja – a hoz csolódó vonal fizikai adottságának megfelelően – rugalmasan változtatható. A VEIKI ben jelenleg páros paritású, 1200 bd sebességű félduplex átvitelt használunk.

A real-time óra fizikai kezelése:

A real-time óra kezelő-programja sorbaállítja az időzítés kéréseket és mindig a le rabb letelő időzítést szolgálja ki.

A kezelő-program kétféle időzítés kérést különböztet meg: a feltétlen időzítés ké amelyet a megadott idő leteltével mindenképpen érvényre juttat, és a feltételes időzíté amely egy CLA művelettel összekapcsolva jelenik meg, és amelyet csak akkor hajt vég megadott CLA-esemény nem következik be.

A helyi perifériák kezelése:

A VT–55000 rendszerben a helyi perifériák csak segédfunkciókat látnak el, a fo tos adatátvitel során nincs rájuk szükség. A rendkívüli eseményeket (pl. hálózatkimarad SYSTEM RESET) a rendszer a konzolírógépre írja ki. A rendszer betöltése lehetséges szalagról, és a központi gép felől, a CCA-n keresztül.

Multitaskos környezet az emulátor programok felé:

A VT–55000 rendszerben minden termináltípushoz egy emulátor program tartoz

Az emulátor program számára a VT–55000 rendszer kötött formátumot ír elő; látor program egy közös adatszegmensből (CDS), egy programszegmensből (LPS) és an kális adatszegmensből (LDS) áll, ahány multiplex alcsatornát, illetve vonalat foglal le a terminál típus.

A megfelelő számú LDS generálását egy minta-LDS alapján, illetve hozzárendelés LPS-khez az EMSV–I végzi indítás után.

A VT–55000 emulátor programja (EM 1)

Az EM 1 emulátor program az EMSV–I-en keresztül értesül a külső eseményekr kiszolgálást a CCA, CLA és a real-time óra felé.

Az EM 1 fő feladata, hogy az ESZR nagygép felé is és a vonalak felé is megfelelő so interface-t adjon. Az ESZR nagygép felé az EM 1 a 2740 basic és 2741 tipusú berend nek megfelelő interface-t adja. A vonalakon kezeli a VT–340 típusú képernyőt és a V mikroprocesszoros képernyő AP–70 típusú változatát.

Az EM 1 a központi ESZR gép felől az alábbi parancsokat ismeri fel érvényes p ként:

- SYSTEM RESET: hatására az EM 1 alapállapotba kerül.
- HALT I/O: átvitel leállítása.
- Enable parancs hatására a modemet készenléti állapotba teszi.
- Write parancs: blokk kiküldése a terminálra.
- Prepare parancs: vétel előkészítése.
- Read, vagy Inhibit parancs: blokk vétele.
- Disable parancs: hatására CE, DE státuszt küld.

Az érvénytelen parancsokat CE, DE, UC státusszal és Command Reject sense bit kezdeti, a Disable vagy SYSTEM RESET utáni – Enable-től különböző – parancsokat Intervention Required sense bittel utasítja vissza.

A vonali hibák lekezelésére a következő általános elvek szerint működik az EM 1:

– A CLA által jelzett fizikai hibákra a terminál típustól függő intézkedést tesz:

a) VT–340: CE, DE, UC státuszt küld és beállítja a Data Check sense bitet.

b) VTS–56100: véges számú kísérletet tesz a hiba elhárítására, s ha ez sikertelen, akkor CE, DE, UC státuszt küld és beállítja az Equipment Check sense bitet.

– A VTS–56100 típusú terminál esetén, a hosszparitás karakter által jelzett blokk-átvi-eli hiba esetén véges számú kísérletet tesz a blokk átvitelére, s ha ez sikertelen, akkor CE, E, UC státuszt küld és beállítja az Equipment Check sense bitet.

– Ha a terminál nem válaszol, akkor a termináltípustól függő intézkedést tesz:

a) VT–340: A VT–340-re semmilyen idézítés nincs.

b) VTS–56100: Ha a VTS–56100 valamilyen olvasási parancsra nem válaszol, akkor véges számú kísérletet tesz a terminál megcímzésére.

Ha ez sikerül, akkor folytatja az eredeti műveletet, ha nem, akkor CE, DE, UC státuszt üld és beállítja az Equipment Check sense bitet.

A rendszer továbbfejlesztésére vonatkozó igények

A VT–55000 a kezdeti nehézségek leküzdése óta a software szempontból kifogástalanul zemel. Az üzemeltetés során jónéhány olyan igény merült fel, amely megkívánja a rendszer ovábbfejlesztését.

Röviden felsoroljuk a fölmerült újabb igényeket:

– *Terminálszám növelése.* A rendszer jelenleg maximum 16 vonalon összesen 32 termi-ált tud kezelni. A vonalszám növelése e fölött nem egyszerűen generálási paraméter, hanem émi program módosítást kíván.

– *Jobb memóriakihasználás.* A rendszer bővítésének nemcsak a fent említett korlátozás, anem az adott tárméret (16 Kszó) is határt szab. Ezért kívánatos a memória dinamikus keze-ése, még akkor is, ha ez valamelyes időveszteséggel jár.

– *Logikai vonalak kezelése.* Az R40 multiplex csatornacímek és a fizikai vonalak közöt-i összerendelést jelenleg egy fix, az EMSV-ben elhelyezkedő táblázat definiálja. Kívánatos, hogy rendszer logikai címekkel dolgozzék, ez a hozzárendelésben nagyobb szabadságot ad. Így je-entősen leegyszerűsödik és hatékonyabbá válik a multipont kezelése, vagy egy duplex algorit-nus megvalósítása.

– *Nagyobb flexibilitás a konfiguráció definiálásában.* Az előző ponthoz kapcsolódik, és zt továbbfejleszti az igény, hogy tetszőlegesen párosíthassunk különböző algoritmusokat.

Jelenleg az emulátor program – egy adott címhalmazra – definiálja az R40 oldali és a onali algoritmust. Kívánatos, hogy különböző R40 oldali algoritmusokat különböző vonali lgoritmusokkal párosíthassunk. Ez lehetővé tenné pl., hogy ugyanazt a terminált két külön-böző alkalmazás – esetleg két különböző fizikai alcsatornán megszólítva – két különböző erminálként kezelhesse. Ez különösen előnyös adott alkalmazások esetén.

– *Üzembiztonsági igények.* Szükséges a javítható és nem javítható hibák világosabb szét-választása, és hogy a rendszer minden hibáról statisztikát vezessen. Az is kívánatos, hogy a endszer jól definiált státusszal rendelkezzék, amely lekérdezhető és szükség esetén módosít-ható legyen.

– *A kapcsolatfelvétel és az átvitel szétválasztása.* A VT–55000 jelenleg passzív multi-plexorként üzemel. Kívánatos, hogy bizonyos önálló kapcsolatteremtő és -tartó kapcsolattal endelkezzék. Ez a lehetőség többek között a terminálkezelőkre (felhasználókra) pszichikai-ag is jó hatással van, ha pl. – az előző ponttal összefüggésben – az R40 kiesésekor a rend-zer önálló hibaüzenetet generálhat, vagy még annak bekapcsolása előtt a felhasználóval a kap-csolatot felveszi és az igényt tárolja.

– *Összekapcsolódási lehetőség egy hozzá hasonló rendszerrel.* Számos okból (megbí tóság, elosztott funkciók) kívánatos, hogy a rendszer képes legyen egy önmagával azono rendszerrel való kommunikációra. Ennek az igénynek még egy lépéssel való továbbfejlesz hogy összekapcsolható legyen önmagával „hasonló" rendszerrel. Ilyen „hasonló" rendsze egy hasonló R10-konfiguráció R40-host nélkül, terminál-koncentrátoros üzemben. Ez az az előzőeknél egy nagyságrenddel súlyosabbnak tűnik.

Az R10 rendszer továbbfejlesztése az itt felsorolt igények kielégítésére törekszik. A lesztést a VEIKI és az MTA SZTAKI munkatársai közösen végzik. A tervezési és megvaló tási munkák nagyobb részét a SZTAKI, kisebb részét a VEIKI munkatársai végzik. Az üz meltetési tapasztalatok és igények a VEIKI-ből származnak.

Irodalomjegyzék

1. Braun P., Horniák G., Nagy E., Terényi L.: Az ESZR R-40 számítógépek be- és kiviteli rend rének és terminálkezelési lehetőségeinek áttekintése. VEIKI jelentés, SZK–3, Budapest, 1974. május
2. Böszörményi L., Braun P., Horniák G., Pásztor Z., Szabados Á-né: Az ESZR R-40 számító nél megvalósítható fontosabb távadatátviteli módszerek és lehetőségek áttekintése. VEIKI tés, SZK–9, Budapest, 1974. december
3. Böszörményi L., Terényi L.: ESZR alapú adatátviteli rendszer hardware-jének kialakítása é tesztprogramja. VEIKI jelentés, SZK–13, Budapest, 1976. június.
4. Braun P., Horniák G., Pásztor Z.: A VEIKI R-40 számítógépéhez csatlakozó ESZR távadat teli rendszer adatkezelő-lekérdező rendszere. VEIKI jelentés, SZK–22, Budapest, 1976. de
5. Pásztor Z.: A NIMFORM lekérdező rendszer. Rendszerterv. VEIKI jelentés, SZK–18, Buda 1975. december

NAGY INFORMÁCIÓ RENDSZER KIALAKÍTÁSA A NIM RÉSZÉRE ÉS AZ ENNÉL ALKALMAZOTT ADATBÁZIS KEZELŐ RENDSZER

Braun Péter — Csala István
VILLAMOSENERGIAIPARI KUTATÓ INTÉZET

Az előadásban a NIM részére, a VEIKI-ben kifejlesztett információs illetve adatbázis kezelő rendszerről adunk tájékoztatást. Az információ rendszer a NIM megbízásából az illetékes IM főosztályok közvetlen vezetése és instrukciói alapján került kialakításra. A kidolgozást a EIKI Számítástechnikai főosztályának osztályai végezték.

Az információs rendszer kialakításának koncepciója 1974-ben került kidolgozásra. A fejlesztés első célja egy olyan feldolgozási rendszer és információbázis kialakítása volt, mely a minisztériumi döntési és tájékoztatási tevékenységhez szükséges információkat a kijelölt adatkörökben gépi úton állítja elő.

A rendszer alapja a meglévő adatszolgáltatási gyakorlat volt, mert ez biztosította a megvalósítás realitását. Célul tűztük ki azt is, hogy gépi úton állítsuk elő a NIM részére kötelezően előírt, továbbmenő adatszolgáltatást.

Elsőként a közgazdasági, energetikai és beruházási információcsoportok kerültek feldolgozásra.

Az energetikai információk különlegesek, mert az energetikában — különösen az ár robbanás óta — igen gyors, napi periódusú információk kellenek, másrészt pedig a NIM energetikai tekintetében országos hatáskörű.

A gyors információszolgáltatás és a továbbfejlesztés lehetőségének megteremtése érdekében egy képernyős rendszer kidolgozását végeztük el, mely a NIM kiszolgálásán kívül a tárca egyes vállalatai felé is számítógépes kapcsolatot biztosít.

Az említett információs csoportok feldolgozása során, valamint az ezekkel párhuzamosan végzett hardware-, alap- és felhasználói software fejlesztéseknél az alábbi főbb tevékenységeket végeztük.

Kialakításra került a közgazdasági alrendszer, mely a statisztikai és mérlegbeszámoló jelentésen alapul, és felöleli a tárca valamennyi vállalatát.

Rendszeresen készülnek havi, negyedéves és éves gyorsjelentések és tájékoztatók, valamint nagy számban különböző feldolgozások eseti kívánságok kielégítésére.

Gépi úton kerül kielégítésre a NIM részére kötelező statisztikai adatszolgáltatás legnagyobb része.

A NIM területén létesülő vállalati és célcsoportos beruházások kezelésére adattár és kezelő rendszer került kialakításra.

Külön rendszer került kidolgozásra a nagyberuházások határidő- és költségfigyelésére.

Az energetikai adatok terén az OEGH és ÁEEF segítségével megszervezésre került valamennyi energiahordozó főbb adatainak napi jelentő rendszere.

A fenti feladatok ellátásának támogatására megfelelő szerződések segítségével kialakítottuk a mágnesszalagon történő adatcserét PM, a MüM, az AFB és más szervekkel.

A feldolgozások és programrendszerek a VEIKI R40 számítógépére alapozva készültek. A számítógép kapacitását a feldolgozások volumenének növekedése miatt bővíteni kellett, üzembe állítása óta 29MB BASF lemezekkel és 512 KB tárkapacitással bővült, így a központi tár kapacitása jelenleg 1 MB.

A kialakított adatátviteli rendszer működtetéséhez egy lekérdező rendszer készült felhasználásra került az IBM CRJE rendszere.

A lekérdező rendszer egy közös, egységes adattár segítségével dolgozik, ebbe a kü ző adattárak megfelelő illesztő programok segítségével emelik át az adatokat. Ez a meg lehetőséget adott arra, hogy a képernyős rendszer segítségével közgazdasági, energetikai, házási, sőt tröszti adatok is elérhetők legyenek a teljes adattárak egységesítése nélkül. E utóbbira az ad módot, hogy terminálok működnek a trösztöknél is és ezek némelyike tén létrehoz lekérdező adatállományt a közös adattárban.

Nagy jelentőségűnek tartjuk, hogy így egy gyors és közvetlen adatkapcsolat alakul egyes trösztök és a minisztérium között.

Az így kialakított rendszer tapasztalatai alapján és az előre látott, valamint a mun rán jelentkező korlátozások feldolgozására a rendszer továbbfejlesztése is megtervezhető.

Az első törekvésünk az adatkezelésnek adatbáziskezelő rendszerekkel történő meg

A szolgáltatások megkívánt, gyors megindítása érdekében a direkt programozást ke használni, de amint lehetett, megkíséreltük kész rendszerek felhasználását.

Megvizsgáltuk a SAWI, MARK4, CFMS, SÁMÁN rendszereket, de részben a hozzá lehetőségeinek hiánya, részben a rendszerek nem megfelelő szolgáltatásai miatt, ezek has latát nem vezettük be.

A problémák sürgető megoldása nem engedte meg a hosszú várakozást, ezért egy célrao tált adattárkezelő rendszert alakítottunk ki.

A feladat, amelyet az R40-es ESZR gépre implementálható adatbázis-kezelő rendsz a NIM információs rendszerében meg kell oldani, a következőkben foglalható össze.

Néhány száz adatszolgáltató pénzügyi-, statisztikai adatait kell az adatbázisban tár úgy, hogy ne csak az alapadatok gyors visszakeresése, hanem azokkal közgazdasági-, stat tikai-, elemző számítások elvégzése és az eredmények tárolása is biztosított legyen.

Az adatok különböző bizonylatokon más-más gyakorisággal jelennek meg, és válto adatszolgáltatók jelentési kötelezettsége is.

A rendszer legyen alkalmas az adatszolgáltatók különböző szempontok szerinti cso sítására és az adatok ennek megfelelő összegezésére.

A már említett számításokat – kívánság szerint – az összesített adatokon is hajts

A nem-programozó felhasználók programozói munka igénybevétele nélkül tudjanak hoc számításokat elvégeztetni és output igényeket kielégíteni.

Az adatbázisba több időszak azonos adatait kell tárolni, hogy azokból idősorokat hessünk.

A triviális feladatok felsorolására nem térünk ki.

A VEIKI SZF illetékes vezetői vállalva az esetleges kudarc kockázatát, 1977-ben e délyezték a fejlesztő munka megindítását.

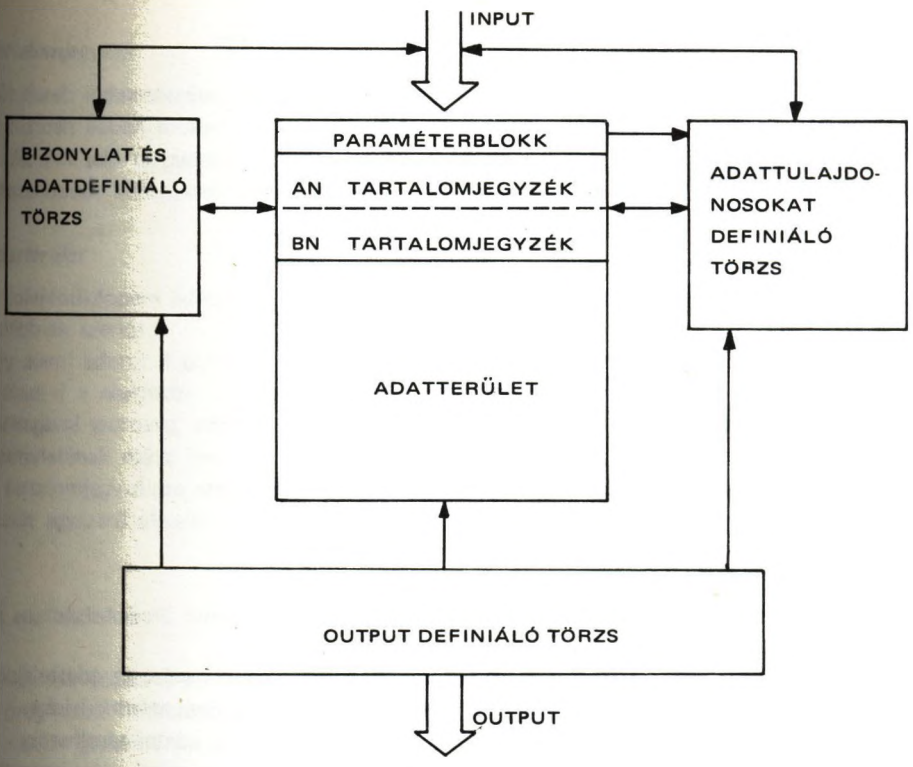
Az ismertetésre kerülő rendszerrel jelenleg sikeres próbafeldolgozások folynak, és ra tervezzük az éles feldolgozások kezdetét.

A rendszer alapgondolata: olyan tárolási és hozzáférési módszer megvalósítása, me vetni képes a gyakori *bizonylat orientált input és a táblázat orientált output* változásait, programozói munka igénybevétele nélkül.

Mivel az adatfeldolgozások jelentős részénél ezt az alapvető feladatot meg kell old reméljük, hogy a rendszer más területeken is használhatónak bizonyul.

Az adatbázis szerkezete

Az adatbázis főbb egységeit és a köztük fennálló kapcsolatot az 1. ábra szemlélte



1. ábra

Vegyük sorra az egyes adatállományok rendeltetését.

Bizonylat- és adat definiáló törzsállomány:

Változó hosszúságú rekordokat tartalmazó IS állomány. Rendeltetése az, hogy a rendszer számára definiálja azokat az *alap- és képzett bizonylatokat,* melyek az adattárban előfordulhatnak.

A lehetséges rekordterveket a 2. ábra tartalmazza.

RL	KULCS		BGY	ESZ	BIZONYLAT NÉV
	BSZ	ϕϕϕ			

RL	KULCS		M.EGYS	T.	F.	ADATNÉV
	BSZ	ESZ				

RL	KULCS		M.EGYS	T.	F.	MEGNV = KÉPLET
	BSZ	ESZ				

2. ábra

A *bizonylatleíró* rekordtípus a következő információkat tárolja:
- bizonylatszám,
- bizonylat megnevezése,
- a bizonylatban lévő elemi adatok száma,
- a bizonylat 1 éven belüli gyakorisága,

Az *adatleíró* rekordtípusnak két változata van:
- *alapadat* leíró,
- *számított adatot* leíró,

Ezek a következő információkat tartalmazzák:
- bizonylatszám,
- elemszám (a bizonylaton belüli sorszám)
- az adat megnevezése,
- az adat mértékegysége,
- a tizedesjegyek száma,
- egy 10 alapú szorzófaktor hatványkitevője,
- a számítási algoritmus képlete (számított adatnál)

Adattulajdonosokat definiáló törzsállomány

Fix hosszúságú rekordokat tartalmazó 18 adatállomány. Rendeltetése az adattulajdonosok (adatszolgáltatók és csoportjaik), valamint a közöttük fennálló *kapcsolatok* leírása.

Adattárban csak a törzsállományban definiált adattulajdonosok adatai tárolhatók. A fix hosszúságú rekord felépítését az adatbázis generálásakor kell megadni, mint generálási paramétert (és ezt az információt a rendszer tárolja). Igy a felhasználónak tág lehetősége van annak eldöntésére, hogy az adattulajdonosokról milyen – a feldolgozások során szükséges – információkat kíván tárolni. (Pl. megnevezés, cím, telefonszám, különböző csoportosítási ismérvek, listázási sorrendek, stb.)

Outputokat definiáló törzsállomány

Változó hosszúságú rekordokból álló IS adatállomány. Rendeltetése, a felhasználó által igényelt tabló outputok „táblatervének" leírása a rendszer számára.

Adattár

Fix hosszúságú rekordokból felépülő DA adatállomány. Rekord-azonosító a *relatív cím.* Az allokált lemezterületet generáláskor paraméterek alapján a következő főbb részekre osztjuk:
- paraméterblokk,
- tartalomjegyzék,
- adatterület.

Paraméterblokk

Mindig az adattár első blokkja. Azokat a rendszerjellemzőket tartalmazza, melyeket az adatfüggetlen programok működésük során felhasználnak.

Tartalomjegyzék

Két darab index-szerűen felépülő címtárból áll. A BN jelű tartalomjegyzék gyors visszaeresést biztosít abban az esetben, ha sok adattulajdonos azonos bizonylatának adataira van szükség. Az AN jelű tartalomjegyzék az előző inverze és akkor használja a rendszer, ha egy adattulajdonos különböző bizonylatokban szereplő adatait kell feldolgozni.

Adatterület

A tulajdonképpeni adatok tárolására szolgál. Az adatok elhelyezése *tömbönként* történik az alábbiak szerint.

Egy elemi adatot 4 byte-ban binárisan tárolunk. Igy egy bizonylat összes adatának egyszeri tárolása *4 x elemszám* nagyságú területet igényel. Ezt a területet a bizonylat megjelenési gyakoriságával szorozva, adódik az a tömbméret, amely szükséges egy adattulajdonos valamely bizonylatának egész éven át történő tárolásához.

A tartalomjegyzékbe ennek a tömbnek a kezdő címe tárolódik. A tömbön belüli címek kiszámítását egyszerű algoritmus végzi.

Az adatbázis-kezelő rendszer felépítése és működése

A rendszer részei a 3. ábrán láthatók.

A programok funkcióik alapján 3 fő csoportba sorolhatók:

- adattár generáló program
- adatbázis-kezelő programok
- rendszer segédprogramok.

A rendszer működésének lényege a következőkben foglalható össze. Miután kellő mérlegelés alapján döntöttünk az adattulajdonosokat és kapcsolataikat definiáló rekord felépítését, valamint az egyidőben tárolni kívánt adatok várható mennyiségéről figyelembe véve meghatároztuk az adattár nagyságát összeállítjuk a generálási paramétereket.

A KABaATGE nevű program ezek alapján létrehozza az üres adattárat. Az első ún. paraméter blokkba beírja a rendszer paramétereit.

Rendszer segédprogramok segítségével létrehozzuk (KAB=UNIN, KAB=RABP) — és betöltjük (KAB=ISFT) az IS törzsállományokat.

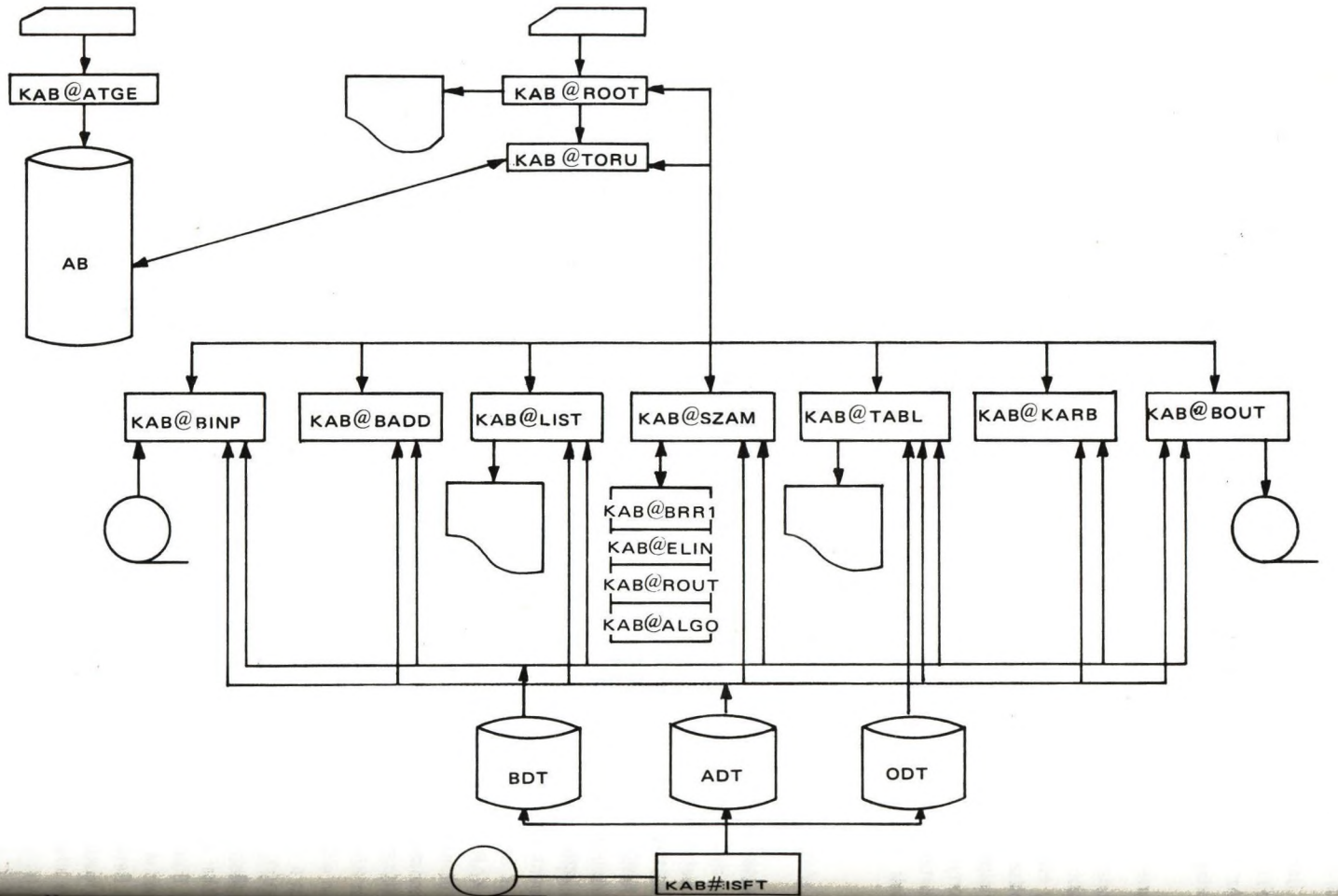
Ezt követően kezdődhet az adattár — felhasználói programokkal ellenőrzött és előírt formában előállított — adatokkal való feltöltése.

Ezt a funkciót már az adatbázis-kezelő rendszer hajtja végre, melynek működését a KABaROOT nevű program szervezi és irányítja. Ennek fő tevékenysége abból áll, hogy:

1. Betölti a KABaIORU nevű load modult, mely az adattár felé irányuló összes I/O művelet fizikai lebonyolítását biztosítja.
2. Beolvassa, ellenőrzi és listázza a végrehajtandó funkciókat meghatározó vezérkártyát, továbbá az egyes funkciók végrehajtásához szükséges paraméter kártyákat.
3. Betölti és indítja a vezérkártyán előírt funkció végrehajtását biztosító load modult, majd a feladat elvégzése után törli.

Az adatátviteli funkciót MOD = BINP vezérkártyával lehet aktivizálni.

A bizonylatszám és adattulajdonosi kód szerint rendezett input rekordokat beolvassa, azonosítóit ellenőrzi és elhelyezi őket az adattárban. A bizonylatonként bevitt adatok elhelyezése a már említett tömbökben történik. A bizonylat tömbön belüli helyét a bizonylatra érvényes idő szabja meg.


KAB@ATGE
KAB@ROOT
KAB@TORU
AB
KAB@BINP
KAB@BADD
KAB@LIST
KAB@SZAM
KAB@TABL
KAB@KARB
KAB@BOUT
KAB@BRR1
KAB@ELIN
KAB@ROUT
KAB@ALGO
BDT
ADT
ODT
KAB#ISFT

A tömb allokálása és kezdőcímének a BN jelű tartalomjegyzékbe való beírása mindig az dott bizonylattípus első bevitelekor történik meg. A további bizonylatok a már kijelölt tömb egfelelő helyére kerülnek.

Ha az adatszolgáltatók egy meghatározott típusú bizonylata mind bevitelre került, akkor épezhetők ennek a bizonylatnak különböző aggregátumai az adatszolgáltatók előírt csoportotása szerint.

Ezt a rendszer-funkciót a MOD = BADD vezérkártyával lehet kérni.

A feladatot végrehajtó KABaBADD nevű modul számára, paraméterkártyán a következő információkat kell megadni:

SZ = nnn — az összegezendő bizonylattípus száma

AT = ee/ii — a bizonylatra vonatkozó dátum év/időszak formában

SK = j=x,y,z,.. — az adattulajdonosi rekord azon csoportképző ismérvet tartalmazó mezőjének a jele, mely az aggregátumképzést megszabja (pl. szakágazati jel). Ha az aggregátumképzést csupán a csoportképző ismérv meghatározott értékeire akarjuk végrehajtani, akkor ezeket az értékeket is meg kell adni.

A modul gondoskodik a létrehozott aggregátum adattárban történő elhelyezéséről és a számára lefoglalt tömb kezdőcímének a BN jelű tartalomjegyzékbe való beírásáról.

Az adattárban elhelyezett adatok, bizonylatonként és adattulajdonosonként kilistázhatók.

Ezt a rendszerszolgáltatást a MOD = LIST vezérkártyával kérhetjük.

A feladatot végrehajtó KABaLIST modul számára a következő adatokat kell paraméterkártyán közölni:

IZSZ = nnn — bizonylatszám

OSZ = ee/ii — a bizonylatra vonatkozó dátum

DTUL = k_1, k_2 ... k_n — az adattulajdonosok azonosító kódja

A rendszer egyik legbonyolultabb és legfontosabb feladata az adatokkal történő tetszőleges számítások lehetőségének biztosítása. Tetszőleges alatt feltétlen és feltételes arimetikai utasítások értendők, melyeket megfelelő szintaktikai szabályok betartásával a rendszer számára bármikor definiálhatunk egy számított adatot leíró rekord segítségével.

A számítási funkciót a MOD = SZÁM vezérkártyával indíthatjuk.

A feladat végrehajtását szervező KABaSZÁM nevű modul a következő paramétereket igényli:

KCS = nnn — annak a képzett bizonylatnak a számát kell megadni, amely a végrehajtandó képletcsoportot tartalmazza.

TID = ee/ii — a képletekben szereplő elemi adatokra vonatkozó bázisidőszak
A képletekben olyan elemi adatok is szerepelhetnek, amelyek érvényességi ideje a bázisidőtől ± irányban eltérhet.

ATI = jj/k_1 – k_n) — Ezzel a paraméterrel az adattulajdonosoknak az a csoportja (részhalmaza) jelölheti ki, amelyekre nézve a számításokat el kell végezni.
jj — az adattulajdonosi rekord egy mezőjének a jele. A zárójelben ezen mező két értéke szerepel.

A modul a képleteket és az adattulajdonosok azonosítóit a megfelelő törzsállományokból beolvassa.

A képleteket a KABaALGO nevű speciális fordító program gépi utasításokká alakítja.

A modul felépíti azokat a táblázatokat, amelyek az operandusként szereplő elemi adatok gyors behozatalát – (KABaELIN) –, valamint az eredmények kivitelét – (KABaROUT) – biztosítják.

Következő rendszer-funkció a táblázatos outputok előállítása. Ennek aktivizálása a MOD = TABL vezérkártyával történik.

A feladatot végrehajtó KABaTABL modul számára két paraméter megadása kötele (vannak opcionális paraméterek is).

A *TAB = nnnn* paraméterrel egy — az Output definiáló törzsállományban megtalá tó — *táblaterv* leírására hivatkozunk. A program ezen táblaterv alapján dolgozik.

A *TID = ee/ii* paraméter a táblázatban előforduló adatok *bázisidejét* definiálja. A táblatervek leírása lehetőséget ad a bázisidőtől való eltérésekre is. Ez idősoros tablók állítását teszi lehetővé.

A rendszer MOD = BOUT vezérkártyával indítható funkciója lehetőséget biztosít a aktualitásukat vesztett adatok archiválására, vagy éppen aktuális adatok mágneses adathc zón történő átadására más szervek felé, stb.

A MOD = KARB vezérkártyával indítható rendszerszolgáltatás feladatai közé tartoz adattár mindenkori aktuális állapotának a kiíratása, a tartalomjegyzékek listázása, stb.

A felsoroltakon kívül kidolgozás alatt áll további rendszerfunkciók megvalósítása i Ilyen például adattartalom szerinti visszakeresés, vagy grafikon-szerű outputok készítése,

A próbafeldolgozások eddigi tapasztalatai a gyorsaságot illetően is nagyon kedvezőe Például a legbonyolultabb feladatokat végző számítási funkció végrehajtása során 250 adó tulajdonosra 20—20 képlet számításainak elvégzéséhez 56, 46 sec CPU időt használt a r szer R40-en (A tényleges gépben tartózkodás kevesebb, mint 10 perc volt.)

A NAGYVÁROSI ÁLTALÁNOS ISKOLAI BEISKOLÁZÁS SZÁMÍTÓGÉPES OPTIMALIZÁLÁSA

Dr. Broczkó Péter
ÁNH

Bevezetés

Hazánkban az elért matematikai eredmények és számítástechnikai lehetőségek hasznosítását jelentős mértékben hátráltatja, hogy a magyméretű és a komplex modellek információellátása megoldatlan. Az eddigi információs rendszerek — érthetően — a hagyományos módszerekkel végzett elemzések adatszükségletéhez igazodtak, a számítógépes matematikai módszerek alkalmazásához szükséges minőségi és mennyiségi igényeket nem képesek kielégíteni.

Minőségileg új alapadatforrást jelent a személyi vonatkozású makroszintű döntések előkészítésében a hazánkban most kialakítás alatt álló — ilyen formában a világon egyedülálló — állami népességnyilvántartás.

Mi a népességnyilvántartás? A Népköztársaság Elnöki Tanácsa 1974. évi 8. számú törvényerejű rendeletével létrehozta az Állami Népességnyilvántartó Hivatalt (ÁNH), melynek feladata a Magyarországon állandó lakóhellyel rendelkező állampolgárok alapvető személyazonosító adatainak és lakcímének nyilvántartása. Az ÁNH első feladata egy olyan, az országban élő bármely személyt egyedileg azonosító személyi szám képzése, amely lehetőséget biztosít arra, hogy különféle ágazati nyilvántartások (oktatásügyi, egészségügyi, munkaügyi stb.) az egységes népességnyilvántartáshoz saját adataikkal kapcsolódjanak.

A népességnyilvántartás a személyi jellegű makroszintű és komplex feladatok korszerű adatforrása, ezért a hasznosításnál ki kell használni az elektronikus számítógépek nyújtotta, minőségileg új lehetőségeket is.

Ez azt jelenti, hogy

— az utólagos, regisztratív adatfeldolgozás mellett tervező, operatív jellegű adatfeldolgozást is folytassunk;

— a rutindöntéseket számítógéppel végezzük.

A jelen anyagban ezen kritériumokat kielégíteni kívánó megoldást ismertetünk.

I. A feladat aktualitása

A manuálisan végzett nagyvárosi általános iskolai beiskolázás hiányosságainak következményeit vizsgáljuk meg két szempontból:

1. a gyermekek és
2. a szülők szempontjából,

majd pedig általánosítsuk a kapott következtetéseinket.

1. A nagyvárosokban távoli iskolába való járás esetén a gyermekek jövet-menet számtalan olyan veszélyhelyzetnek vannak kitéve (forgalmas úton való átkelés, tömegközlekedési eszközökön való átszállás stb.), amelyek közül csak egyetlen egynek az egyszeri szerencsétlen alakulása a gyermek számára súlyos, életre szóló következményekkel járhat, esetleg pedig végzetessé válhat.

Ezeken a gyermekeken pedig elsősorban nem az „Óvjuk a gyermekeket a közlekedésben" kampányok segítenek (bár természetesen ezek is hasznosak), hanem a közelebb lévő, kedvezőbben megközelíthető iskolába való beiskolázás.

2. A szülők szempontjából vizsgálva – beiskolázást, képzelhetjük, mit ér annak a s nek a munkanapja, aki képtelen megszervezni a nagyvárosban egy távoli iskolába járó kis mekének az oda-vissza kísérését.

Vagy mit jelent az egyéni kérések merev elutasítása a többgyermekes szülőknek, mi reggel a gyermekeiket több iskolában kell leadni, este pedig többől kell összeszedni.

Végeredményben tehát megállapíthatjuk, hogy az optimális általános iskolai beiskol napjaink egy komoly infrastruktúrális kérdése. Mindezideig azért nem jelentkezett megold feladatként, mivel nem voltak meg a feltételek a megoldás korszerű módszerekkel történő mogatására. Az állami népességnyilvántartás adatbázisának kialakításával viszont létrejött a említett problémák hatékony kiküszöbölésének potenciális lehetősége.

II. A feladatmegoldás matematikai modellje

A számítógépes rendszer matematikai modellje végzi a beiskolázandó gyermekek álla lakcíme és az iskola közti távolság minimalizálása révén a beiskolázás optimalizálását.

Az alkalmazott jelölések értelmezése:

a_i – az i-edik fizikai címen lakó beiskolázandó gyermekek száma

m – a beiskolázandó gyerekekhez tartozó fizikai címek száma

b_j – a j-edik fizikai címen lévő iskola első osztályai férőhelyeinek száma

n – az iskolák fizikai címeinek száma

x_{ij} – az i-edik – a beiskolázandó gyermekekhez tartozó – fizikai címről és a j-edik is lai fizikai címre beiskolázott gyermekek száma

c_{ij} – az i-edik – a beiskolázandó gyermekekhez tartozó – fizikai címnek és a j-edik i lai fizikai címnek a távolsága.

A matematikai modell

$$\sum_{j=1}^{n} x_{ij} = a_i \qquad i = 1, 2, \ldots, m$$

$$\sum_{i=1}^{m} x_{ij} = b_j \qquad j = 1, 2, \ldots, n$$

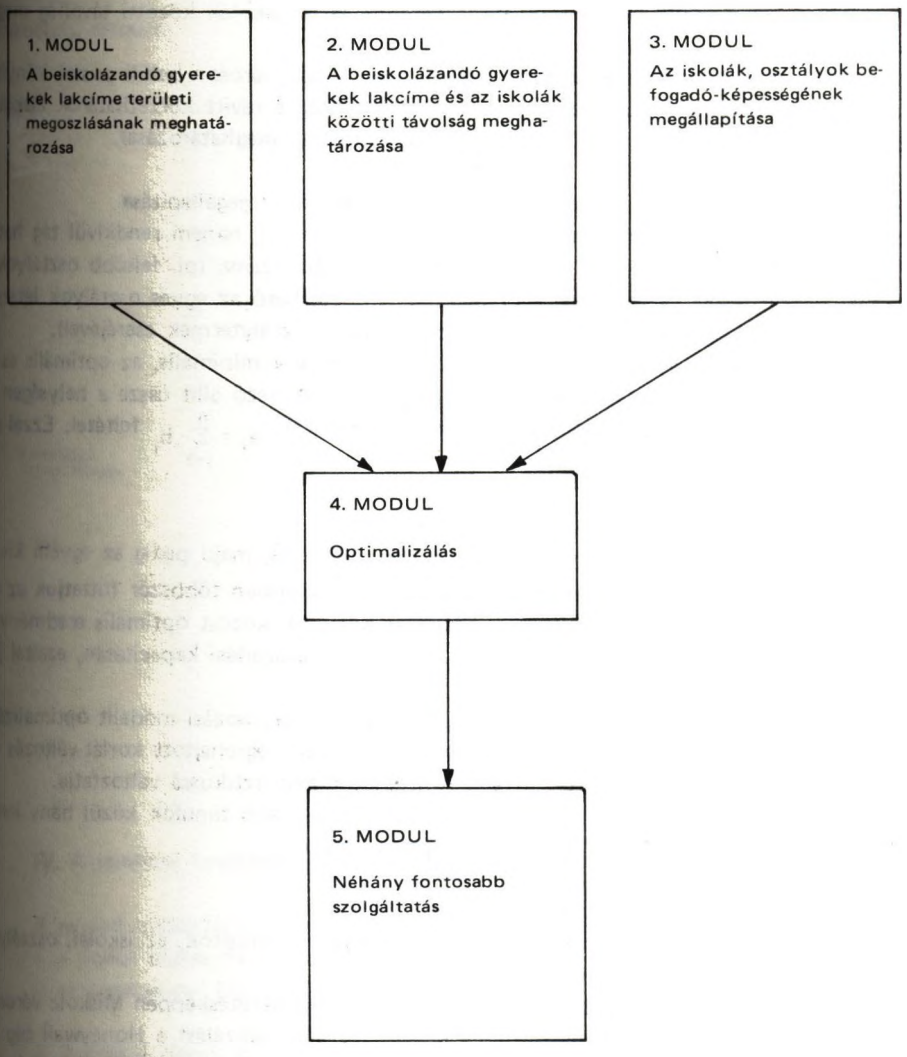
$$x_{ij} \geqslant 0$$

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{m} b_j$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} \; x_{ij} \rightarrow \min$$

III. A számítógépes tervezési rendszer

A beiskolázás optimatizálását végző modellnek három bemenő adatállománya van é futása eredményeképpen kapott adatállományból teljesítjük a szolgáltatásokat. A feldolgo ban résztvevő állományok egy-egy tevékenységsorozat eredményei illetve forrásadatai. A t kenységek egymáshoz való kapcsolódását az ábra szemlélteti, tekintsük át mi is ennek m lelően a feldolgozás menetét.



A számítógépes tervezési rendszer nagyvonalú folyamatábrája

1. MODUL A beiskolázandó gyermekek lakcíme területi megoszlásának meghatározása.

A nagyvárosi beiskolázandó gyermekek (azok a gyermekek, akik az adott év szeptember 1-ig töltik be a 6. életévüket) adatait kiírjuk a népességnyilvántartási adatbázisból. Nagyvárosnak ebből a szempontból elvben számíthatunk minden olyan helységet, ahol legalább két helyen található elsőosztály. Meg kell jegyezni, hogy a modell hatékonyan csak a nagyobb városok esetén üzemeltethető, ahol sok helyen van első osztály.

Szétválogatjuk a gyermekeket helységenként és a továbbiakban csak az egyes helységekben végzendő munkákat ismertetjük.

Az adott helységen belül kisebb körzetekbe soroljuk a gyermekeket és megállapítjuk, hogy az egyes körzetekbe hány gyermek esik (az $\bar{a}$ vektor meghatározása).

2. MODUL A beiskolázandó gyermekek lakcíme és az iskolák közötti távolság tározása

A jelenleg már megvalósított esetben ideiglenesen a nagyvároson belüli postai irá számokat használtuk fel körzetazonosítónak és a várostérképre rávitt körzethatárok ala határoztuk meg a köztük lévő relatív távolságot. (a C mátrix meghatározása).

3. MODUL Az iskolák, osztályok befogadóképességének megállapítása

Az iskolák, osztályok befogadóképessége nem merev szám, hanem rendkívül tág között mozoghat. Változtatható az indítható elsőosztályosok száma (pl. felsőbb osztály szevonásával, többváltásos tanítás megszervezésével) változtatható az egyes osztályok lé is (pl. pótlólagos padok beállításával, különböző nagyságú osztálytermek cseréjével).

Az iskolák egy erre a célra kialakított űrlapon jelentik a minimális, az optimális maximális elsőosztályos férőhelyszámot. Ezek alapján a modellező állít össze a helység lül egy olyan iskolai kapacitás-sorozatot, hogy teljesüljön a $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$ feltétel. Ezz határozásra kerül a $\overline{b}$ vektor.

4. MODUL Optimalizálás

Maga az optimalizálás két fázisban történik, egy előzetes, majd pedig az egyéni gok figyelembevétele után egy végleges fázisban. Mindkét fázisban többször futtatjuk malizálást végző modellt, ugyanis a kapott, az adott korlátok között optimális eredmé elemezve célszerűen megválasztva változtatunk az iskolák befogadási kapacitásán, ezált tősen javíthatunk az előzőleg kapott optimális eredményen.

Megjegyezzük, hogy bár számítógéppel egy lineáris programozási modellt optimal viszont az eredményelemzés és az elemzés eredményeképpen végrehajtott korlát-változt olyan beavatkozás, amely végső soron a feladatmegoldást heurisztikussá változtatja.

A kapott megoldás megadja, hogy az i-edik körzetben lakó tanulók közül hány j-edik körzetben lévő iskolába (az X mártix meghatározása).

5. MODUL Néhány fontosabb szolgáltatás

Ezek közé tartoznak a gyermekek számára nyomtatott értesítők, az iskolai, oszt kénti névjegyzékek és a beiskolázás statisztikai elemzése.

A modell jelenleg még kísérleti stádiumban van. Szemléletetésképpen Miskolc vá natkozásában éles adatokkal is megvalósításra került. Az optimalizálást a Honeywell c Mathematical Programing System (MPS) programcsomagja végezte. Az adatok illesztés programcsomagról való leszedésére 13 felhasználói programot kellett készíteni, melyek FORTRAN, GMAP és ACL nyelven készültek.

Különösen kedvezőek a futási időre vonatkozó tapasztalatok. Maga az optimaliz 5–6 perc start-stop időt vett igénybe, mely az iskolai befogadási kapacitás változtatás végzett utóoptimalizálási eljárás során 50%-kal csökkent, mivel az optimalizálási folyam nem elölről, hanem az előző optimális megoldás kimentett bázis-megoldásától indított

A megvalósított modell legnagyobb hiányossága az, hogy a távolságadatokat visz durván, a postai irányítószámok alapján meghatározott relatív távolságok alapján szám Ennek elsősorban az volt az oka, hogy a postai irányítószám szerepel a népességnyilv si adatok között és így jelentősebb kódolási ráfordítások nélkül (2945 gyermekre vég el az optimalizálást), a gyakorlatban végigpróbálhattuk az iskolai befogadó-kapacitás m lelő változtatása útján elért fokozatos eredményjavítási módszert és kaphattunk egy, a timális megoldást szemléltető eredményt. Ennek alapján osztálynévsorokat és beiskolá tesítőket nyomtattunk (1. melléklet).

1. melléklet

Beiskolázási értesítő:

Kedves Pajtás !

Szeretettel köszöntünk abból az alkalomból, hogy hamarosan iskolás leszel.

A leendő iskolád - az előzetes felmérések szerint - a

[illegible] 5.

szám alatt található. Amennyiben ez nem megfelelő, erről 5 napon belül értesitést várunk. Egyuttal felhivjuk figyelmed, hogy körzeti orvosod két héten belül felülvizsgálatra vár.

Mégegyszer gratulálunk neked, mint ifju iskolásnak, kivánunk jó egészséget, jó tanulást:

A leendő iskolád
pedagógusai

Hivatalból
dijátalányozva !

IV. A rendszer továbbfejlesztésének lehetőségei

A modell továbbfejlesztése két vonalon történhet:
– a fizikai címek és a távolságszámítás fejlesztése:
– az oktatásügyi bázis fejlesztése.

1. A fizikai címek és a távolságszámítás fejlesztése

Mivel az ezirányban történő eredményes fejlesztés a modell gyakorlati bevezetésének feltétele, ezért ebben az irányban különösen intenzíven dolgozunk.

A modell szemléltetésképpen történő megvalósítása irányítószámokkal azaz, ún. *területi jellegű fizikai címekkel* történt. Járható útnak tűnik az adott városoknak az irányítószámoknál mélyebb pl. háztömb mélységű felbontása, manuális kódolása. Vizsgáltuk a számítógépes kódolás kérdését is, azonban a jelenlegi kaotikus lakcímhelyzetben ez kilátástalannak tűnik.

Vizsgáltuk a *pontszerű fizikai címek* kezelési lehetőségét, mely modellünknél a legpontosabb megoldást nyújthatja. Ennél manuálisan kódoltuk az egyes személyek, illetve iskolák síkbeli koordinátáit. Készítünk egy IDS–I adatbázis szervezésű számítógépes térképet, melynek segítségével nem légvonalbeli lakóhely-iskola távolsággal számolhatunk (a légvonalbeli gyakorlatilag használhatatlan, hiszen hiába van 20 méterre a lakás és az iskola, ha van köztük egy patak és csak kilométeres kerülővel közlekedhetünk köztük), hanem a tényleges távolsággal. Sőt a térképen haladva súlyozhatjuk kisebb súllyal a tömegközlekedési eszközök útvonalán történő közlekedést, nagyobb súllyal a forgalmas utakon való gyalogos áthaladást,

így a beiskolázási feldolgozáshoz még megfelelőbb távolságadatokat kapunk. Az optimali előkészítésénél meghatározzuk minden gyermek állandó lakóhelyének a legközelebb eső 5 kolától való súlyozott távolságát és ezekkel a távolságadatokkal hajtjuk végre az optimali lást.

Jelenleg már próbaadatokon működik a számítógépes térképet tartalmazó IDS–I ad ziskezelő rendszerrel működő adatbázist megkreáló, valamint a „térképen" a közlekedést ző program.

A megoldás nagy hátránya, hogy bár a „térkép" maga folyamatos karbantartással é át alkalmazható, a gyermekek lakcím-koordinátáit évenként újra kell manuálisan kódolni.

A számítógéppel való kódolás érdekében az ingatlannyilvántartásban rejlő lehetőség vizsgáltuk. Azonban az ingatlannyilvántartás jelenlegi azonosítója (helyrejzi szám) és adat talma nem teszi ezt lehetővé.

2. Az oktatásügyi bázis fejlesztése

A pedagógiai szakirodalomban széles körben ismertetett kompenzáló iskolamodell fe tése kívánkozik ide, melynek lényege a következő. Tekintsünk el a jelenleg a világon álta nosan elfogadott merev életkor-határhoz kötött beiskolázástól. A gyermekek testi és szel fejlődése ugyanis eltérő ütemű és hiába homogén egy osztály életkor szerint, ha fejlettsé szempontjából heterogén.

Sok pro és kontra állásfoglalás van a kompenzáló iskolamodellről. Itt a feladatunk csán csupán két dologra kellene felhívni a figyelmet:

1. A kompenzáló iskolamodell lehetővé teszi az iskolákban nagy problémákat okoz mográfiai hullámzásokból fakadó korosztályonkénti jelentős létszámhullámzás kiegyenlítés mely a fejlettségi „mérce" változtatásával könnyen elérhető.

2. A kompenzáló iskolamodelltől való idegenkedés alapvetően a fejlettség mértéké mérési nehézségeire vezethető vissza. E tekintetben a népességnyilvántartás létrehozása, a mélyi szám bevezetése korlátlan új lehetőséget nyithat meg. Ez ugyanis lehetővé teszi a g mekek egészségügyi helyzetének, testi és szellemi fejlődési történetének nyomonkísérését a fejlettség mértékének elbírálása nem egy adott időpontban végzett vizsgálat, teszt alapj hanem a születéstől rendszeresen végzett ilyen vizsgálatok sorozata alapján dinamikusan é sokkal megbízhatóbban lenne végezhető.

NEV	ANYJA NEVE	SZEMELYI SZAM/SZULHELY			LAKCIM	
BARTAL ILONA	KISS ILONA	27212092504 MISKOLC II	3508	NAGY ILONA	95 000018	0 00 3 0 0
BUDRIGA MAGDOLNA	FARKAS IBOLYA	27305085697 MISKOLC II	3508	FUT+	95 000023	0 00 1 0 0
CSONKA KATALIN	NYESTE JOLAN	27306176503 NYIREGYHAZA	3508	JAGENY#S	95 000027	0 00 0 0
DARAGICS GABRIELLA	LADANYI MAGDOLNA	27306096193 MISKOLC II	3508	CSARAVEZ#R	94 000030	0 00 2 0 0
DERKAS IBOLYA	SZAR+ MARIA	27302163563 GELEJ	3508	GARDAI SANDOR	95 000002	0 00 0 0
DUMA MARIANNA	SANDOR MARIA	27306226607 TORNYOSNEMETI	3508	HOCH JANOS	95 000006	0 00 0 0
FERENCZI GIZELLA	VAJK+ ERZSEBET	27303044001 MISKOLC I	3508	HEJA	95 000046	0 00 0 0
GACSI IRMA	T+TH ETELKA	27306066128 MISKOLC I	3508	CEMENTGYAR	FF 000000	2 01 17 2 0
GERCSAK ELLA	DOROS ILONA	27302183632 RAKACA	3508	GARDAI SANDOR	95 000010	0 00 0 0
GAZ RITA	MOLNAR ZSUZSANNA	27212282621 MEZONAGYMIHALY	3505	KISDOBOS	94 000008	0 00 0 0
GYETVAN IZABELLA	ALEXAI ILONA	27301303183 GYONGYOS	3508	EGRI	95 000009	0 00 0 0
HAJT+S LAURA	BIHARI ERZSEBET	27305285209 SAJ+SZENTPETER	3508	CSARAVEZ#R	94 000035	0 00 0 0
HERBAK EMMA	HERBAK ROZALIA	27308251723 BUDAPEST	3508	SZPUTNYIK	95 000036	0 00 0 0
IVANICS KLEMENTINA	CSENDOM BORBALA	27301132908 MISKOLC I	3508	ARANY JANOS	95 000136	0 00 0 0
KISS ERIKA	TIHI MARIA	27301092791 MISKOLC II	3508	IZS+ MIKL+S	95 000025	0 00 0 0
KOVACS ISTVAN	MARK+ MARIA	17212209478 MISKOLC I	3508	FUT+	95 000013	0 01 3 0 0

EGY FA–STRUKTÚRÁJÚ ON–LINE KÉPKIÉRTÉKELŐ RENDSZER (SEGAMS)

Csirik J.–Csernay L.–Makay Á.–Máté E.
JATE – SZOTE

Az elmúlt években a gammakamerákhoz kapcsolt kisszámítógéprendszerek a napi rutinvizsgálatok nélkülözhetetlen eszközei lettek az izotópdiagnosztikai laboratóriumban. Általános kijelentés, hogy egy nukleáris medicinai rendszer

a) tartalmazza az előforduló feladatok megoldásához szükséges feldolgozási lehetőségeket (funkciók)
b) biztosítsa összetett kiértékelési eljárások végrehajtásához e funkciók összekapcsolhatóságát (komplex programok).
c) egy felépítési elvvel segítse a rendszer használatát, a vizsgálatok értékelését (különböző színű nyomógombok, előre definiálható makroutasítások, feladatra orientált célnyelv stb.).

A napi gyakorlat szempontjából a különböző rendszerek nemcsak a képfeldolgozó funkciók számában és algoritmusaiban különböznek, hanem elsősorban a rendszer használatának módjában. Egy adott rendszer alkalmazásakor ezen használati módtól függően különböző fokú számítástechnikai képzettség, továbbá a kiértékelési funkciók és azok célszerű csoportosításának ismerete elengedhetetlenül szükséges. Az értékelő orvosok az előbb felsorolt ismeretek elégtelen volta miatt a rendszerek által nyújtott összes lehetőséget ritkán használják ki; így a nyerhető információk egy része elvész, vagy csak jelentős fáradsággal és időveszteséggel kapható meg. Ezen gyakorlati problémát vizsgálatát célul tűztük ki és ennek során először különböző adatfeldolgozó rendszerek használati módjait tanulmányoztuk. Megállapítottuk, hogy az elérhetőnél általában kisebb hatékonyság oka elsősorban az, hogy a rendszerek készítői az orvosokról a szükségesnél nagyobb számítástechnikai felkészültséget és memorizálási képességet tételeznek fel. Ez azt eredményezi, hogy a rendszer használata a program tervezője és írója szempontjából ugyan valóban egyszerű, de az orvos feladaláról – még egy kézikönyv vagy egy ún. „command table" megfelelő gyorsaságú forgatásának sajátítása esetén is – fáradságos, sok hibalehetőséggel járó folyamat marad.

Az egyszerűsítési törekvések ellenére általában hiányzik egy olyan – a felhasználó szempontjából következetesen végigvitt és az alapismeretek megfelelő szintjéig lebontott – rendszer illetve konstrukciós elv, amely az adott rendszer által nyújtott szolgáltatások egészét könnyen érthetővé, áttekinthetővé és így elérhetővé tenné az orvos-felhasználó számára.

Ezen felismerésünk után elhatároztuk, hogy megkíséreljük a már jól bevált, magasszintű software rendszerek előnyeit egy következetesen végiggondolt és végigvitt elv realizálásával a kiértékelő orvos számára maximálisan kihasználni. Így készült el a SEGAMS, amely olyan hierarchikus (fa-struktúrájú) rendszer, amelynél az orvos az egész értékelési folyamatot a rendszerhez csatolt alfanumerikus display segítségével dialógus formájában irányítja. Alapvető célunk kitűzésünk egy orvosbarát, orvosbiztos, áttekinthető rendszer kifejlesztése volt. Ennek érdekében biztosítottuk, hogy a kezelő a munka minden fázisában lássa a rendszer által biztosított azon lehetőségeket, amelyek közül az adott pillanatban orvosi szempontból célszerű választani. Ezen lehetőségek csoportosíthatók, és a csoportok az orvosi értékelés stratégiájának és taktikájának megfelelően hierarchikusan rendezhetők, vagyis belőlük egy fa-struktúra alakítható ki.

A fa-struktúra csomópontjaiban ún. táblák találhatók. A táblák az alfanumerikus display-n kerülnek kiírásra és bemutatják azon lehetőségeket, amelyek közül az adott pillanatban a rendszer kezelője választhat. A lehetőségek elvileg két csoportra oszthatók:

1. végrehajtható (feldolgozó programok aktivizálása) (az ehhez esetleg szükséges paraméterek lekérdezése dialógussal)

2. a táblák hierarchikus rendjében különböző irányokban elvégezhető mozgások (ugrások) lebonyolítása.

A táblák táblaelemekből épülnek fel. Minden egyes táblaelem egy kétbetűs azonosítóból és a hozzárendelt funkció lényegét magyarázó szövegrészből áll. Valamely kétbetűs azonosító gépelésére a hozzárendelt funkció kerül végrehajtásra.

Egyszerű példaként tekintsünk egy adott kép élesítésére, feldolgozására szolgáló mintatáblát:

SW	gyenge símítás	SM	közepes símítás
SS	erős símítás	UC	uniformitáskorrekció
BG	háttér-levonás	PP	paraméterek printerre
DT	display tábla	RE	visszatérés az előző táblához

Az itt felsorolt lehetőségek közül az első három sor elemei egy-egy feldolgozó programot jelentenek. Ezek a háttérlevonás kivételével a felhívás – azaz a megfelelő két betű leütése után – automatikusan végrehajtódnak. A háttérlevonásnál a programnak paraméterekre van szüksége, ezért elvünknek megfelelően a végrehajtás előtt kérdések jelennek meg az alfanumerikus display-n: ezek megválaszolásával határozhatjuk meg, hogy alsó vagy felső levonást kívánunk-e végezni, a levonást %-ban vagy impulzusszámban kívánjuk-e végrehajtani, s végül a háttérlevonás mértékét. A DT azonosító aktiválásával egy hasonló szerkezetű, de a hierarchikus struktúrában alacsonyabb szinten elhelyezkedő táblát hívhatunk. Erről egy már feldolgozott kép különböző megjelenítésére szolgáló funkciókat aktivizálhatunk. Az RE funkcióval ahhoz a hierarchikusan magasabb szinten elhelyezkedő táblához jutunk vissza, amelyről fenti mintatáblánkat hívtuk.

Tapasztalataink alapján a kiértékelés könnyítésére az alfanumerikus display felső részén a kiértékelés teljes folyamata alatt célszerű megjeleníteni azon adatok összefoglaló táblázatát, amelyeket a felvételt megelőző dialógussal vettünk fel és amelyek méréstechnikai, felvételtechnikai stb. információkat szolgáltatnak. Az összefoglaló alatti hely szolgál a táblák megjelenítésére. Fontos kérdés a táblák méretének célszerű megválasztása. A méretre nézve az abszolut felső korlátot az alkalmazott alfanumerikus display fizikai tulajdonságai határozzák meg. Az általunk használt esetben pl. a displayre 16 sor írható, egy sor = 80 karakter. A display felső 8 sorát azonban a vizsgálat dialógusának összefoglaló bemutatására használjuk. A rendelkezésünkre álló hely tehát 8x80 karakter.

Ettől függetlenül a táblaméret megválasztásánál nyilván szem előtt kell tartani az egyes funkciók között fennálló orvoslogikai kapcsolatokat, továbbá azt, hogy az egyes tábláknak könnyen áttekinthetőknek kell lenniök. Ennek alapján mi úgy döntöttünk, hogy az egyes táblaelemeket legfeljebb két oszlopba (ezen belül egymás alá), és legfeljebb hét sorba írjuk (egy táblára tehát maximum 14 funkció kerülhet).

A táblákból felépíthető hierarchikus struktúra konstrukciójának ismertetése előtt vizsgáljuk meg azt, hogy milyen igényeket kellett software rendszerünknek kielégíteni:

a) a táblák flexibilisen felépíthetők legyenek

b) a táblák módosítása viszonylag gyors, egyszerű legyen

c) biztosítani kell a szövegek egyszerű kezelését, továbbá az egyik nyelvről a másikra való könnyű áttérést.

Ehhez:

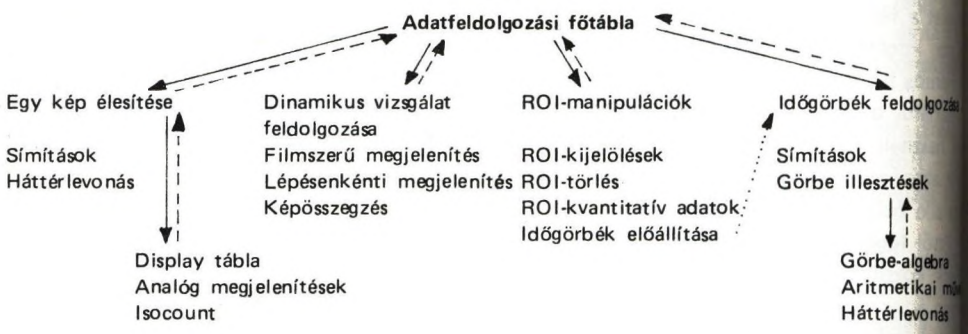
1. Olyan programkönyvtárat kellett előállítani, amelyből az ott szereplő program memória tetszőleges helyére töltve futtathatók. Ezt egy ún. relokálható könyvtár felép vel értük el. Ebben helyeztük el az alaprutinokat (amelyek nem végeznek önálló feldo tevékenységet, csupán más feldolgozó programok szubrutinjai) és a végrehajtható funk (vagyis magukat a feldolgozó programokat). A relokálható könyvtárban lévő programo tulajdonsággal rendelkeznek:

a) tetszőleges táblákba beépíthetők

b) az összetett kiértékelési folyamatokat automatikusan végrehajtó komplex prog alkotó elemeiként is működőképesek.

2. Szövegkönyvtárat kellett létrehozni, amely egyaránt alkalmas a rendszer strukt leíró táblaszövegek és a programokban előforduló szövegek egyszerű megadására, módo ra és kezelésére. A megvalósított szövegkönyvtárba beépített szövegek kétbetűs (belső) sítókkal címezhetők. A vezérlő program és az egyes funkciók a megfelelő kétbetűs azo megadásával hivatkoznak a kívánt szövegre, így programszinten csupán a szövegek neve tett. Rezidens szubrutin gondoskodik a kért szövegek megkereséséről. A keresés mindig móriában kezdődik, és csak akkor történik disc-hez fordulás, ha a kívánt szöveg nem t tó meg a központi memóriában. A választott megoldás igen nagy előnye, hogy a teljes szer tetszőleges nyelvű változata csupán a szövegkönyvtár lefordításával könnyen előállí Egy angol nyelvű rendszerből ily módon néhány óra alatt elkészíthető pl. annak franci vű változata.

Egy konkrét képkiértékelési struktúra alaphelyzetben több feldolgozási irányt eng Ezek számát az adott rendszer felhasználási célja és az orvosi kiértékelő logika szabálya meg. Mivel ez az ún. adatkiértékelő alaptábla is egy, az előzőekben definiált táblát jele benne elhelyezhető elemek száma szintén korlátos, esetünkben maximum 14 lehet. Az tábláról aktivált feldolgozási irányok táblái már végrehajtható funkciókat ill. további ér si irányokat tartalmaznak. (ld. az 1. ábrát) A táblák közötti egyszerű közlekedés biztos fontos kívánalom. Erre rendszerünkben 3 különböző módot biztosítottunk:



1. ábra Egy adatfeldolgozó rendszer egy részfájának bemutatása

Megjegyzés: A ——→ a fa-struktúrában való (szabályos) előrehaladást,
a - - - - - → a visszatérést,
a ..... → pedig az egyik ágról a másikra való ugrást jelöli.

1. *előrehaladás,* valamely tábláról újabb, alacsonyabb szinten elhelyezett táblák hí

2. *visszatérés,* valamely tábláról az előző, magasabb szinten elhelyezett tábla vissz sa. Ezt a minden tábla utolsó sorában szereplő „visszatérés a megelőző táblához" valósí meg.

3. szabálytalan ugrások, amelyek lehetővé teszik a struktúra egyik ágáról egy másik ágára orvoslogikailag indokolt ugrás elvégzését. A szabálytalan ugrás végrehajtható automatikusan vagy a táblán szereplő megfelelő funkció kiválasztásával. Automatikus ugrást végzünk pl. a ROI táblán elhelyezett, az időgörbék konstruálását végrehajtó program lefutása után. Mivel feltehetjük, hogy a kiértékelés következő lépése a konstruált időgörbék feldolgozása lesz, az időgörbék elkészülte után nem a ROI-táblát, hanem automatikusan az időgörbék feldolgozását biztosító táblát kapjuk (ezt a példát ugyancsak szemléltetjük az 1. ábrán).

A táblák összeállításakor döntő szempont a funkciók csoportosítása.

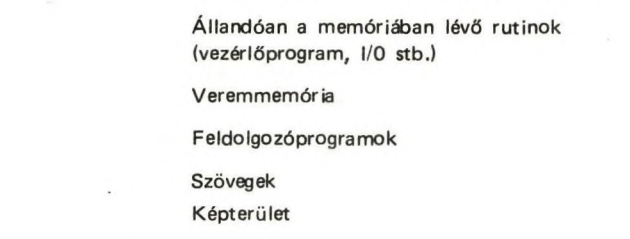
Általános irányelv, hogy egy táblára lehetőleg orvosilag egy feldolgozási célt szolgáló rutinok kerüljenek (pl. egy adott kép élesítése; display eljárások; ROI manipulációk; list módú adatok feldolgozása stb.).

A definiált fa-struktúra és táblák, továbbá a korábbiakban már ismertetett relokálható programok és szövegkönyvtár felhasználásával a teljes rendszer előállítását egy ún. generátor program végzi el. Működési elvét a 2. ábra szemlélteti.

2. ábra A generálás működési elve

A szövegkönyvtárban a táblaszövegeken és a programokban előforduló szövegeken kívül minden egyes táblához az adott tábláról hívható funkciók listája is helyet kapott. Ez a lista központi szerepet játszik a rendszer generálásakor és a kész rendszer működése közben. A lista kétbetűs azonosítópárokból áll. A párok első tagja a funkciónak a display-re kiírt táblán látható neve, a második tagja a funkciót megvalósító program neve a relokálható programkönyvtárban. Ez a megoldás biztosítja, hogy a rendszer használója minden funkcióra a funkció lényegét magyarázó szöveg természetes rövidítésének gépelésével hivatkozhasson, továbbá azt is, hogy a rendszer struktúrájának különböző pontjain az orvosilag azonosnak tekinthető funkciók azonos nevet kaphassanak még akkor is, ha a hozzájuk tartozó programok különböznek egymástól. A generátor program működése során a hívható funkciók listáiban szereplő programokat betölti a relokálható programkönyvtárból és összeszerkeszti az illető programokból hívott szubrutinokkal. Az így összeszerkesztett programokból felépíti a rendszerkönyvtárat, a funkciólistákat pedig kiegészíti a funkciók rendszerkönyvtárbeli eléréséhez szükséges információkkal. (disc, memóriacím)

Az elkészített rendszer memóriabeosztását a 3. ábra szemlélteti.



3. ábra A memória beosztása a feldolgozás során

A kész rendszer működését a vezérlőprogram irányítja. A központi memóriában egy verem van kialakítva. A verem elemei táblaelemek. A verem tartalma egyértelműen meghatározza, hogy a feldolgozás a rendszer struktúrájának mely pontján tart. A vezérlőprogram a verem tetején megadott név alapján kikeresi a hierarchia adott szintjéhez tartozó tábla szövegét, kiírja az alfanumerikus display-re.

Az orvos kétbetűs azonosító gépelésével választ a felsorolt lehetőségek közül. A vezérlőprogram kikeresi a hívható funkciók listáját és ellenőrzi, hogy a begépelt azonosító szerepel-e a listán. Amennyiben nem szerepel, a vezérlő visszatér a tábla szövegének kiírásához. Ha szerepel, akkor a generátor által elhelyezett információk alapján (szükség esetén a rendszerkönyvtár megfelelő részének betöltése után) szubrutinként felhívja a kért funkciót megvalósító programot. Amennyiben az adott programnak nincs szüksége paraméterekre, akkor végrehajtás után visszaadja a vezérlést a vezérlőprogramnak. Amennyiben paraméterre van szükség, úgy kérdés következik, amelyben az alfanumerikus display-n megjelenik a kérdezett paraméter neve, és ha ilyen létezik, a helyes válasz felső és alsó határa. Ezen segítséggel az értékelő folyamatosan követni tudja válaszait. Amennyiben ennek ellenére tévedne, úgy a kérdés megismétlődik. A paraméterek begyűjtése után az adott program végrehajtódik és a vezérlővel folytatódik a végrehajtás. A kérdések feltevése előtt a vezérlő ellenőrzi, hogy az adott azonosítójú szöveg a memóriában van-e. Amennyiben nincs, úgy a szövegkatalógusból kikeresi a megfelelő mágneslemez-címet, betölti a szöveget a memóriába, majd megtörténik a szöveg kiírása.

Ha a hívott funkció a fa-struktúrában az alacsonyabb szinten elhelyezkedő táblára való áttérést jelenti, akkor ennek a táblának a neve kerül a verem tetejére. Az RE funkció egy visszahúzza a veremmutatót, s ezzel biztosítja az előző táblára való visszatérést.

Az előbbiekben vázolt konstrukció lehetővé teszi, hogy a relokálható könyvtárban a programokból tetszőleges rendszereket állítsunk elő. Az izotópdiagnosztikai laboratóriumban egy gammakamerához kapcsolt kisszámítógép-rendszernek általában sokrétű feladatot megoldania. A gyakorlatban az történik, hogy a speciális diagnosztikai témákkal foglalkozó kutatók és orvosok a legáltalánosabb, legtöbb funkciót tartalmazó fix rendszert is korlátozónak, szűkösnek érzik.

Ezeken a helyeken általában a diagnosztikai céloknak megfelelően további funkciók célirányos komplex programok alkalmazására, fejlesztésére kényszerülnek. Ezt a lehetőséget természetesen biztosítani kell. Elvi megoldásunk lehetőséget teremt arra, hogy ilyen speciális igények nagy részét már eleve kielégítő és ebből a szempontból felesleges feldolgozási lehetőségektől mentes célrendszereket generáljunk a fa-struktúra felhasználásával. Ebben az esetben a már kifejlesztett funkciók egy része egyáltalán nem kerül egy ilyen célrendszer táblájába, éppen mint egy komplex program obligát része, indirekt módon kerül felhasználásra.

Az eddig elmondott elvek természetesen tetszőleges konfigurációjú számítógépen megvalósíthatók. A realizációt nagy mértékben megkönnyíti, ha a vezérlést alfanumerikus display-n végezhetjük.

Végezetül a fa-struktúra felhasználásának általánosítási lehetőségére szeretnénk felhívni a figyelmet. Az eddigiekben csupán a vizsgálatok kiértékeléséről és az itt használt fa-struktúráról volt szó. Valójában az említett struktúrát a rendszer működése során mindvégig felhasználjuk. Bekapcsoláskor pl. az alfanumerikus display-n az előzőekkel teljesen megegyező felépítésű táblázat jelenik meg, amely a rendszer alaptevékenységeit írja le. (adatfeldolgozás, -felvétel, tesztelés, off-line programozás stb.) és a résztevékenységeket minden lehetséges helyen, a korábbiaknak megfelelően struktúráljuk.

Befejezésül megjegyezzük, hogy a fenti elvek szerint megvalósított SEGAMS rendszerből jelenleg kb. 20 működik, jelentősen könnyítve a kiértékelés munkáját.

EGY LOGIKAI ALAPÚ KÉMIAI INFORMÁCIÓKEZELŐ RENDSZER ELMÉLETI MEGFONTOLÁSOK ÉS GYAKORLATI TAPASZTALATOK

Darvas Ferenc–Szeredi János–Futó Iván–Rédei János
SZKI

1. Bevezetés

A kémiai struktúrák számítógépes kezelése kevéssé ismert, de nagy műszaki-gazdasági jelentőségű terület, ahol a problémák jelentős része még megoldatlan.

A kémiai struktúrák számítógépes kezelésének alapproblémája, hogy a matematikailag nyítatlan gráfoknak megfelelő struktúrák csak nehézkesen tárolhatók és kezelhetők a szokás adatfeldolgozási módszerekkel [1]. Éppen ezért terjedt el néhány, a vegyületek lineáris tárol lehetővé tevő rendszer (Wiswesser, Gremas stb.) [2]. E rendszerek hátránya, hogy a bennük dolt vegyületek nem kereshetők vissza tetszés szerinti szerkezeti részlet (szubstruktúra) alapj Tetszés szerinti szubstruktúra-visszakeresést biztosító rendszert – a szakirodalom szerint – cs néhányat közöltek [3].

A PROLOG nyelv felépítésénél fogva különösen alkalmas gráfkezelési problémák megol sára, továbbá a kémiai szerkezeten alapuló, de a kémiai és biológiai tulajdonságokat is tartalm automatikus következtetési rendszerek kiegészítésére. FORTRAN nyelvű statisztikai program tartalmazó programrendszerünkről az elmúlt évben számoltunk be [4]. Úgyszintén a közelmúl ban vált ismeretessé az amerikai kidolgozású SYNCHEM rendszer, amely automatikus következ tetés felhasználásával szintén kémiai információkezelési problémát – szintézistervezést – old meg [5].

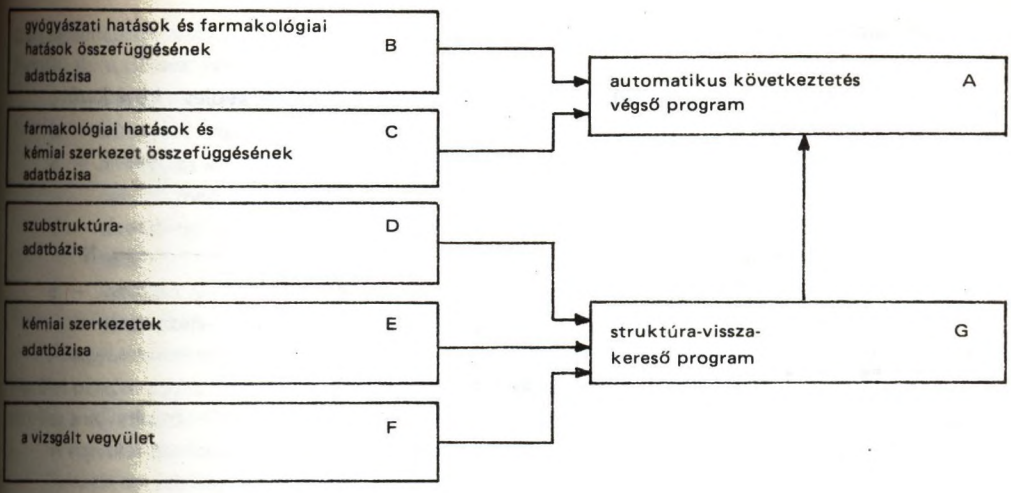
A továbbiakban az SZKI-ban általunk kidolgozott kémiai információkezelő rendszert fo juk ismertetni.

2. A rendszer célkitűzése

A rendszer célja gyógyszerkutatási munkában történő segítségnyújtás. A közvetlen cél a hogy az adatbázisban tárolt vegyületek új farmakológiai és gyógyászati felhasználását követ keztessük a vegyületek kémiai szerkezete, továbbá a kémiai szerkezet és a farmakológiai hatás illetve a farmakológiai hatások és a gyógyászati alkalmazások összefüggése alapján.

3. A rendszer felépítése

A rendszer felépítésének vázlatát az 1. ábra tartalmazza. A rendszer vezérlését az autom tikus következtetést végző program (A) biztosítja. A program a gyógyászati hatások és a farma kológiai hatások közötti összefüggések (B), és a farmakológiai hatások és a kémiai szerkezet k zötti összefüggések (C) alapján következtet a vegyületek (F) új farmakológiai és gyógyászati h tásaira.



1. ábra

A rendszer logikai vázlata. A nyilak az információáramlást jelzik.

A farmokológiai hatások és a kémiai szerkezet között természetesen nagyon sokrétű az összefüggés. Programrendszerünk csak azokat az összefüggéseket tartalmazza, amelyek a vegyületekben levő kémiai szerkezeti részletek (a továbbiakban: szubstruktúrák) és a farmakológiai hatások között egyértelműen fennállnak. A farmakológiai hatás és a kémiai szerkezet közötti összefüggésekben a szubstruktúrák megnevezése szerepel. A szubstruktúrák megnevezése és a szubstruktúrák által reprezentált kémiai szerkezet között a szubstruktúra-adatbázis (D) teremt kapcsolatot. A szóban forgó kémiai szerkezeteket (E) a vizsgálat alá vont vegyületekben egy külön program keresi vissza (G). Ezt a programot az automatikus következtetést végző program hívja meg abban az esetben, ha a következtetési lánc folytatása egy-egy szubstruktúra meglétének bizonyítását igényli.;

4. Tudásreprezentáció. A következtetési eljárás

A rendszerben az alábbi hierarchikusan egymásra épülő tudásszintek kapnak helyet.

1. Gyógyászati hatás.
2. Farmakológiai hatás.
3. Kémiai tulajdonság (=szubstruktúra).
4. A kémiai szerkezet.

A rendszerben a tudásreprezentációt olyan módon szerveztük meg, hogy a felsőbb szinteken szereplő fogalmakat mindig az alacsonyabb szint fogalmaival definiáltuk. Például:

A gyógyászati hatás szintje:

* Azok a vegyületek, amelyek görcsgátló farmakológiai teszteken hatásosak, rendszerint alkalmazhatóakcentrencephalikuseredetű „petit mal" epilepszia kezelésére.

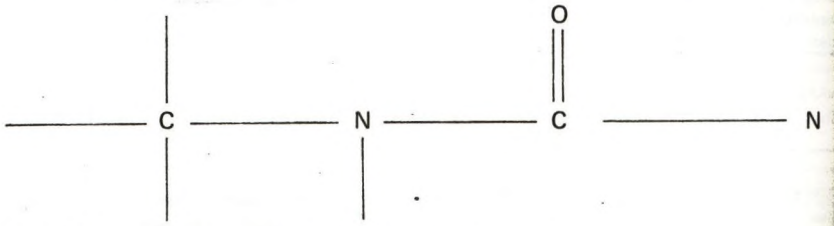
A farmakológiai hatás szintje:

** A pentiléntetrazol-görcsgátlás elnevezésű teszt görcsgátló farmakológiai teszt.

*** Azok a vegyületek, amelyek szubsztituált fenilkarbamid-struktúrát tartalmaznak, rendszerint hatásosak a pentiléntetrazol-görcsgátló teszten, feltéve, hogy nem tartalmaznak hidrofil szubstruktúrákat.

A kémiai tulajdonság szintje:

A fenilkarbamid-szubstruktúrának megfelelő kémiai szerkezeti képlet a következő:

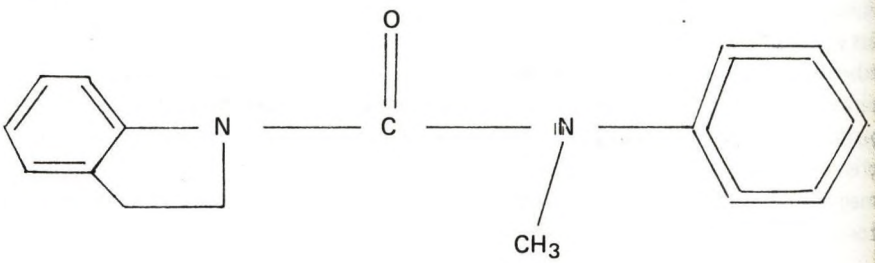


* ** *** állítások formalizált alakja a következő:

A hidrofil szubstruktúrák a következők:
- CH
- COOH
- NH_2 , ... stb.

A kémiai szerkezet szintje:

Ezen a szinten konkrét vegyületek találhatók, pl. az alábbi tartalmazza a keresett szubstruktúrát:



A négy tudásrétegben szereplő ismeretanyag egységesen Horn-klózokban van kifejezve, így ezeket a PROLOG interpretere közvetlenül kezelni tudja. A farmakológiai és a gyógyászati hatásra vonatkozó ismereteinket a következő formában tároltuk:

```
*     +gyógyszere (*vegyület, petit-mal)
            -farm_hat (*vegyület, görcsgátló).

**    +farm_hat (*vegyület, görcsgátló)
            -farm_hat (*vegyület, pentiléntetrazol-görcsgátló).

***   +farm_hat (*vegyület, pentiléntetrazol-görcsgátló)
            -tartalmaz (*vegyület, fenilkarbamid)
            -not [hidrofil (**) and tartalmaz (*vegyület, **) ].
```

Ez a forma hasonlít a korábban gyógyszerinterakcióra vonatkozólag kidolgozott tudásreprezentációhoz [6], továbbá a terápiás javallatokat szolgáltató MYCIN rendszer [7] tudásreprezentációjához. A programrendszerben egy saját fejlesztésű vegyületábrázolást és szubstruktúra-rend

szert alkalmazunk, amely a következő előnyökkel rendelkezik a vegyületek szokásos mátrix formájú ábrázolásával szemben:

a – a vegyületszerkezet finomabb ábrázolását teszi lehetővé,

b – könnyebb visszakeresési lehetőséget biztosít,

c – a vegyület szimmetria tulajdonságai, amelyek a mátrixba történő leképzéskor eltűnnek, itt megmaradnak,

d – tárolási helymegtakarítással jár, mivel nincs szükség a mátrix nulla elemeinek tárolására,

e – azokban az esetekben, amikor a mátrix-típusú ábrázolás előnyösebb, a klóz-formában tárolt szerkezet könnyen átalakítható mátrixformára.

A vegyületszerkezet Horn-klózok formájában történő ábrázolásának az a lényege, hogy a vegyület minden egyes kötését külön-külön Horn-klóznak tekintjük. Egy kötés szabványos ábrázolása a következő: +OS (*atomszám1, *atomfajta1, *kötésfajta, *atomszám2, *atomfajta2).

A vegyület ábrázolása a vegyületben szereplő kötések felsorolását jelenti.

A kötés ábrázolása tehát egy öt argumentumú relációnak felel meg. Megadjuk a kötésben szereplő mindkét atom sorszámát és fajtáját (1., 2. és 4., 5. argumentum) továbbá a kötés fajtáját (3. argumentum). A változó megnevezése előtt szereplő csillag a PROLOG nyelv konvenciója szerint arra utal, hogy az argumentumban tetszés szerinti értékű, ill. jelentésű változó szerepel. Ez programozástechnikai szempontból jelentős előny: a mátrix-típusú ábrázolásban a logikai formában leírható minőségi különbségeket (szénatom, nitrogénatom stb.) csak úgy kezelhetjük, hogy a minőségi különbségeket (szénatom, nitrogénatom stb.) csak úgy kezelhetjük, hogy a minőségi jellemzőket előzetesen számmá konvertáljuk valamilyen megegyezés szerint. Egységes konverziós formula azonban elvi alapon nem adható meg, gyakorlati szempontok alapján pedig a konverziós formulák túl nagy sokasága képzelhető el.

Az ábrázolás másik közvetlen előnye a nagymértékű függetlenség a vegyületek felírásmódjától. A vegyületek jelentős részében az atomok között kötés sem egyes kötésnek, sem kettős kötésnek, hanem a két kötésfajta közötti átmenetnek felel meg („delokalizált kötések"). Ezeket a delokalizált kötéseket tartalmazó vegyületek többféleképpen írhatjuk fel, a lehetséges felírásmódok száma 2^n, ahol n az egymástól független delokalizált kötésrendszerek száma.

A mátrix-típusú ábrázolásoknál a kötésfelírás egységesítésére kényelmetlen, elvileg megalapozatlan és a gyakorlatban nehezen ellenőrizhető kikötéseket tesznek. A logikai alapú ábrázolásban ezzel szemben a delokalizált kötéseket egységesen egy külön erre a célra definiált „delokalizált" kötésértékre konvertáljuk, és a molekulát ezekkel a kötésértékekkel tároljuk.

5. A rendszer megvalósítása. Tapasztalatok a rendszer alkalmazásával kapcsolatban

A rendszert az SZKI Siemens 4004 számítógépén, a Köves Péter és munkatársai által adaptált interaktív PROLOG interpreter segítségével valósítottuk meg. A rendszer jelenleg kísérleti stádiumban van, teljes kiépülése a következő esztendőben várható.

A gyógyászati hatásokra és a farmakológiai hatásokra vonatkozó adatbázisok jelenleg 100 következtetési szabályt tartalmaznak, ezek bővítését tervezzük. A szubstruktúra-visszakereső program és az automatikus következtetést végző program működik, de a két program összekapcsolását csak 1980-ra tervezzük.

Irodalom

[1] W. T. Wipke, S. R. Heller, R. J. Feldmann. E. Hyde;
Compute Manipulation and Representation of Chemical Information. Wiley, N. Y., 1974

[2] E. G. Smith; The Wiswesser Line-Formula Chemical Notation, McGraw-Hill, N. Y. 1968.

[3] F. G. Stockton. R. L. Merrit; J. Chem. Doc. **14** 166 (1974)

[4] Darvas Ferenc, Futó Iván, Szeredi János, Bendl Judit, Köves Péter; PROLOG alapú gyógyszertervezési programrendszer.
Előadás az NJSZT által rendezett „Programozási Rendszerek '78" c. konferencián, Szeged 1978. november.

[5] H. L. Gelenter, A. F. Sanders, D. L. Larsen; Science, **197** 1041 (1977)

[6] Darvas F., Futó I., Szeredi P., Program gyógyszerkölcsönhatások visszakeresésére és új gyógyszerkölcsönhatások megtalálására mechanikus következtetés segítségével.
„Számítógép az Orvostudományban" c. szimpózium kiadványa, Szeged, 1976. 413–422 (Szerk. Muszka Dániel)

[7] E. H. Shortliffe, R. Davis, S. G. Axline, B. G. Buchanan, C. C. Green, S. N. Cohen; Computer-Based Consultations in Clinical Therapeutics Explanation and Rule Acqusition Capabilities of the MYCIN System. Computers and Biomedical Research **8** 303–320 (1975)

[8] D. H. Rouvray; Chemistry, **45** 6 (1972)

– – – – –

EGY SZÁMÍTÓGÉP–LEÍRÁSI MÓDSZER FŐBB VONÁSAI

Dávid Gábor
MTA SZTAKI

1. A számítógép-leírás lehetősége

A számítógépek architekturájának – azaz komponenseinek és azok működésének – leírása napjainkban a kutatások középpontjába került. A probléma nem újkeletű, mégis az architektura leírását célzó, sikerrel befejeződött kísérletek száma nagyon csekély és csak egy-egy részterületen vezettek eredményre. Ennek a „részleges sikertelenségnek" az oka ma már nyilvánvaló: hiányzik egy egységes tárgyalásmód, azaz jól definiált fogalmak egy rendszere, melynek alapján az architektura minden szinten – a hardware-től az alkalmazási feladatokig terjedően – tárgyalható lenne. Az analógiák ezen szintek működési elvei között nehezen lelhetők fel és a formalizálhatóságot nehezíti, hogy egyediségükben, konkrétságukban sokféle tulajdonság egyidejű realizálásának bonyolultságában találkozunk velük. A nyilvánvaló elvekre – így pl. a tárolás, vezérlés, belső állapotátmenetek elveire – támaszkodva azonban gyakorlati szakember számára áttekinthetetlen, használhatatlan leírási módszerekhez jutunk el.

A probléma – majdani – megoldásához azonban egyre közelebb kerülhetünk, ha a technológiai fejlődés által lehetővé váló új elveket következetesen végiggondoljuk. Itt most csak utalnánk arra, hogy „egyre mélyebb szintre hatol be a software, mondhatni, hogy a hardware software-esedik", ugyanakkor a software egyre inkább él azokkal a lehetőségekkel, hogy az egyes software jellegű komponenseket a software szempontjából hardware úton valósíthatunk meg. Az architektúra szintjei közötti határ elmosódik és ezzel a gyakorlat bizonyítja az architektura különböző szintjei közötti felépítési és működési elvek azonosságát, kicserélhetőségét, vagy kiválthatóságát.

2. A számítógép-leírás célja

Minden kísérlet a számítógép-leírásra a célját leszűkítette. Ilyen cél lehet a működési elvek ismertetése, dokumentáció, különböző gépek összehasonlítása, egyes komponenseknek a méretezése, szimulálás, verifikálás, tervezés. Újabban a működési elvek kutatására is dolgoztak ki leíró nyelveket (átstruktúrálhatóság, vezérlési struktúrák nyelve stb.).

Az itt ismertetésre kerülő leírási módszer megkísérli tárgyát és célját általánosabban megfogalmazni:

– rendszerek *működésének és struktúrájának* leírására tesz kísérletet úgy, hogy
– a rendszer *fejlesztésének* minden fázisában eszközt akar adni a tervező kezébe.

A *rendszer-leírásnál* azt tesszük fel, hogy minden rendszer egy, vagy több komponensből áll, amelyekből struktúrát felépítve létrehozunk egy ellátandó funkciót realizáló új objektumot, ami lehet újra komponense más rendszernek.

A *rendszerfejlesztés* során a legfontosabb lépések
– a probléma-megfogalmazás
– megvalósítás
– ellenőrzés
és ezek ismételt alkalmazásához a dekomponáláson keresztül a „top-down" megközelítéskor, vagy szintetizáláson keresztül a „bottom-up" stratégia esetén.

A probléma megfogalmazását és a megvalósítását szétválasztjuk a „mit és hogyan ol meg" kérdéseknek megfelelően. Itt a probléma-megfogalmazását *(funkcionális) specifikáció* a megvalósítást *implementációnak* hívjuk:

— *a funkcionális specifikáció* = mit akarunk megoldani?

— *implementáció* = hogyan akarjuk megoldani?

A két részt szeretnénk különválasztani. Ennek oka az, hogy a rendszerfejlesztés sor egyes fázisokban nem egyenletes a két rész szerepének fontossága — így a top-down strat esetén kezdetben a hangsúly a funkcionális specifikáción van (rész-problémákra való bont azok feladatainak leírása stb.), míg a fejlesztés végső szakaszában az implementáció kap n szerepet (hardware realizálás, programozás stb.).

A szintektől többé-kevésbé független a harmadik összetevő, az *ellenőrzés.* Az ellenőr az egyes szintek implementációjának az összevetése a specifikációval és — mivel az egész r szer egyúttal a valóságos feladatimplementációjának tekinthető — a rendszer funkcionálás az ellenőrzése. Módszerei azonban szintenként különbözőek. Tipikus ellenőrzési módszer a szimuláció, vagy modellezés; nem tipikus még, de ígéretes módszer a helyesség-bizonyítás.

3. A frame fogalma

A leírás módszer elemi egysége a *frame,* keret. Egy framenek három fő komponense

IS: interface-specifikáció

FS: funkcionális specifikáció

IM: implementálás

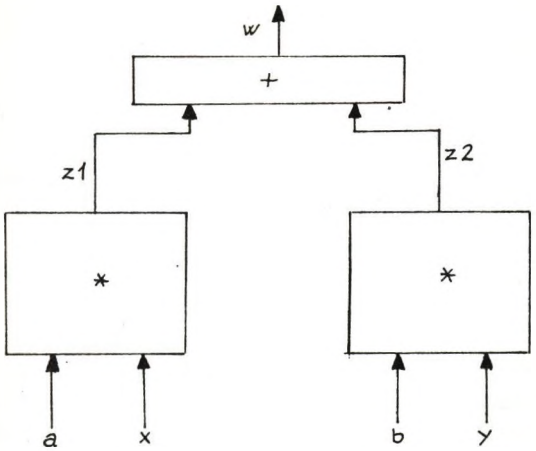
A rendszer-készítés során a felhasználó a frame-t különböző mélységig dolgozhatja k a frame ezen három komponense közül vagy az implementáció, vagy a funkcionális specifi ció is hiányozhat. A frame interface-specifikációja, az IS azonban mindig kell. IS-ben kell mondani a felhasználónak, hogy hogyan hivatkoznak a frame-ra. Tartalmazza a frame nev paramétereit:

frame frame-név (IPL; OPL)

paraméter deklarációk;

ahol IPL, OPL rendre input, ill. output paraméter-lista, (lista: szimbólumok vesszővel elvá tott sorozata). A paraméter-deklarációk specifikálják a paraméter-listában szereplő szimbólu kat (lásd később részletesebben). Az interface-specifikáció az input/output specifikációt íg de csak forma szempontjából.

A frame szerepe hasonló a data-flow architektúra [1] elemi összetevőihez. A data-flo nyelvekben adott funkciók egy halmaza, amelyek mindegyikének meghatározott inputja é outputja van. Ha egyik funkció outputjára egy másik inputként hivatkozik, akkor az infor ció ebbe az irányba továbbítódik. Minden funkció akkor hajtódik végre, amikor az összes inputja megérkezik. Ha van a rendszerben + és * (összeadás és szorzás), akkor az ax + by téket a következő architektura valósítja meg:



vagy leírva

* (a, x; z1)
* (b, y; z2)
+ (z1, z2; w)

(látható, hogy a leírás sorrendje nem lényeges!)

Ettől a leírási módszertől több szempontból eltérünk:

– a frame nem egy funkció, hanem egy funkció-osztályt ír le (és ennek egy-egy példányát generálhatjuk),

– a frame nem egy objektum, hanem ugyanolyan objektumok egy osztálya (és ennek egy-egy példányát generálhatjuk).

– a vezérlés nem automatikus, hanem az implementációs részen belül kell leírni a vezérlési struktúrát.

Ezeknek az eltéréseknek az oka az, hogy egyrészt a frame olyan objektumok összességét írja le, amelyek az adott funkciót hasonló módon látják el, s a felhasználásuk során (megfelelő aktuális paraméterekkel ellátva) ilyen objektumokat létre kell hozni, azaz konkrét frame-eket kell deklarálni; másik oldalról pedig az indokolja az eltérést, hogy a vezérlésnek a data-flow gépekben való implicit megvalósítása helyett a gyakorlatban inkább szükség van a vezérlésnek explicit megadására, azaz az egyes komponensek akkor is végrehajthatják feladataikat, ha nem minden input értéket kapott még meg a komponens. A data-flow eredeti modelljében egy komponens akkor és csak akkor aktivizálódik, mikor minden input értékét megkapta.

4. Két nyelv: specifikációs és inplementációs nyelv

Miután a frame-ben egyaránt kell szólnunk a frame által végrehajtott funkcióról és a kérdéses funkció realizálásáról, ezért ezen „két-arcúságot" elfogadva „két-nyelvűség" megvalósítására törekszünk.

A két nyelv közül a *specifikációs nyelv* elsősorban *leíró jellegű,* mivel a rendszerkészítés során különböző célokra (szimulálás, dokumentálás stb.) egyaránt felhasználható kell, hogy legyen. Az *implementációs nyelv* azt a programot írja le, ami realizálja a specifikációs nyelven megfogalmazott funkciót, így *algoritmikus* nyelv.

A specifikációs nyelv szerepe az, hogy a frame felhasználásakor az implementáció helyett írja le – általában magasabb szintű nyelven – a frame viselkedését. Így elegendő információt kell nyújtania

– dokumentáláshoz,

– szimuláláshoz,
– különböző verifikációs módszerekhez,
– tervezéshez.

Ugyanakkor az implementációs rész tartalmát, *értelmét* is el kell tudni mondania, így erőteljes *szemantika leíró* jelleggel kell bírnia.

A lehetséges nyelvek között a matematika nyelvei (rekurzív függvények, logika nyelve így PROLOG, vagy SL) jöhetnek számba, vagy egy „széleskörűen" elfogadott, értett, így implicit módon szemantikával ellátott programozási nyelv (SIMULA'67).

Az implementáció nyelvének a szerepe teljesen lokális a frame-re nézve. (Más kérdés, hogy egy rendszer kifejlesztése során célszerű egyazon nyelvet használni bizonyos fázisokban a rendszer-készítők kommunikációja érdekében.) Miután az implementáció szerepét a specifikációs rész teljesen leárnyékolja, ezért csak a frame-en belüli szerepe fontos: tudnunk kell verifikálni a két rész azonosságát: az implementáció megfelel-e a specifikációnak.

Ez egyúttal azt is jelenti, hogy az implementáció nyelve nem egyetlen nyelv: lehet magasszintű programozási nyelv, vagy assembly-nyelv, logikai nyelv, vagy hardware; és egy frame élete során az implementációs nyelv változhat. Fontos követelmény azonban, hogy az implementációs nyelvekben

– lehessen már leírt frame-eket aktivizálni,
– dinamikusan változó vezérlési struktúrákat lehessen leírni.

Az implementációs nyelvnek egyszerre kell *végrehajtható és interpretálható* nyelvnek lennie. A végrehajthatóság követelménye nyilvánvaló: a rendszer teljes realizálásakor csak az implementációs részek „maradnak" meg, és a rendszernek működnie kell. Az interpretálhatósághoz kell, hogy az ellenőrzés során különböző verifikációs, tesztelési módszereket alkalmazhassunk.

A „két-nyelvűség"-hez szeretnénk még visszatérni és két megjegyzést tenni:

– abban hiszünk, hogy a „két-nyelvűség" célszerűbb. Pontosabban: az elképzelhető, hogy a fenti követelményeket egyetlen nyelv is ki tudja elégíteni (ALPHARD), azaz az „egy-nyelvűség" lehetséges, de a fentiek miatt tudjuk, hogy nem szükséges.

– a „két nyelv" végülis „több nyelv". Ha adva van specifikációs és implementációs nyelvek egy-egy halmaza, és az egyes implementációs nyelveknek meg van adva az értelmük specifikációs nyelvben, akkor egy frame leírásánál bármely ilyen nyelvpárost használhatunk.

S ugyanakkor van egy harmadik nyelv is: a frame-leírás nyelve. (Ez azonban egyetlen nyelv.) Ez adja a frame-ek interface-specifikációját, nyelvi „keretet" ad a specifikációs és implementációs résznek és lehetővé teszi kontrol-struktúrák leírását. Ezeket egy speciális esetet illusztráljuk (részletes leírás megjelenik az MTA SZTAKI kiadásában) [2]

5. Egy megvalósítás: Architecture Language AL

Az AL-ben az **alap**vető adattípusok a bit(n), byte(n), amelyek n hosszúságú bit, vagy byte-szervezésű információ (tároló, sín stb.) reprezentálható. Ezek tartalma mindig *egész*. Ezeket az interface-specifikációban használhatjuk.

Az AL specifikációs nyelve

A *specifikációs részben* a következő nyelvi konstrukciókat engedjük meg:

begin ... end;
if ... then ... else ...;
while ... do ... od;

értékadások.

A kifejezések az elemi matematikai függvények kompozíciói. Bonyolultabb függvények önállóan leírhatók a *function* alapszóval,

function függvény-név (formális paraméterek) paraméter-specifikáció;
begin; törzs *end;*

Minden szimbólumhoz deklarálhatunk attributumokat, amelyek típusa lehet integer, real, boolean. Az attributumok deklarálása

típus-deklatátot szimbólum.attributum-név *begin* törzs *end;*

formájú, például

real A.CI;

vagy *boolean* A.event 1
begin
A.eventI: = if(A.CI=0.1) *then* false *else* true;
end;

Minden deklarált szimbólumhoz automatikusan egy *idő attributumot* asszociálunk, amelyre

szimbólum.time

formában hivatkozunk. A frame-nek magának is van ideje, amelyre a specifikációs részen belül és kívül is hivatkozhatunk és belül értéket adhatunk. A frame ideje a vezérlés megkapásakor 0, idő-konstansok ehhez igazodnak és kilépéskor az adott érték. Az egyéb szimbólumok időattributumát a frame idejéhez viszonyítjuk. Az időintervallumokban fennálló állapotokra vonatkozó predikátumokat a predikátum után írt

[időkifejezés1, időkifejezés2]

(időkifejezés1 $\geqslant$ időkifejezés2) érvényességi specifikáció fejezi ki. Időkifejezés bármely kifejezés. Például

if (C $\geqslant$ 50) [−20, A.time+2] ...

ahol −20 (idő)konstans, a frame aktivizálásának időpillanata előtti 20 egységet jelöl, míg A. time az A szimbólumhoz rendelt időértéket jelöli. Azt, hogy $-20 \leqslant$ A.time +?, ellenőrizni kell és a C $\geqslant$ 50 logikai értéke false, ha nem áll fent az érvényesítési specifikációra ez a megkötés. Ha nincs érvényesítési specifikáció, akkor mindig a szimbólumok idejének aktuális értéke a frame idejével egyenlő. Minden szimbólum, amelyet deklarálunk, voltaképpen struktúra is, lévén bit(n),vagy byte(n). Összetett struktúrákat is létrehozhatunk, itt csak a vektorokat mutatjuk be:

„struktúra-név" „részstruktúra-név" [n_1; n_2]

ahol struktúra-név tetszőleges, a „részstruktúra-név" már deklarált szimbólum, n_1 és n_2 ($n_1 \leqslant n_2$) a vektor határait jelölik. Az így kapott struktúra elemeire

struktúra-név [index]

formában hivatkozhatunk, amelyek típusa már egyezik a „részstruktúra-név" típusával. Típusegyezés nemcsak a deklarációs szintre vonatkozik, hanem a már definiált attributumokra is, azaz a „részstruktúra-név" attributumait az új struktúra elemei örökölhetik, de újakat is lehet deklarálni a „struktúra-név" szimbólumhoz.

6. Az AL implementációs nyelve

AL implementációs részének általános felépítése
deklarációk
cimkézett modulok
vezérlési algoritmus
vezérlés-aktivizálás

A vezérlési struktúrák leírása a Petri-hálókon alapul, itt most csak hivatkozunk azokat részletesen ismertető cikkekre (On the Basic Concepts of a Module Language, MTA SZTAKI

Közlemények, vagy Description of Dynamic Control Structures, ALGORITHMS'79, Csehszlovákia). A vezérlési algoritmus az implementációs részben leírt cimkézett modulokat „felfűz" egy, vagy több vezérlési struktúrára, amit a vezérlés-kijelöléskor meghatározunk és végrehajtunk.

AL-ben a deklarációk az implementációs részben egyrészt a vezérlési struktúrával kapcsolatosak, másrészt a specifikációs részben leírt deklaráció kiegészítése lehet, azaz megfelel a fentiek szerinti szabályoknak. Fontos azonban, hogy az implementációs részben frame-ket használunk. Frame-ek példányait generálhatjuk a

frame frame-név (aktuális nevek listája);

formában, ahol az aktuális nevek listája szimbólumok egy listája, mindegyike a „frame-névvel specifikált frame egy-egy példányát reprezentálja.

A cimkézett modul voltaképpen vezérlési struktúra nélküli „program"

cimke (paraméter):
logikai kifejezés—modul-törzs;

ahol a „cimke" tetszőleges (egyértelmű) szimbólum, a paraméter elhagyható, de egész típusú változó, a logikai kifejezés szimbólumokra vonatkozó kiértékelhető predikátumokból van felépítve. Szerepük a következő: A modul-ra cimkével (aktuális paraméterrel) hivatkozhatunk a vezérlési struktúrának az algoritmusában. A modul akkor hajtódik végre, ha a vezérlési struktúrában a cimkére kerül a vezérlés és ugyanakkor a logikai kifejezés értéke igaz.

A modul-törzs aktuális framek egy listája aktuális paraméterekkel. Leírja az ilymódon egyként vezérelt frame-k *topológiáját* akként, hogy a frame-ek aktuális paraméterei között az azonos szimbólumok azonos információs pályákat reprezentálnak, szerepük a frame input vagy output-listában már meg van határozva; típusnak és attributumnak egyezni kell. Az információs pályák két modul törzsében leírt aktuális frame-ek között is futhatnak, mivel a modul csak a vezérlés szempontjából önálló egység.

A frame formája
frame interface specifikáció
specification specifikációs rész
implementation implementációs rész
endframe;

Ez az a keret, amelyet a rendszerkészítés során tartalommal tölt fel a felhasználó, s amely egyszerre egysége a probléma megfogalmazásnak és a realizálásnak is. Önállóan kidolgozható a benne hivatkozott frame-ek implementálása nélkül is.

A munka jelenleg kísérleti fázisban van, egyes részeket már realizáltunk.

Hivatkozások

[1] Dennis, J.B.: First version of a data flow procedure language. Lecture notes in computer science, 19 G. Goor, J. Hartmenis (szerk.) Springer-Verlag, 1974, 241—271

[2] Dávid Gábor: Architecture Language MTA SZTAKI Tanulmányok (kiadás alatt)

PARADOCS HOMOGÉN PÁRHUZAMOS SZÁMÍTÓGÉP ÉS DATAFLOW NYELVE

Domán András
BME

A tanulmány architekturát javasol egy nagypárhuzamosságú számítási rendszer kialakítására, amely a problémakörnyezetnek megfelelően könnyen struktúrálható. Ilyen architektúra gazdaságosan építhető az LSI ill. a modern VLSI technológia alkalmazásával.

1. Bevezetés

Párhuzamos számítási rendszerekről manapság egyre többet lehet hallani és ez nem csupán az újdonság divatteremtő erejének hatása. A téma előtérbe kerülése egyrészt a szükségletek, igények változását tükrözi, másrészt a megnövekedett lehetőségek szükségszerű követelménye. A számítástechnika megváltozott követelményei közül egyre nagyobb szerepet kap a számítási teljesítmény növelése, mivel egyre több az olyan feladat, melyhez a megoldási feltételt jelentő számítási sebesség (vagy gazdaságos sebesség) jóval meghaladja a hagyományos elven működő számítógépek belátható sebességhatárát.

A hagyományos (soros) gépek számítási teljesítményét a technológia bármely szintjén lényegében a komponensek kapcsolási sebessége és az átviteli késleltetés korlátozza. (E sebességeket az elektromágneses hullám terjedési sebessége ill. az atomi méretek határolják be.) A nagyobb teljesítmény elérésére olyan technikát kell alkalmazni, amely megengedi a különböző akciók párhuzamos, egyidejű végrehajtását. Természetesen a párhuzamosság különböző szinteken megtalálható volt már a legegyszerűbb gépeknél is. Pl: bit-párhuzamos aritmetika ill. memóriahozzáférés, később pedig az autonóm I/O csatornák. Ezeknél a megoldásoknál a párhuzamosság nem általános jelenség, igen korlátozott szerepű. A tényleges párhuzamosság – melynél az egyidejű, szimultán működés minden szinten, különösen processzor szinten érvényesül – olyan architekturális elv, amely a fénysebesség okozta működési sebességkorlátot lényegében feloldja. A sebesség persze a párhuzamos számításoknak korántsem az egyetlen motivációja. Sürgeti többek között még a megbízhatóság és az az igény is, hogy a számítógép architektúrák rugalmasabban idomuljanak a megoldandó problémákhoz, jobban tükrözzék az algoritmusok struktúráját.

A számítógépek párhuzamos működése a számítástechnika minden területén komoly változást fog előidézni. Az alkalmazások olyan területekkel bővülnek, amelyek eddig megoldhatatlanok vagy gazdaságtalanok voltak (real-time nagytömegű adatfeldolgozás, jelfeldolgozás, irányítási rendszerek, mesterséges intelligencia, tudományos számítások, nagy fizikai modellek stb.). A párhuzamosság pozitív hatást gyakorol a gazdaságosságra, megbízhatóságra is; alapvető változást fog eredményezni a software rendszerekben (párhuzamos programozás, párhuzamos algoritmusok). A fejlett számítástechnikai kultúrával rendelkező országokban a párhuzamos számítási rendszerek kutatása 8–10 éve folyik és az utóbbi években ez a folyamat lavinaszerűen felgyorsult, főként az LSI/VLSI – technológia előretörése nyomán. Számos kisebb-nagyobb párhuzamos számítógépet már megépítettek (pl. Illiac IV. CDC–STAR, STARAN, Cmmp, CRAY–1, PEPE, TI–ASC, AP 120B, IBM 3838, SMS 201, ICL–DAP stb.).

A technológiai fejlődés és a párhuzamos számítási folyamatok elemzése, értékelése együttesen olyan architektúrák kutatás-fejlesztését ösztönzi, melyek alapelemei (processzorelemek) azonosak és egyszerű felépítésűek, csupán 1–2 szónyi memóriát tartalmaznak. Lényeges ugyanakkor az is, hogy szervezésük ill. a működtetés sajátságai ne korlátozzák az al-

kalmazható processzorelemek számát, így a párhuzamosság fokát. Ezeknek a követelmények is eleget tesz a PARADOCS (PARAllel Dataflow Organic Computing System) homogén párhuzamos processzor, melynek modellje ill. alkalmazási környezete kerül ismertetésre a továbbiakban.

2. Párhuzamos feldolgozás (Parallel Processing)

2.1 A párhuzamosság lényege

A számítási párhuzamosság a különböző feldolgozási, számítási akciók egyidejű, szimultán végrehajtását jelenti. A megoldandó feladatok független particiókra bontásával a particiókhoz rendelt műveletvégző egységek egyidejűleg, párhuzamosan működhetnek – lényegesen lerövidítve így a számítási időt.

Ismeretelméleti oldalról tekintve, érdemes végiggondolni azt, hogy feladatmegoldásainkban, a valóság megismerésében mindig modellekkel dolgozunk: a valóság modelljeivel. Márpedig a valóság egyidejű folyamatok rendszere, még akkor is, ha léteznek bizonyos logikai egymás után rendelések (szekvenciák). Soros modellezési technikával a valóságot közvetlenül nem tudjuk modellezni, csupán a valóság párhuzamos modelljét utánozhatjuk, szimulálhatjuk. Ugyanakkor a párhuzamosság általánosabb fogalom is, mivel ennek határesete a soros modellezés (a párhuzamosság foka ekkor 1). A modell és a modellezési eszköz – esetünkben a számítógépes feldolgozás – a párhuzamosság szintjén közelíthetők egymáshoz.

2.2 A párhuzamos feldolgozás problématerületei

A párhuzamos számítások tanulmányozása, kutatása már is több, jól elkülöníthető területre oszlik, melyek körvonalai a szakirodalomban is érzékelhetők. Ezek: a párhuzamos számítógép architektúrák (konstrukciói), programnyelvek ilyen gépekhez, párhuzamos algoritmusok, alkalmazási területek, párhuzamos folyamatok vezérlési kérdései stb. Mielőtt azonban néhány területről rövid betekintést adnánk, érdemes kissé részletesebben elemezni a hagyományos számítógép architektúrák problémáit. Így érthetőbbé válnak az architektúra megváltoztatását célzó markáns törekvések.

2.2.1 Hagyományos számítógépek hátrányai

Neumann János volt az első, aki átfogóan megfogalmazta a digitális számítógépek alapvető működési és architekturális elveit (1946). Ezek az elvek azóta is változatlanok és a ma működő legtöbb gép ezekre épül.

Az elvek lényege:

– alacsony szintű (gépi nyelvű) *utasítások* és *adatok tárolására* ugyanaz a tárolókészlet szolgál,

– a tároló *lineáris*an szervezett sorszámozott (címezhető) egységekből áll,

– a számítás *szekvenciális* központi irányítás mellett történik.

Ezek az elvek biztosították a digitális számítógépek strukturális egyszerűségét és a logikai elemek minimális számát. Ameddig a logikai (aktív) elemek sokkal drágábban és kevésbé megbízhatóak voltak, mint a memóriaelemek, kevés külső berendezés üzemelt és a digitálisan megoldott problémák egyszerűek voltak – ezek az elvek feltétlenül progresszívnek bizonyultak. Ma azonban a számítási technológia gátjává váltak. A hagyományos számítógépek strukturális egyszerűsége ma már strukturális merevségnek bizonyul és komplikációkat eredményez a software rendszerekben is, amelyek lassan elérik a komplexitás felső határát anélkül, hogy közben a felhasználók minden igénye kielégülne. Nyilvánvaló az is, hogy a klasszikus tárolók lineáris felépítése a legtöbb valóságos feladatnál rosszul idomul az adat- és programstruktúrához. Ennek az is következménye, hogy bonyolult adatstruktúrát kezelő feladatnál igen

belső memóriakapacitás, kifinomult hardware processzor vagy igen nagy gepidő szükséges. Problematikus a hagyományos memória azért is, mert a programozót rákényszeríti, hogy ne csak a kiszámított értékekkel törődjön, hanem azok „lelőhelyére" is gondolnia kell. A memória persze a számítási teljesítmény növelésének közvetlenül is komoly akadálya. A memória és a CPU közé képzelt „cső" a számítógép egyik legszűkebb keresztmetszete (bottleneck); az adatforgalom nagy részében nem effektív adatokat szállít, csupán az adatok nevét. Ezeknek a neveknek (hivatkozásoknak) az előállítása amellett nagy műveleti teljesítményt köt le.

Az alapvető „bottleneck" a Neumann típusú számítógépeknél a szekvenciális központi vezérlés elve, amely az aszinkron működés, az elosztott feldolgozás gátja. A szekvenciális vezérlésnek a korlátozott átbocsátóképességen kívül az alacsony kihasználtság is követelménye. Több százezer vagy millió alkatrészből álló rendszernek (a memóriát is beleértve) egyidejűleg csupán néhány száz eleme lehet aktív. Összefoglalva a Neumann-elvű gépek alapproblémáját megállapítható, hogy a gép struktúrája nem képes a számítási folyamat struktúráját hatékonyan adaptálni és ennek feloldásához az alapelvek teljes felülvizsgálata szükséges.

2.2.2 Párhuzamos számítógép architektúrák

Párhuzamos számítógépek osztályozására a legfontosabb és leginkább elterjedt Flynn taxonómiája [1]. Ez az utasításokat (I) és adatokat (D) egyszeres (S) ill. többszörös (M) stream-ekre (folyamokra) osztja. (Az osztálynevek a rövidítések összetételéből származtathatók.) Ennek megfelelően a Neumann típusú gép a SISD kategóriába sorolható. A SIMD (egyszeres utasításfolyam, többszörös adatfolyam) osztály magába foglalja a tömbprocesszorokat (pl. Illiac IV.) és az asszociatív processzorokat. Ezek lényege, hogy a központi egység egy szekvenciális program végrehajtása során az utasításokat (vezérléseket) minden processzorelemhez egyidejűleg eljuttatja, mintegy szétsugározza (broadcast). Az identikus processzorelemek ezt azonosan végrehajtják különböző adatelemeken: lokális memóriájuk tartalmán. Az asszociatív processzor esetében (pl. STARAN, PEPE) az adat ill. processzorcímzés asszociatív, azaz tartalom (és nem cím) szerint történik. A SIMD processzorok problémái: specializált alkalmazási területek a processzorelemek funkcionális uniformitása miatt (lineáris algebra, parciális differenciálegyenletek, adatfeldolgozás stb.), memória hozzáférési konfliktusok, stb. A pipeline processzorok alkotják a SISD architektúra-osztályt. A pipeline processzorok (pl. CDC-STAR, IBM 360/195, CRAY-1, TI-ASC, AP 120B, MATP) a futószalag elvet alkalmazzák. Bizonyos részfeladatokra specializált processzorok mintegy „csövet" alkotva, sorban kapcsolódnak egymáshoz és mindegyiket a számítási folyamat soros dekompozíciójával nyert részfolyamatokhoz egyenként hozzárendeljük. A feldolgozandó adatfolyam (stream) ezen a „csövön" (pipe) halad át úgy, hogy bármely időpillanatban a stream különböző elemei a feldolgozás különböző fázisában tartózkodnak. A pipeline elvet a feldolgozás legkülönbözőbb szintjein lehet alkalmazni. A tárgyprogramot tekintve stream-nek, ún. utasításfeldolgozó pipeline alakítható ki (pl. IBM 360/195). Tekinthetjük stream-nek az adatok sorozatát is: aritmetikai ill. makropipeline stb. (pl. CDC-STAR, TI-ASC). A negyedik Flynn-osztály a MIMD kategória, a multiprocesszorok osztálya. A multiprocesszor (pl. Cmmp, Univac 1110, Honeywell MULTICS, Cm*) olyan számítási rendszer, amelyben a processzorok különböző adathalmazokon egyidejűleg különböző feldolgozást végeznek és közös hozzáférésük van a memóriákhoz, valamint az I/O csatornákhoz. A komplexumot egyetlen integráns operációs rendszer vezérli. A multiprocesszor valójában Neumann típusú gépek együttese és problémái is hasonlóak ahhoz – más szinten. Kivételt jelent az ún. Dataflow gépek csoportja ill. irányzata, melynél az „adatvezérelt" elv maximális aszinkronitást, maximális párhuzamosságot biztosít anélkül, hogy külön integráns vezérlési struktúrát kellene kialakítani.

Az osztályozással kapcsolatban meg kell jegyezni, hogy a Flynn kategóriáival elkülöní-

tett architekturális vonások általában nem tisztán jelentkeznek a megvalósított párhuza mítógépeknél, hanem keverten.

2.2.3 Párhuzamos programozási nyelvek

A párhuzamos programozás területére inkább a problémák, mint az eredmények a lemzőek. A jelenlegi párhuzamos architektúrák túlságosan meghatározóak a programozá nyelvek kialakításában. Más programozási és algoritmus elveket használunk ugyanazon fe dat megoldására egy tömbprocesszoron, egy pipeline gépen vagy multiprocesszoron. Pár mos programozási nyelvek létrehozását erősen gátolja még a Neumann-elvű gépek ill. a mann típusú nyelvek tudatunkba merevedésének hatása is. Jelenleg a párhuzamos progr zásnak különböző útjai figyelhetők meg. Vannak olyan programnyelvek, melyek hagyom nyos magasszintű programnyelvek kiterjesztésével valósítják meg a párhuzamosságot, pl. Tranquil (Algol), Ivtran (Fortran), PFOR (Fortran). Más nyelvek, melyek egy-egy tényl vagy absztrakt architektúrához kapcsolódnak, lényegében nyelvi előzmények nélkül – a huzamosság exponálásával – születnek. Egy más irányzat a szekvenciális szemléleten ne változtatva, olyan rendszert javasol, amely a hagyományos nyelven megírt programot a séges párhuzamosítás szempontjából elemzi és olyan kódra ill. vezérlő táblázatokra fordí amely párhuzamos gépet működtet. Így a programozónak nem kell tudnia, milyen szám gépen fut a programja.

2.2.4 A Párhuzamos algoritmusok, alkalmazások

A párhuzamos numerikus algoritmusok a párhuzamos információfeldolgozás nélkülö tetlen elemei; ezek nélkül hatásos feldolgozás nem képzelhető el. Jelenleg már sok felad feladatosztályra létezik kidolgozott párhuzamos algoritmus. Kidolgozták pl. az aritmetik kifejezések párhuzamosításának csaknem teljes elméletét. Igen jelentős szerepet kap az al ritmusok kialakításában az az észrevétel, hogy a nagy számításigényű feladatok (ezeket é mes és célszerű párhuzamosítani) jelentős része megfogalmazható mátrix formájában (pl. differenciálegyenletek, lineáris egyenletrendszer, Fourier transzformáció, konvolúció, kor ciószámítás, polinomszorzás, approximáció stb.). Ezek megoldása jól párhuzamosítható m rixszámítások útján lehetséges. A párhuzamos számítási algoritmusok területe igen nagy t dés előtt áll (a szakirodalomban jól követhető ez). Vannak azonban bizonyos tényezők, amelyek nehezítik a kutatást. Ezek alapvetően abból erednek, hogy a párhuzamos algorit sok nem a soros – szokásos – algoritmusok általánosítása; az ilyen algoritmusok kifejez séhez döntően új szemlélet szükséges, ami különbözik a soros gépek és programozási nyel környezetétől.

Párhuzamos számításokat ma már sok területen alkalmaznak. Elsősorban: real-time nyítási rendszereknél: katonai rendszerek, repülési, irányítási rendszerek; mesterséges int gencia, képfeldolgozás, nagy fizikai modellek: meteorológiai- hidrodinamikai- nukleáris m lezés, geológia, térképészet, orvosi alkalmazások: rtg tomográfia, ekg, eeg; adatfeldolgoz operációkutatás stb. területén.

3. PARADOCS párhuzamos processzor modellje

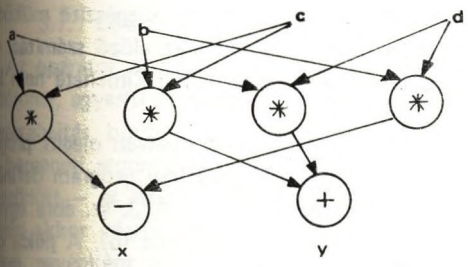
3.1 Dataflow séma és a DFLL módosított Dataflow nyelv

A DFLL, a javasolt párhuzamos számítógép a PARADOCS közepes szintű progra si nyelve, amely egyben párhuzamos számítási algoritmusoknak egy alkalmas leírási mód az inherens párhuzamosságot jól kifejező lehetőséggel. A Dataflow (adatfolyam) sémán nyelvi reprezentációjának a DFLL-nek ismertetése feltárja a modell lényegét, működését ezért az itt következőkben foglalkozunk ezzel, megelőzve az architektúra tárgyalását.

Minden számítási folyamat elemi műveletekre (operátorokra) bontható, amelyek bemeneti adataikon (operandusaikon) ill. kimeneti adatukon (eredményükön) keresztül kapcsolódnak egymással. Az ilyen rendszer nagyfokú párhuzamosságot mutat, feltételezve az operátorok független működését. A klasszikus Dataflow séma, amely Dennis, Karp-Miller, Rodrigez, Gostelow munkái [2], [3] nyomán vált ismertté, ezt a gondolatot viszi tovább és lényege röviden a következő:

1. A műveletek akkor és csak akkor hajtandók végre, ha az összes operandusuk rendelkezésre áll (adatvezérlés, aszinkronitás)
2. A számítás a kiszámított értékekre alapul és nem a helyekre, ahol azok találhatók (funkcionalitás)

Elfogadva ezeket az elveket könnyen megvalósítható egy nagymértékben konkurrens, aszinkron programozás ill. programvégrehajtás, a párhuzamos programkonstrukciók (fork-join, cobegin stb.) szükségessége vagy bármilyen fajta programanalízis nélkül. Az adatvezérlés fogalma itt alapvető fontosságú, arra utal, hogy minden adat vezérlést hordoz magában, másszóval az adatfolyamat és a vezérlési folyamat nem válik külön – mint a hagyományos számítási rendszerek esetében.



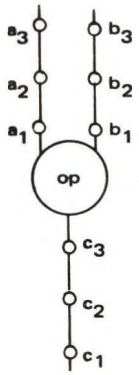
$$x + yi = (a + bi) - (c + di)$$

1. ábra Két komplex szám szorzatának egyszerű Dataflow sémája

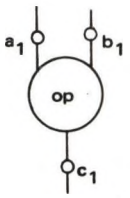
A PARADOCS modell alapjául szolgáló módosított Dataflow séma valójában egy irányított gráf, az ún. Dataflow gráf, amely információs csatornákkal (élekkel) kapcsolódó funkcionális operátorokból (csúcsokból áll. Ezek az operátorok olyan függvényt definiálnak (pl. összeadás, szorzás, feltételes adat- ill. vezérlésátadás), amely az input élen megjelenő információból (adatérték) output információt képez. Az output élek további operátorok input éleibe kapcsolódnak ... és így tovább; ilyen módon alkotva az operátorstruktúrát.

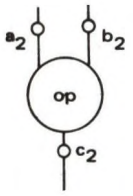
A PARADOCS Dataflow koncepciójának lényegéhez tartozik az, hogy a modellben nincs önálló funkciójú, címezhető memória (operatív tár). Ennek az a magyarázata, hogy minden adatot (argumentumot), amelyet memóriából olvasnánk ki, itt megfelelő operátorok kimenetéről nyerjük. Így a műveletek adatokon (eredményeken) keresztül kapcsolódnak egymáshoz memória közbeiktatása nélkül. (Ehhez nyelvi megfeleltetése a változó nélküli programozás. Ahol a hagyományos programozás szerint változót szerepeltetnénk, oda a változót generáló, létrehozó operációt – végső esetben input operációt – írunk.) Természetesen ez olyan nagy számú, egymástól függetlenül működő operátort igényel, amennyi az adott algoritmusban az elemi műveletek száma; és ez a szám igen tekintélyes lehet. Van azonban a párhuzamosság megvalósításának más lehetséges módja is. Megengedhetünk az operátoroknak egynél többszöri működést is, melynél az algoritmust leképező operátorok száma lényegesen kevesebb lehet,

nem minden esetben csökkentve a párhuzamosság fokát. Ez a működési mód az ún. statikus párhuzamosság 4, amely valójában a pipeline működési mód (2.a ábra). Az előző, „re-entrant" működési mód az ún. dinamikus párhuzamosság (minden „software operátor"-nak egy „hardware operátor" felel meg; lásd 2.b ábra).



2.a ábra

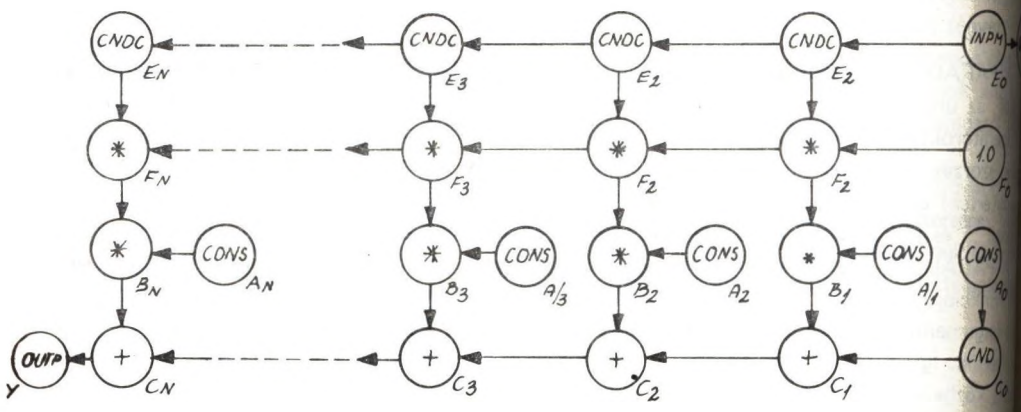




2.b ábra

A statikus párhuzamosság esetében, az adattorlódás elkerülését egy kiegészítő működési feltétel biztosítja. Eszerint az operátor nem működhet addig, ameddig az előző számítási eredményt a rákövetkező, kapcsolódó operátorok nem fogadták el, azaz amíg kimenete nem üres.

A Dataflow gráf és a módosított Dataflow nyelv, a DFLL ill. ezen a nyelven megírt program kölcsönös egyértelmű megfeleltetései egymásnak. Egy jól formált Dataflow program definiál egyrészt az operátorokat, másrészt azok kapcsolódását, a konfigurációt. A 3. sz. ábra egyszerű polinomszámítás algoritmusának Dataflow gráfja (a) és DFLL programja (b). A példa csupán illusztráció, részletei később válnak érthetővé.



3.a ábra Egyszerű polinomszámítás pipeline elvű algoritmusának Dataflow gráfja

```
          TITLE(40,##POLINOM-KIÉRTÉKELÉS##);
COMMENT   OPERÁTOR DEFINICIÓK;
          FOR I:=0 STEP 1 UNTIL N DO    BEGIN
            A[I]:-CELL(CONS,INPR(I),1);
            B[I]:-CELL(MULT,PNZ,2);
            C[I]:-CELL(ADD,PNZ,2);
            D[I]:-CELL(CNOT,PNZ,1);
            F[I]:-CELL(MULT,PNZ,2); END ;
            G[ ]:-CELL(CNOT,PNZ,2);     E[ ]:-CELL(INPR,N,1);
            X:-CELL(OUTP,RANK(BASE),-1); Y:-CELL(OUTP,RANK(BASE),-1);
            F[ ]:-CELL(CONS,1.0,2);
COMMENT   GRÁFIKUS ALPRAS;
          FOR I:=  STEP 1 UNTIL  DO
GRAF:-G[I],DA(G[I-1],E[I]),DA(A[I],F[I],DA(F[I-1],F[I],SA(G[I-1])));
          GRAF:-X.SA(G[ ]); GRAF:-Y.SA(G[N]);GRAF:-G[ ],SA(A[ ]);
              EXCUT(4,  );
```

3.b ábra A polinomszámítás DFLL programja

3.1.1 Operátorok

Az operátor egyszerű aritmetikai, logikai ill. ezekhez kapcsolható vezérlési műveletek szimbóluma, egyesíti magában a műveletvégző és speciális vezérlő funkciókat. A tényleges műveletvégrehajtás, bármely operátornál, csak akkor jön létre, ha az adott operátor összes szükséges bemeneti argumentuma rendelkezésre áll – azaz az operandust előállító, előző művelet befejeződött – és emellett az operátor kimenete üres. A művelet eredménye az operátor kimenetén akkor jelenik meg, ha az eredmény előjele (ill. 0 értéke) megegyezik az előírt feltételekkel, egyébként nem. Műveletvégzés után az input értékek megszűnnek.

Az operátor definíció DFLL formája:

<NAME>:–CELL (<OPCOD>,<COND>,<ADDR>);

ahol

NAME – operátorazonosító (lehet tömbelem is)

OPCOD – műveleti azonosító (lásd 1. sz. táblázat)

COND – eredményre vonatkozó feltételi előírás, lehetséges értékei: P, N, Z ill. ezek kombinációi (adattovábbító és kommunikációs operátoroknál más jelentés; 1. tábl.)

ADDR – operandus hivatkozás (előírja, hogy az eredmény, mint a rákövetkező kapcsolódó operátorok bemeneti adata hányadik operandusként kerüljön tovább), lehetséges értékei: 1, 2 ill. 3 (OUTP operátornál értelmezés más, lásd 1. sz. táblázat)

	azonosító	jelentés	argumentumok száma	megjegyzés
adatváltoztató műveletek	ADD	Y = A + B	2	
	SUB	Y = A – B	2	
	MULT	Y = A x B	2	
	DIV	Y = A / B	2	
	AND	Y = A ∧ B	2	
	OR	Y = A ∨ B	2	

	azonosító	jelentés	argumentumok száma	megjegy
adattovábbító műveletek	CNDCT	Y = A (B)	1	
	CONS	Y = cons	0	COND=cons
	IDNT	Y = cons	1	COND=cons
kommunikációs műveletek	INPS	Y = x,x,x...	0	COND=adatstr
	INPM	$Y = x_1, x_2, \ldots$	0	hossza
	OUTP	A ill. B outp.	1	COND=RANK (§<NAME>§
vezérlő műv.	SWTCH	$Y = \begin{cases} A \text{ ha } C \geq 0 \\ B \text{ ha } C < 0 \end{cases}$	3	
összehasonlítás és csere	OCHP	*IF* feltétel igaz	1	$A_{kezd} = +\infty$
	OCHN	*THEN* CHANGE (A,B);	1	$A_{kezd} = -\infty$
	OCHZ	Y = B;	1	$A_{kezd} = 0$

Műveleti előírások (A,B,C változók az 1,2,3 operandusok szimbóluma; x — beolvasott a

1. sz. táblázat

Példa: A :– CELL (SUB, PZ, 2);

jelentése: létrehoztunk egy A-nevű operátort, amely aktivizáláskor kivonás műveletét v el két operandusán és ha az eredmény pozitív vagy zérus, akkor megjeleníti kimenetén, a kapcsoló operátorok 2. argumentumaként.

Q [I] :– CELL (OUTP, RANK (§O§),I);

jelentése: a létrehozott Q [I] azonosítójú operátor a bemenetére kerülő adatot megjel fejlécben Q karakter és I változó értéke fog szerepelni.

3.1.2 Dataflow gráf

A már létrehozott operátorok összekapcsolásával tetszőleges párhuzamos számítási t túrát alakíthatunk ki. Ez a struktúra maga a Dataflow gráf ill. konfiguráció és mint irá gráf, alkalmas nyelvi eszközökkel leírható. A DFLL nyelv két alapstruktúrát definiál és rekurzív kiterjesztésével építi fel a Dataflow gráfot.

Alapstruktúrák:

1. SA = Single Argument (4.a ábra)

GRAF :– a.SA (b);

4.a ábra

„b" csúcsot (operátort) „a"-ba irányítja és GRAF értéke az „a"-ra mutatás lesz (a,b,G vatkozási típus azonosítók, míg „ :– " hivatkozási értékadás).

2. DA = Double Argument (4.b. ábra)

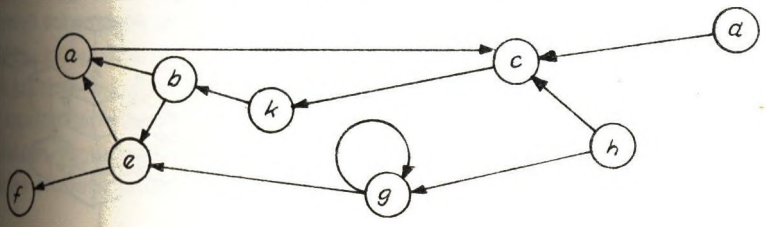
GRAF :– a.DA (b,c);

4.b ábra

mind a „b" mind a „c" csúcsot „a"-ba irányítja, GRAF értéke az „a"-ra mutatás lesz.
A struktúrák szuperponálhatók, így bármely gráf megadható csupán egy alapstruktúra alkalmazásával. A második struktúra (DA) pl. megadható az első (SA) segítségével:

GRAF:–a.DA (b,c); = { GRAF:–a.SA (b); GRAF:–a.SA (c); vagy GRAF:–a.SA (b); GRAF:–GRAF.SA (c);

Példa egy összetettebb részgráf megadására (5. ábra):



GRAF :– f.SA (e.DA (g.DA (g,h), b.SA (k.SA (c.DA (h,d)))));
GRAF :– c.SA (a.DA (b,e));

Dataflow gráf és DFLL leírása

5. sz. ábra

Abban az esetben, ha a Dataflow gráf ismétlődő részgráfokat tartalmaz – ami igen gyakori – akkor a részgráf leírását ciklusba is ágyazhatjuk (Algol szintaxist alkalmazva). Ebben az esetben az operátorazonosítók tömbelemek (3. ábra).
A DFLL szintaktikája megegyezik a SIMULA 67 (Algol) szintaktikájával (részletes ismertetésére a korlátolt terjedelem miatt itt nem vállalkozhatunk). A DFLL, amely az alkalmazás szempontjából szimulációs nyelv (a PARADOCS lehetséges programnyelve) igen rövid idő alatt lett implementálva SIMULA 67-ben (CDC–3300; MTA–SZTAKI). A SIMULA egyedülálló kifejezési erejének köszönhetően a teljes DFLL értelmező-szimuláló program mindössze 120 soros. Ebbe a programba van beágyazva a DFLL program, mint prefixelt blokk, így ennek szintaktikai ellenőrzését maga a SIMULA fordító végzi.

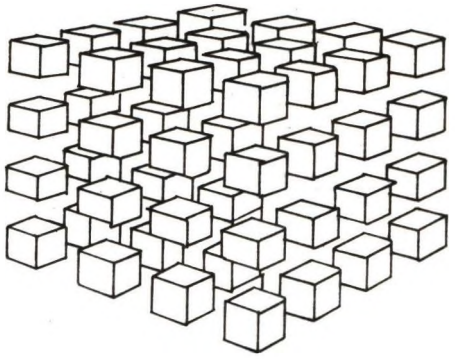
3.2 A PARADOCS felépítése

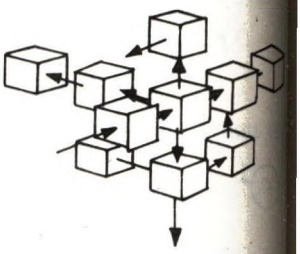
3.2.1 Fizikai konfiguráció

A PARADOCS párhuzamos processzor koncepcióját a növekvő számítási igények, valamint a párhuzamos rendszerek és a sejtautomaták területén elért eredmények hívták életre. Az architektúra kialakítása és az alkalmazási környezet kimunkálása több éve folyik. A cél olyan univerzális számítógép elvének és megvalósítási lehetőségeinek kidolgozása, amely nagyszámú (esetleg 10^4–10^5) egyszerű működésű, egyszerű kapcsolódású processzor elemből áll.

A PARADOCS architektúrája a sajtautomaták elvén alapul. Ez racionális megoldást nyújt a processzor elemek (sejtek) összekapcsolására és a feladatosztályok igen széles spektruma szá-mára optimális párhuzamos számítást biztosít. A rendszer flexibilitása azt jelenti, hogy az architektúrában kialakítható konfigurációk jól tükrözik a megoldandó feladat algoritmusát, jól képesek azt adaptálni.

A dolgozat célja szempontjából elegendő az arhitektúrát úgy tekinteni, mint azonos processzorok tömbstruktúrált hálózatát, amely folyamatosan számol és adatot továbbít a processzorok terében, a sejttérben. Minden sejt – egy szabályos 3-dimenziós tér rácspontjaiba kézelve – 6 másik sejttel szomszédos, információcserét velük bonyolíthat le. A ténylegesen szomszédos sejteken kívül létezik még egy virtuális szomszéd, ez önmaga a sejt. Így bármely processzor elem hét processzornak továbbíthat ill. azoktól kaphat információt. Ez a rögzített kapcsolódás az ún. fizikai konfiguráció.





fizikai konfiguráció *logikai konfiguráció*

7. sz. ábra

Létezik emellett – bármely konkrét megoldandó feladat algoritmusához kapcsolódva – az ún. logikai konfiguráció (7. ábra), amely a fizikai konfiguráció gráfjának részgráfja, a sejttérben információterjedés lehetséges pályáit reprezentálja az adott algoritmus esetében. Ezt a logikai konfigurációt programozással alakíthatjuk ki olyan módon, hogy a működés kezdetén minden sejthez egyértelműen hozzárendelünk egy utasítást, amely többek között a sejt kimeneti információinak irányítását is előírja – a hét lehetséges irány közül akármennyit kiválasztva. Az így létrehozott logikai konfigurációban – programozott struktúrában – azután az információk, mint valami csőrendszerben, egyidejűleg áramlanak, miközben minden gráfpontban, sejtben valamilyen műveletvégzés történik rajtuk. Legkedvezőbb esetben az információk folyamatosan haladnak, tárolásuk nem szükséges.

3.2.2 Az egyedi sejt felépítése

Minden sejt egyszerű CPU-t és – a felhasználó szempontjából – 4 regisztert tartalmaz, ebből három az operandusok, egy pedig az utasításszó tárolására. Az utasításszó 4 önálló funkciójú információt tartalmazó mezőből áll:

a) műveleti kód (4 bit)
b) operandus cím (2 bit)
c) feltételmező (3 bit)
d) irányításvektor (7 bit)

Ezek az információk egyértelműen meghatározzák azt, hogy aktivitás esetén a sejt milyen műveletet végez (a); az eredményt a kapcsolódó sejtek mely regiszterébe juttatja (b); az eredményre vonatkozólag milyen feltételeket érvényesít (c); és az eredményt mely sejtek kapják meg a lehetséges 7 közül (d). A sejt sajátossága, hogy eredmény-információja egységes értékű minden irány felé, különbség csupán az, mely szomszédok azok, amelyek ezt az információt meg is kapják. A sejtek egyidejű konfliktusmentes működését a Dataflow elv biztosítja. Minden aktív processzor elem, a működés során, műveletet hajt végre és az eredményt addig tartja a kimenetén, amíg azt a kapcsolódó sejtek inputként nem tudják fogadni. Ezután a sejt műveletvégzésre újabb információkat fogadhat. A sejtek így folyamatosan működnek addig, ameddig aktivitás van a térben.

3.2.3 A DFLL és az architektúra

A DFLL és az architektúra összevetése mindezek után természetesnek tűnik. Minden operátornak egy-egy sejtet feleltetünk meg, melynek utasítása az operátordefinícióból származtatható. A logikai konfiguráció pedig nem más, mint az operátorstruktúra, a Dataflow gráf. Mindezen rokonságok ellenére egy DFLL program leképezése a sejttérre nem problémamentes. A DFLL, az operátorok kapcsolódására, a gráfra vonatkozólag nem tartalmaz korlátozásokat. Ezzel szemben a logikai konfiguráció, mint a fizikai konfiguráció részhalmaza szigorú geometriai, kapcsolódási szabályokat követ. Az ellentmondás feloldása – a gráf kibővítése – egy későbbiekben tervezendő DFLL fordítónak lesz a feladata. Addig is egyedi, intuitív megoldások könnyen alkalmazhatók.

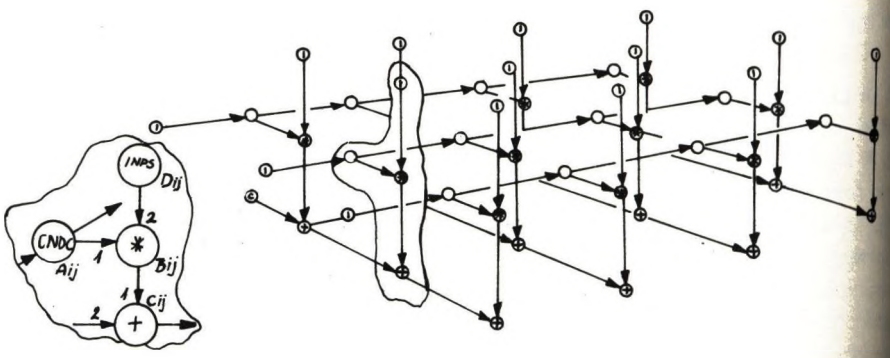
3.2.4 Globális architektúra

A párhuzamosság a természetben, így annak modelljeiben – feladatmegoldásainkban – nem abszolutizálható. Minden algoritmus tartalmaz szekvenciális elemeket is, melyek megoldása hagyományos számítógépen kényelmesebb, gazdaságosabb lehet. Emellett a párhuzamos számítási módszerek, algoritmusok, software- és egyéb eszközök fejlettsége nem jutott még arra a szintre, amelyet a szekvenciális számítások gyakorlata kb. 30 éves fejlődés után elért. Mindezekből az következik, hogy a jövőben valószínűleg – de a közeljövőben biztosan – a párhuzamos számítógépek csak szekvenciális géppel valamiféle feladatmegosztásban lesznek képesek működni. A PARADOCS rendszerben a sejttér információs környezetét is egy soros gép (host) végezné, melynek fő funkciói: DFLL programok fordítása (geometriai optimalizálással kiegészítve), betöltése a sejttérbe (load), majd működés közben (operate) a sejttér pereme és a külvilág közötti interface valamint hibafigyelés.

3.3 A kutatások jelenlegi állása

Az első architekturális elképzelések megszületését egy FORTRAN nyelvű szimulátor elkészítése (ICL–1905; MÜM–SZÁMTI) segítette, ennek feladata volt a sejt és a sejttér logikai működési helyességének vizsgálata, az architektúra fejlesztése és kezdeti algoritmusok kidolgozása. Ezt egy hosszabb irodalomkutatás követte, mely az architektúra koncepció helyét, az alkalmazási és algoritmusosztályokat jobban körvonalazta. Az irodalomkutatás egyik eredménye egy szakbibliográfia készítése volt [5]. A párhuzamos algoritmusok leírási kényszere és a Dataflow koncepció inspirálták a DFLL nyelv (DataFlow Like Language) kidolgozását, mely az algoritmusok fejlesztésének szinte nélkülözhetetlen eszköze lett. Több feladatosztály egy-egy reprezentáns párhuzamos algoritmusa készült el ezen a nyelven: mátrixszorzás (8. ábra), keresés, rendezés (9. ábra), parciális differenciálegyenletek, gyors Fourier transzformáció, polinomszámítás (3. ábra), mintaillesztés, Turing gép szimuláció stb.) A bemutatott DFLL mintaimplementációkhoz magyarázatként mellékeltük a Dataflow gráfot.) További algoritmus fej-

lesztési irányok: lineáris algebra további feladatai, jelfeldolgozás, alakfelismerés, differenci egyenletek megoldása, statisztikai számítások stb. A párhuzamos algoritmusok kiterjedt k tásával és a DFLL fejlesztésével egyidejűleg vizsgálatok folynak a sejttér megvalósításának technikai, technológiai lehetőségeiről. Ezzel kapcsolatban elkészült a sejt egyszerűsített C nyelvű (Chu: Computer Desing Language) hardware leírása. Ennek alapján egyetemi hallg bevonásával a sejt néhány mintaelemének megvalósítása folyik. A host egy Intel 8080-ra lő mikrogép vagy LSI–11 lesz.

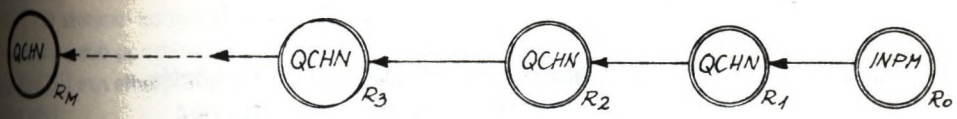



```
COMMENT     OPERATORDEFINICIOK:
            FOR I:=1 STEP 1 UNTIL N Do A[0,I]:=CELL(INPN,N,1);
            FOR J:=1 STEP 1 UNTIL N Do
            FOR I:=1 STEP 1 UNTIL N Do    BEGIN
              A[I,J]:=CELL(CNDCT,PNZ,1);
              D[I,J]:=CELL(INPS,N,2);
              B[I,J]:=CELL(MULT,PNZ,1);
              C[I,J]:=CELL(ADD,PNZ,2);      END ;
            FOR I:=1 STEP 1 UNTIL N Do    BEGIN
              C[I,0]:=CELL(CONS,0,0,2);
              E[I]:=CELL(OUTP,RANK(*E*),I);   END ;
COMMENT     STRUKTURALIS LEIRAS:
            FOR I:=1 STEP 1 UNTIL N Do    BEGIN GRAF:=E[I].SA(C[I,N]);
            FOR J:=1 STEP 1 UNTIL N Do
GRAF:=C[I,J].DA(C[I,J-1].PT,I).DA(D[I,J].DA(C[I,J].A[I,J].SA(A[I-1,J])));  END ;
      EXECUTE(25,1);
```

8. sz. ábra Mátrixszorzás Dataflow gráfja és DFLL programja

3.4 Következtetések

Az ismertetett modell Dataflow elvű, sejtszerű elrendezés architekturális, nyelvi és ritmus kérdéseibe próbált rövid bepillantást adni. (Hasonló, de nem univerzális célú archi túrakutatást több helyen is folytatnak [6], [7].) Megállapítható, hogy az eddig kidolgozo nagypárhuzamosságú algoritmusok igen jól adaptálhatók ilyen rendszerre, melynek fő szi si elve a (többdimenziós) pipeline. Ugyanakkor az egyszerű, uniform elemekből álló rend Neumann-típusú számítógéppel kapcsolódva gazdaságos megoldást kínál LSI–VLSI techno gián történő realizálásra.




```
DATAFLOW BEGIN   INTEGER I,M; M:=INTVAL;
         BEGIN   REF (SEQIT) ARRAY P,E[0:M];
          TITLE(40,'**RENDEZES**');
COMMENT   OPERATORDEFINICIOK;
          FOR I:=0 STEP 1 UNTIL M Do   BEGIN
          P[I]:-CELL(QCHN,M2,2); E[I]:-CELL(OUTP,BANK(SPS),I); END ;
          P[0]:-CELL(INPM,2*M,2);
COMMENT   STRUKTURALETRAS;
          FOR I:=1 STEP 1 UNTIL M Do
      GRAF:-E[I].SA(P[I]).SA(P[I-1]); GRAF:-E[0].SA(P[0]);
           EXECUTE(3*M);
          END  SORT;
        END  DATAFLOW BLOKK;
  END ;
      FINIS
```

9. sz. ábra Rendezés Dataflow gráfja és (teljes) DFLL programja

Irodalomjegyzék

[1] Flynn, M.J.: Some Computer Organisation and their Effectiveness, IEEE Trans. on Computers 1972/9.

[2] Dennis, J.B.: First Version of a Data Flow Procedure Language. MAC TM 61. MIT 1975.

[3] Arvind–Gostelow, K.P.: Dataflow Computer Architecture: Research and Goals. Technical Report □ 113 University of California, Irvine 1978.

[4] Treleaven, P.C.: Principal Components of a Data Flow Computer. EUROMICRO 1978.

[5] Domán, A.: Párhuzamos számítási rendszerek. Szakbibliográfia. NJSZT–MTA SZTAKI 1978.

[6] Kung, H.T.–Leiserson, C.E.: Systolic Arrays for VLSI. Comp. Sci. Rev. Carnegie Mellon University 1977/78.

[7] Weimann, C.F.–Grosch, C.E.: Parallel Processing Research in Computer Science. Proc. Computer Architectures 1978.

POSTA DIGITÁLIS TÉRMODELL

Számítógépes térkép felhasználása a távközlés területén tervezési, számítási feladatokhoz

Emődi Ervinné–Szász György
POSTA SZÁMÍTÁSTECHNIKAI ÉS SZERVEZÉSI INTÉZET

1. Bevezetés

Intézetünk, a Posta Számítástechnikai és Szervezési Intézet, a Posta Kísérleti Intézettel (továbbiakban: PKI) együttműködve létrehozta a Digitális Térmodell (továbbiakban: DTM) rendszert, amely egy alap-fileból (DTM-file) és az ehhez kapcsolódó felhasználói programrendszerből áll.

A DTM-file egy számítógép által értelmezhető térkép, amely Magyarország terepadatait tartalmazza.

A DTM-filet felhasználó programrendszer elsősorban a vezetéknélküli távközlés területén felmerülő műszaki-tudományos számításokat és hálózattervezési feladatokat kívánja megoldani.

A DTM alapelvét térképész szakemberek bevonásával a PKI dolgozta ki. Az adatok rögzítését, valamint a programrendszerek kifejlesztését Intézetünk végezte.

A DTM-file szerkezetét, annak részletességét azok a feladatok határozzák meg, amelyeken azt felhasználni kívánjuk.

Az általunk kialakított DTM-file felhasználásával különféle feldolgozások végezhetők, pl.

- terepmetszetek, besugárzási-, árnyék-, csillapítástérképek készíthetők,
- műsorszóró hálózatok, rádiótelefon hálózatok tervezhetők,
- hullámterjedési, ellátottsági, zavartartási, jel/zaj számítások végezhetők.

Bár a rendszer elsősorban postai felhasználásra készült, a DTM-file lehetőséget nyújt minden olyan terepadatot igénylő feladat megoldásához, amelyet a rögzített adatok pontossága kielégít.

A DTM-rendszerben eddig a következő alrendszereket fejlesztettük ki:

- A DTM-file-t létrehozó és módosító alrendszer;
- Terepmetszet készítő alrendszer;
- Mikrohullámú összeköttetések nyomvonalainak, állomás telephelyének és antenna magasságának meghatározását végző alrendszer.

A rendszert és részeit az 1. ábra szemlélteti.

2. A DTM-file kialakítása

A DTM-rendszer tervezésénél a felhasználás során megkívánt pontosságon kívül természetesen nem lehetett figyelmen kívül hagyni a rendelkezésre álló számítógépeket (Honeywell-220 R–30, RC–3600) és anyagi eszközöket sem. Ezek figyelembevételével egy hálós szerkezetű, mintaelemes megoldás kialakítására került sor. A terepet egyenlő közű koordináta hálóval lefedve a háló vonalai kb. 200x200 m méretű területelemeket határolnak be.

A felhasználói igények kielégítéséhez a terep tengerszint feletti magasságának adataira, valamint a vizsgált terület felszíni viszonyaira van szükség. Ezek az információk minden egyes területelemre jellemzőek és azokat egy-egy adatszóban (tétel), 6 decimális karakterben fejezzük ki és tároljuk. Balról az első négy karakter a területelem maximális tengerszint feletti magassága

tartalmazza méterben. Az ötödik karakterben a maximális és minimális magasság közötti különbség (D) szerepel kódolt formában, 5 méteres léptékben. (Pl.: H=700, D=6, akkor a területelem legalacsonyabb pontjának meghatározása: 700—5x6=670 m.) A hatodik pozícióban a felszínborítási kód (F) van elhelyezve, amely a felszín borítottságát jelzi (pl.: szántóföld, erdő, város, vízfelület stb.)

A háló méretéből adódó felbontási finomság és a file felhasználásával elérhető pontosság között nyilvánvaló szoros kapcsolat van. Sűrűbb hálóval nagyobb pontosság érhető el, azonban a hálózat sűrítésének határt szab a felhasznált térképek méretaránya.

A térképen levő információk leolvashatósága korlátozott, a térképrajz pontatlanságai és a papíranyag deformálódása miatt, amihez járulnak még a szubjektív tényezők (a leolvasó személy koncentráltsága például).

További szempont a felbontási finomság meghatározásánál a munkaigényesség. A mintaelem mérete és a mintaelemek mennyisége között négyzetes összefüggés van. A háló méretét felezve, a mintaelemek száma négyszeresére nő, a manuális és a gépi feldolgozási idő szintén, ezzel vonva a költség és átfutási idő növekedését is.

E megfontolások alapján határozták meg a DTM egy elemének méretét az említett 200x200 méterben. A térképről való adatfelvétel az 1:25 00 méretarányú koordináta hálós térképszelvények digitalizálásával történt.

A DTM-file 3312000 területelem adatait tartalmazza összesen mintegy 20 millió karakter terjedelemben. 50 területelem adatai egy oldalt, azaz egy rekordot; 720 rekord egy kötetet alkot és a teljes file 92 kötetből áll. Az egy rekordba tartozó tételek olyan területelemek adatait tartalmazzák, amelyek a terepen,,illetve a térképen É—D irányban helyezkednek el egy összefüggő kb. 200 m széles 10 km hosszú sávot képezve. Egy kötetnek pedig egy, az 1:100000 méretarányú térképszelvény felel meg.

Az adatrögzítést lyukszalag-technikával végeztük. Rögzítésre kerültek rekordonként a rekord kulcsai: kötetszám és oldalszám, valamint a rekordnak megfelelő terület délnyugati sarokpontjának földrajzi koordinátái és természetesen a rekord tulajdonképpeni adatait tartalmazó 50 adatszó (tétel). Ezután a rögzített adatokat mágnesszalagon (2 tekercses mágnesszalag file-ként helyeztük el és programokkal ellenőriztük. A rekordkulcsok helyességének és rekordok sértetlenségének az ellenőrzésére felhasználtuk, hogy a rekord kulcsa (kötetszám, oldalszám) éppen úgy pontos topográfiai jellemzését adja a rekordnak, mint a rögzített földrajzi koordinátái, így ezen két adatpár között egyértelmű megfeleltetése lehetséges.

Fontos a rekordban elhelyezkedő magasság-adatok ellenőrzése is. Ezeknek ellenőrzését két irányban végeztük. Az egyik, az É—D irány, a rekordon belüli tételek egymás közötti összhangjának a vizsgálata volt, míg a másik a terepen K—Ny irányban szomszédosan elhelyezkedő 2 rekord azonos földrajzi szélességen elhelyezkedő tételének vizsgálata, összehasonlítása volt.

A szomszédos területelemek illeszkedését — akár É—D, akár K—Ny irányban — együttesen vizsgálva, lehetőség nyílik a hibák jelentős részének automatikus felismerésére. Ugyanis ha egy területelem legalacsonyabb pontja (D) magasabban helyezkedik el a szomszédos területelem legmagasabb (H) pontjánál, akkor a két területelem határvonalán magasságugrás van.

Az ellenőrzés alapjául az a feltevés szolgált, hogy a természetben, különösen Magyarország terepviszonyait tekintve igen valószínűtlen az olyan függőleges falú szakadék előfordulása, amely szakadék pontosan párhuzamos az elemeket elválasztó hálózati vonalakkal és pontosan a határvonalon helyezkedik el. Elfogadva azt a feltételezést, hogy ilyen szakadék előfordulásának valószínűsége kisebb az adathibák valószínűségénél, célszerű volt ezt az ellenőrzési eljárást alkalmazni. A gyakorlati munka során a vázolt feltevés beigazolódott és jelentős mennyiségű hibát sikerült ily módon felszámolnunk.

A számítógépes ellenőrzéseken kívül vizuális ellenőrzést is folytattunk, különös tek arra a körülményre, hogy a fedettségi kód ellenőrzésére nem volt lehetőség korrekt algor szerkeszteni.

A többszörösen ellenőrzött és javított file-t Honeywell-2200 gépünkön egy 2 tekerc mágnesszalag file-on alakítottuk ki, majd azt RC—3600 gépünkön IBM kódba konvertáltu végül két R—30 lemezcsomagon helyeztük el.

A felhasználói szempontok már kezdettől fogva mágneslemezes feldolgozást igényel Igyekeztünk a legcélszerűbb lemezes megoldást megkeresni, amely a legtakarékosabb elhel és a legegyszerűbb hozzáférést biztosítja.

A rekordokat a terepen való elhelyezkedés szerint (kötet emelkedő, földrajzi hossz csökkenő, földrajzi szélesség csökkenő sorrendben) rendeztük. A program az így rendezett rekordjaiból elhagyta a kulcsokat, koordinátákat és csak a tényleges terepadatokat (magas mélység, fedettség) tartotta meg. Egy-egy tétel 6 karakteres adatát binárisra alakította és e 3 byte-os mezőben helyezte el. A program 1/3 kötet (240 rekord) adatait egy mátrixban (200 sorból áll és minden sor 60 mezőt tartalmaz) helyezte el. Ily módon egy kötet (720 re adatai egymás után három mátrixban férnek el. A mátrix 12000 mezőjét a DTM-elemek te való elhelyezkedésének sorrendjében töltöttük fel adatokkal és egy-egy mátrix adatait egy cilinderen helyeztük el. Ily módon egy 36250 byte kapacitású cilinderen 1/3 kötet: 200x60x3=36000 byte mennyiségű adat fért el. A program ezek szerint alakította ki a DTM lemez file-t 276 cilinderen. (Egy cilindernek egy 12 kmx40 km terjedelmű terület felel meg

A továbbiakban a lemezeken levő adatok azonosítása az adatkiemelő programrendsze tal használt táblázat segítségével történik. A táblázat a terület koordinátáit és a vonatkozó ki cilinder címeket tartalmazza. A file-ból az adatokat felhasználásra úgy kapjuk vissza, hogy a adatkiemelést vezérlő program az imput igények alapján meghatározza a kiemelkedő adat ko nátáját. Ezt a programban levő táblázattal összevetve megállapítja a vonatkozó cilinder fizik címét. Az adatkiemelő program a cilinderen tárolt adatokat az operatív memóriában, egy 20 elemes mátrixban helyezi el. A kiszámított koordináták meghatározzák a mátrix sor, oszlop dexét is, így a megfelelő elemek adatait a mátrixból felhasználásra kiemelhetjük. Amennyibe koordináták által meghatározott DTM-elemek adatai a mátrixban már nem találhatók meg, a ram a megfelelő cilinder adatait helyezi el mátrix alakban az operatív memóriában és a kiem abból folytatja.

3. A DTM-file adatait használó alrendszerek

A felhasználó igények alapján megállapíthatók voltak bizonyos tipikus feladatok:

- terepmetszet készítés,
- Fresnel-zóna tisztaságának vizsgálata,
- besugárzási- és árnyéktérkép készítés,
- csillapítás- és térerősség térkép készítés stb.

Ilyen viszonylag egyszerű, tipikus feladatokat végző programokból és megfelelő össze feladatra orientált főprogramokból alrendszerek szerkeszthetők össze, amelyek összetett fela tok elvégzésére alkalmasak. Többek közt:

- összeköttetetés tervezése,
- telephely kijelölése és antenna magasság meghatározása,
- optimális hálózat tervezése.

3.1 *Terepmetszet készítés*

Egy terepmetszet két adott földrajzi pontot összekötő gömbi főköríve a terepvonulat magasságát ábrázolja a távolság függvényében. A feldolgozás elvégzéséhez szükséges magassági és fedettségi adatokat az adatkiemelő programtól kapják a terepmetszet készítő programok.

A terepmetszet készítő program a felhasználási céltól függően meghatározott tényleges földsugárral (K földsugár tényező felhasználásával) megszerkeszti a földfelszín görbültségét, és ezzel korrigálja a DTM-file-ból kapott tengerszintre vonatkoztatott magassági adatokat.

Tekintettel arra, hogy a földsugár változtatásával jó közelítésben modellezni lehet a légköri törésmutató változásait, az időbeli folyamatok statisztikai paramétereit a földsugártényező változtatásával szokás meghatározni. Ezért a programot alkalmassá tettük azonos terepszakasz több különböző megadott földsugár tényező esetén történő vizsgálatára.

A terepmetszet készítése a legalapvetőbb felhasználói program, általában minden hullámterjedési számítás alapja. Az összeköttetés tervezése a terepmetszet felhasználásával történik, a Fresnel-zóna és az átlátási vonal akadály mentességének vizsgálatával folyik. A szokásosan használt földsugártényezők értéke:

- Fresbel-zóna vizsgálatánál K=4/3,
- Átlátási vonal vizsgálatánál K=0,7

Ezek a számértékek a légköri törésmutató gradiens statisztikájának kiemelt pontjait képviselik.

Az átlátási vizsgálatokat végző programot úgy alakítottuk ki, hogy az a terepmetszetet, az átlátási vonalat és a Fresnel-ellipszoid függőleges metszetének alsó vonalát a printerrel egyszerre tudja kinyomtatni, vagy plotterrel megrajzoltatni. Az imput adatként szolgáló felhasználói igényeket lyukkártyákon, vagy SLK–4 mágneskazettán rögzítjük. Ezeket az adatokat egy programmal ellenőrizzük és egy mágnesszalagon helyezzük el. Az esetleges hibák javítása után a mágnesszalagot a terepmetszet rajzoló program dolgozza fel.

3.2 *Összeköttetés tervezése*

Valamely mikrohullámú összeköttetés tervezésének a megkezdésekor rögzítésre kerülnek a tervezett két végpont, az összeköttetés alapvető paraméterei és az összeköttetéssel szemben támasztott minőségi követelmények. A megengedhető legnagyobb szakaszhosszúság ismeretében el kell dönteni, hogy szükséges-e az összeköttetéshez közbenső állomást alkalmazni a végpontok közötti távolság áthidalására. Ha két végpont távolsága nagyobb a megengedett értéknél, a közbenső területen kell egy olyan pontot kiválasztani, amely alkalmas a kívánt minőségű összeköttetés biztosítására és az adott lehetőségek figyelembevételével optimális megoldást nyújt. Az optimalizálás célfüggvényének az antenna, illetve az antennatornyok magasságát szoktuk tekinteni, mivel a költségek jelentős részét ez adja meg. Ezért a figyelembe vehető területet a DTM-elemek által meghatározott finomságú bontásban végigvizsgáljuk, szakaszokat képzünk, meghatározzuk a szakaszok terepmetszetét és azok alapján kiszámítjuk az antenna magasságokat.

A vázolt műszaki probléma megoldására „Mikrohullámú összeköttetések nyomvonalának, állomás telephelyének és antenna magasságának meghatározása" elnevezésű alrendszert alakítottunk ki. Az alrendszer egy előkészítő (file kialakító) és a feldolgozást végző programokból áll. Az előkészítő program a rögzített felhasználói igényeket egy lemez file-ben – melyet MOZAIK-file-nak nevezünk – helyezi el. A MOZAIK-file-ban előjegyzésre kerül:

- a köztes terület 3–8 ponttal (köztes terület, melyen az összeköttetés létesítésére alkalmas telephelyet a feldolgozó programoknak keresniök kell),

– a köztes területen belül a felhasználó által letiltott helyek vagy területek,
– 2–5 fix pont, amelyeken a felhasználó által meghatározott antennák vannak (ezekkel k
összeköttetést keresni a kijelölt köztes területen),
– a fix pontokon elhelyezkedő antennák magassága,
– a köztes területen megengedett maximális antenna magasság,
– az összeköttetés frekvenciája,
– K-tényező értéke,
– az opcionális lehetőségek igénybevétele.

A feldolgozó programot a következő programokból szerkesztjük össze:
– egy feladat orientált főprogram,
– adatkiemelést végző alprogramok,
– magasságmódosító alprogram,
– Fresnel-ellipszoid számító alprogram,
– antenna magasság számító alprogram,
– MOZAIK térkép-nyomtató alprogram.

A felsoroltakból összeszerkesztett program a feldolgozás során egy egységes programként működik. Imputként felhasználja a DTM-file-t, valamint a MOZAIK-file-ban elhelyezett adatok

Mint említettük, a file-ban el van helyezve maximum 5 fix-pontnak a koordinátája és meg van határozva egy köztes terület. A program szakaszokat jelöl ki egy-egy fix-pont és a köztes terület mindegyik mezője között. Igy a köztes terület mindegyik elemére annyi szakaszt képez, ahány fix-pontot határozott meg a felhasználó.

A kijelölt szakaszokkal a program a továbbiakban egyenként foglalkozik. A program a szakasz minden elemének kiszámítja a koordinátáit, majd ebből a fizikai címeket (cilinder-index, mátrix sorindex, oszlopindex) a kiszámított kulcsok alapján a szakasz minden elemének terepadatait a DTM–2---file-ból. Ehhez a program a vonatkozó cilinder terljes tartalmát az operatív memória már ismertetett 200x60 mező méretű mátrixba olvassa be, majd az adatkiemelést a sorindex és oszlopindex felhasználásával hajtja végre, a megfelelő mezőben a kulcsokat a kiemelt magasság-adatokkal cserélve fel. Igy egy szakasz magasság-adatai az operatív memória megfelelő tömbjében rendelkezésre állnak. A továbbiakban a program a tengerszint feletti magasságot a föld görbületnek megfelelően módosítja. A módosítás a MOZAIK–file-ben előjegyzett adatok alapján a felhasználó aktuális igénye szerint történik. A program ezután az összeköttetés minőségére előírt követelmények alapján, vagy a Fresnel-ellipszoid, vagy az átlátási vonal akadálymentességét vizsgálja. Meghatározza az ezt kielégítő antennamagasságokat minden szakaszra külön-külön. Az egy területelemre kiszámított antennamagasságok közül a legkedvezőtlenebbet veszi figyelembe. Amennyiben az inputként megadott megengedhető maximális antennamagasságot ez az érték meghaladja, a területelemet összeköttetés létesítésére alkalmatlannak nyilvánítja.

A MOZAIK-file egy e célra kijelölt mátrix részének elemeiben gyűlnek össze a kiszámított antennamagasságok. A teljes mátrix kialakulása után annak tartalmát kódolva, a magasság-adatokat bizonyos határok között egy karakterrel jelölve, az alkalmatlannak minősített helyeket egy eltérő karakterrel megjelölve, sornyomtató segítségével végső eredményként egy MOZAIK-térképet alakít ki a program.

4. Befejezés

Nagyvonalakban ismertettük a DTM-rendszert, és annak eddig elkészült 3 alrendszerét. További alrendszerek kialakítását is tervezzük, többek közt a közeljövő feladatait:
– Reflexiómentes területek meghatározása;
– Akadályos hullámterjedés esetén kialakuló térerősség számítása;
– URH rádiótelefon besugárzás számítás.

Természetesen ezzel nincs lezárva az alrendszerek sorozata, minthogy eddig is számos más ...tai igény jelentkezett, és természetesen a DTM-rendszer nem csak postai, hanem sok postán ...üli feladatban is alkalmazható volna.

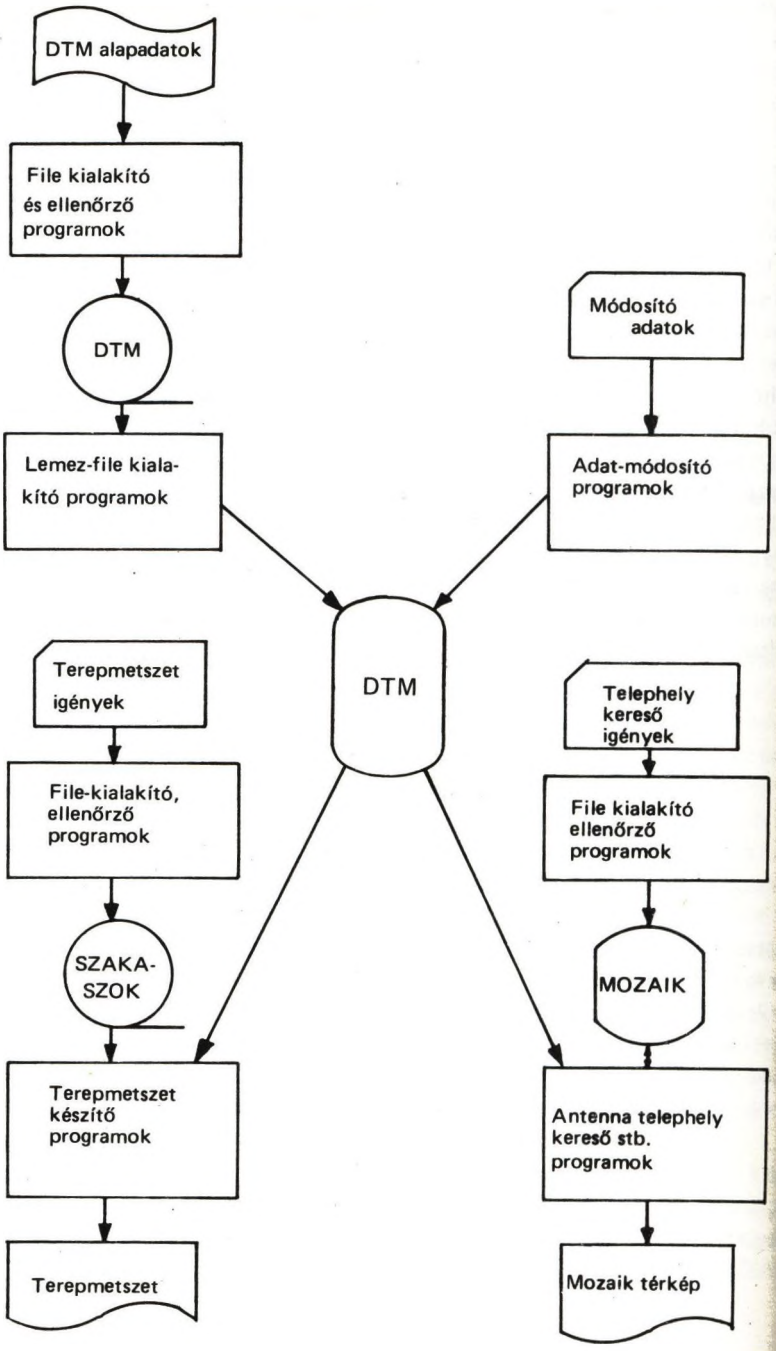
A kialakított DTM-rendszerrel a vezetéknélküli távközlés területén a műszaki tervezés ...y hatékony eszközt kapott. A manuális tervezés során csak egyszerűsített módszer alkalmazható, ...elyet a mérnöki tapasztalat és a mérési eredményekből levonható következtetések támogatnak. ...tervezési határidő betartása érdekében a telepítésre alkalmas terület megfelelőnek becsült pont...ára, minimális számú paraméter figyelembevételével egyszerűbb matematikai összefüggések fel...használásával történik a vizsgálat. A számítások alapjául szolgáló terepmetszetek készítése gé...es és időigényes feladat. Ha ezt külső céggel végezteti a tervező, a szerződéskötés az átfutási ...időt lényegesen megnöveli, ezért általában saját dolgozói (kvalifikált munkaerők) készítik a te...repmetszeteket is, ami nem nevezhető gazdaságosnak.

A számítógépes rendszert igénybevevő tervezésnél az összes lényeges paraméter figyelembe ...vető, bonyolultabb matematikai összefüggéseken alapuló, tehát nagyobb pontosságot nyújt...ó tervezési módszer alkalmazható. Lehetőség nyílik arra, hogy ne csak a szubjektív megítélés ...apján kiválasztott pontokra történjen a vizsgálat, hanem az összes lehetőséget végig vizsgálva, ...legjobb eredmény alapján történjen a tervezés.

A DTM-rendszerhez kapcsolódó feldolgozási igények gyűjtését, összehangolását, további ...rendszerek tervezését a PKI végzi.

Előadásunk anyagának összeállításánál felhasználtuk Koós Árpád, a PKI tudományos fő...munkatársa által az e témában készített tanulmányokat.

DTM rendszer



1. ábra

R–10 ALAPU HÁLÓZATI KOMMUNIKÁCIÓS PROCESSZOR

Ercsényi András
MTA SZTAKI

0. Bevezetés

Az ESZR-távfeldolgozás egyik nagy gondja – hazánkban és a többi szocialista országban is – a multiplexerek kérdése. Néhány évvel ezelőtt a VIDEOTON megjelentetett egy VT 55000 jelzésű, R–10 alapú programozott multiplexert. A berendezés kielégítően működik ugyan – pl. a VEIKI R40 központú terminálrendszerében – az újabb igények, a növekvő szolgáltatási és üzembiztonsági követelmények azonban továbblépést igényelnek.

A SZTAKI és a VEIKI is résztvett a VT 55000 kifejlesztésében, munkatársai sok ismeretet szereztek e területen. Ez, párosítva a VEIKI terminál-üzemeltetői tapasztalatával – amely ESZR vonalon messze a legjelentősebb az országban – elegendőnek bizonyult a multiplexer továbbfejlesztése feladatának sikeres megoldásához.

Az előadás az újonnan kifejlesztett R–10-alapú kommunikációs processzor általános ismertetését adja.

1. Új igények – továbbfejlesztési célok

VT 55000 jelzésű, R–10 alapú multiplexer több helyen működik az országban. A SZTAKI-ban is van egy olyan – fejlesztési – R–10 konfiguráció, mely multiplexerként is dolgozhat. Szinkron és aszinkron terminálok egyaránt kapcsolódhatnak rá, teljesen egyidejűleg működve. Üzemszerűen ugyan még nem indult be, de RJE illetve ORION-szövegszerkesztő – aszinkron terminálokon az utóbbival – próbafuttatások folynak velük.

Innen – és más helyekről szerzett – tapasztalatok alapján rendszerezve, általában fontossági sorrendben leírva, a legszükségesebb fejlesztési kérdések az alábbiak voltak:

(1) *Üzemeltetés biztonságának jobb megoldása:* ez a multiplexer, mint rendszer állapotának lekérdezését, szisztematikus hibakezelést, működési statisztikák készítését, stb. jelenti.

(2) *Gazdaságosabb memóriakezelés;* azaz áttérés a fix buffergazdálkodásról a dinamikusra, ami rögtön a terminálszám növekedését is eredményezi.

(3) *Logikai vonalkezelés kialakítása:* így megszüntethető a multipoint vonalak fizikai szintű kezelése a magasabb szinteken.

(4) *Bővíthetőség biztosítása;* ez flexibilisebb konfigurálhatóságot ad, on-line R–10-periféria terminálként történő használatát eredményezi.

Mindezekhez néhány olyan fejlesztési célkitűzés is járult, amely már a számítógéphálózatokkal kapcsolatban lesz fontos.

Így,

(5) *Dinamikus kapcsolaftelépítés;* ezzel lehetőség nyílik a fix alcsatornacím-adatvonal hozzárendelés helyett tetszőleges helyre történő bejelentkezésre.

(6) *Illeszkedés számítógéphálózathoz;* ez általánosan a terminál működtetés és az adatátvitel szétválasztását jelenti.

A fentiek miatt a VT 55000 funkcionális rendszerétől néhány jelentősebb eltérést kellett végrehajtani, program-technikailag pedig teljesen új alapokon kellett elindulni. Az utóbbit

indokolta az is, hogy a fejlesztést az R—10 IDOS rendszerében végeztük, ami nem kompa lis R—10/OS-szel. Az előadás utolsó fejezetében az áttérés biztosította rendkívül előnyös p ramfejlesztői környezetről is lesz egy rövid ismertetés. (Bár újraírási munkák jelentkeztek, VT 55000 fejlesztésében szerzett tapasztalatokat hasznosan lehetett kihasználni.)

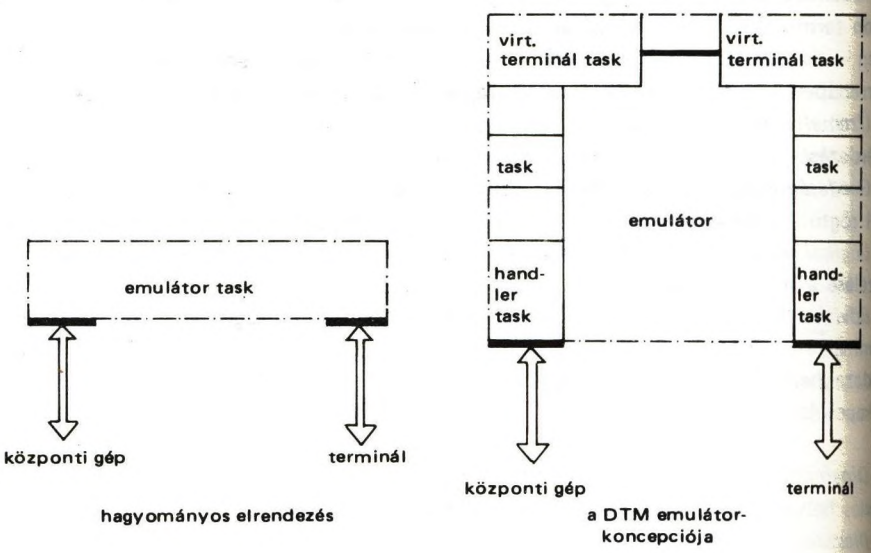
2. Az új miltiplexer felépítése
(Általános rendszertechnikai meggondolások)

A multiplexer-feladatok megoldása tipikusan multitask feladat. Így az új miltiplexer rendszertechnikailag a task-koncepció alapján épül fel.

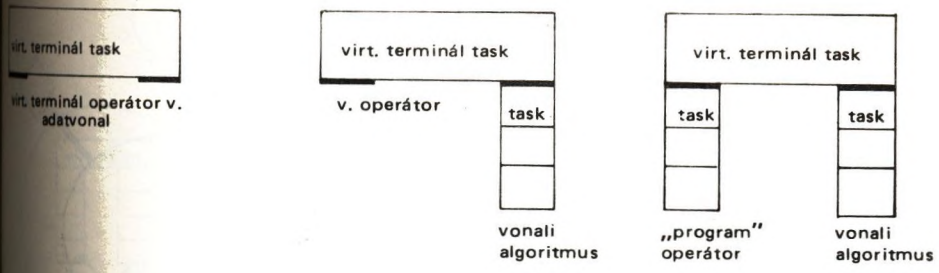
Elődjétől eltérően azonban nem egysíkú, hanem hierarchikusan egymásra épülő elem ből áll. Az elemek szabványos módon kommunikálnak egymással.

A task-hierarchiák — mert több független ilyen van — legalsó szintjein az ún. handler taskok vannak. A handler-task típus a szuperhandler-típusú átvitel-vezérlők általánosítása. (eredeti ötlet egy VIDEOTON software-ben található.) A szuperhandler egy olyan „csak vá vezérlő, azaz fizikai átvitel-indító, elemző, melyben az adott vonalra a konkrét műveleteke egy formálisan zárt, nem vezérlő jellegű program végzi. Ez úgy történik, hogy a szuperhan ler táblázatosan ismeri ezeket a „programokat", és a megfelelő időben „kikapcsol" azokra. Rendszerünkben ennek olyan általánosítása történt meg, mely szerint ezen programok task ként fogalmazódnak meg, és aktiválásuk is — ezen task szempontjából — standard módon tén ik.

A rendszerben több task-csoport van, mivel újszerű módon a multiplexer csatornaoldali és adatvonal-oldali taskjait különválasztottuk. Mindegyik hierarchia-csúcsára olyan elem illeszkedik, melyekkel a függetlenül dolgozó csoportok összekapcsolhatóak; ezeket az elemeket virtuális terminálóknak nevezzük. Így a hagyományos emulátor koncepciót „virtuális terminálokba konvertáló félemulátorok"-módszere váltotta fel. (1. ábra.)

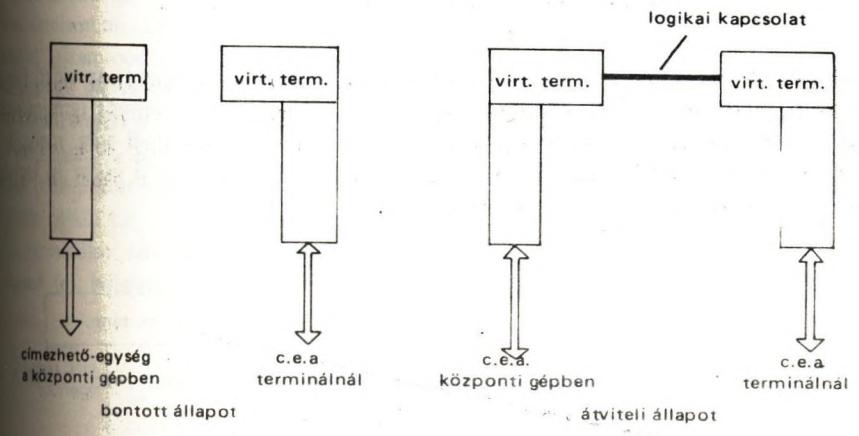


1. ábra Az emulátor-koncepció módosítása



2. ábra Virtuális terminál task-realizálása

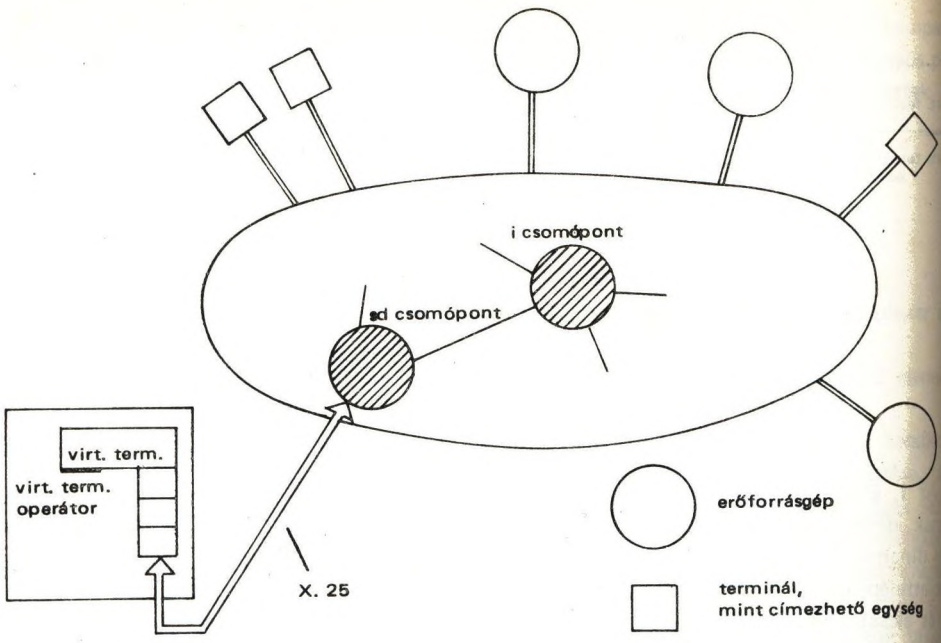
Az 1. ábra azonban nem teljesen adja vissza a logikai struktúrát. Ugyanis az összekötő csak időlegesen, a kapcsolat-felépítés után van jelen, mivel a céljaink alapján nem fix, hanem dinamikus kapcsolat alakítást vezettünk be. Ez azt jelenti, hogy megkülönböztetünk kapcsolatfelépítési és átviteli fázist. (3. ábra) Továbblépve, a felépítési fázisban nem adatvonalak és alcsatornák, hanem a központi gép ill. a terminál címezhető egységei kapcsolódhatnak. Pl. a központi gép különböző távfeldolgozó programjai címezhető egységek lehetnek.



3. ábra A dinamikus kapcsolatkialakítás

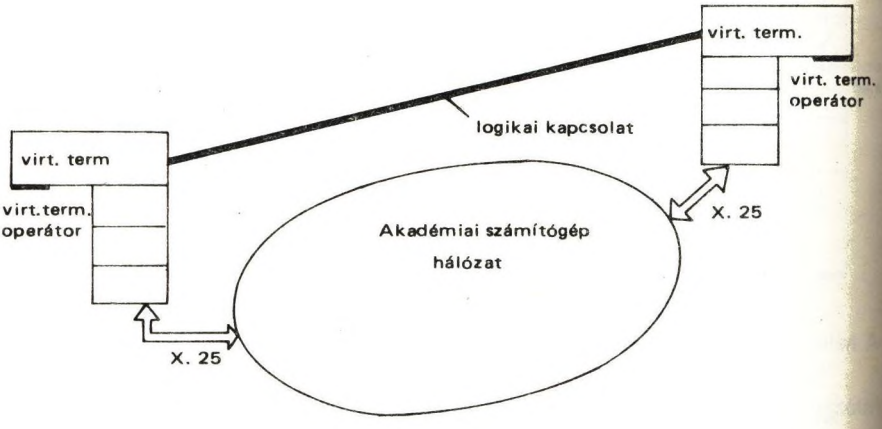
3. A hálózati illeszkedés biztosítása

A hálózati illeszkedés általános kérdése konkrétan az Akadémiai Számítógép Hálózattal kapcsolatban merült fel. Ez a hálózat – kísérleti célokkal ugyan, de – biztosítja majd, hogy egy – akadémiai – felhasználó könnyen, egyetlen terminál mellől elérje az akadémiai intézetek üzemelő számítógépek bármelyikének tetszőleges távfeldolgozói szolgáltatását. A hálózatból a felhasználó X.25 típusú kimeneten egy jól definiált „virtuális terminál"-ként létszik. (4. ábra)

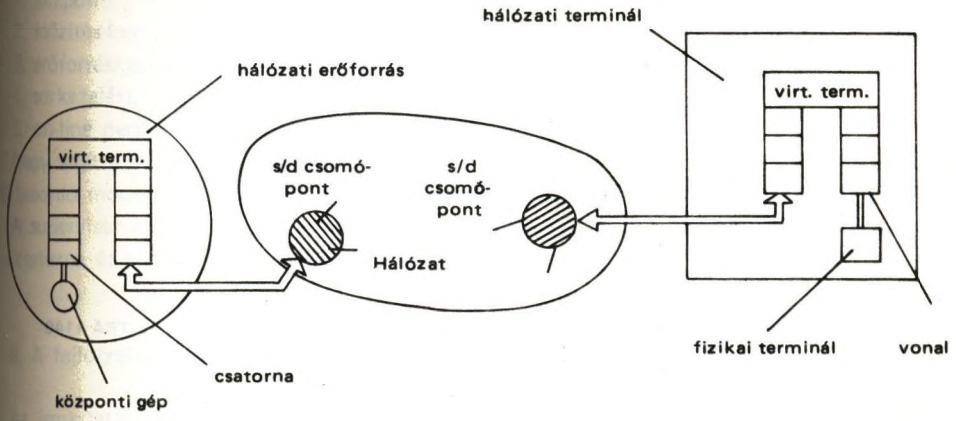


4. ábra Az Akadémiai Számítógép Hálózat

A hálózati illesztés tehát X. 25 „algoritmusú" terminál kezelését jelenti az előző fejezetben ismertetett rendszerben. Egy központi gép-terminál jellegű hagyományos terminálkapcsolatnál a gép oldali hálózati illesztését egy olyan virtuális hálózati terminál adja, melynek „operátora" a csatorna-taskok együttese – implicite beleértve a központi gépet is. Ugyanígy a terminál oldal is értelmezhető. (5., 6. ábra)



5. ábra Címezhető egységek kapcsolódása a hálózaton



6. ábra Gép-terminál kapcsolat megvalósítása

Az előzőek alapján a hálózati illesztést is adó multiplexert három független taskcsoport alkotja, melyből kettő-kettő a felépülési fázisban virtuális terminál operátor, illetve adatkimenetként hagyományos multiplexálást, az erőforrásgép illetve a terminálok hálózatra kapcsolását biztosítja.

Ilyen értelemben már nem beszélhetünk a „központi gép termináljairól", illetve csak dinamikus értelemben. Ezért a terminálvezérlő állapotában meg is különböztetünk „hálózati" és „kapcsolat" üzemmódot. Az előbbiekben a terminál egyetlen, a hálózaton kialakítható terminálrendszernek illetve terminál-terminálkapcsolatnak sem tagja.

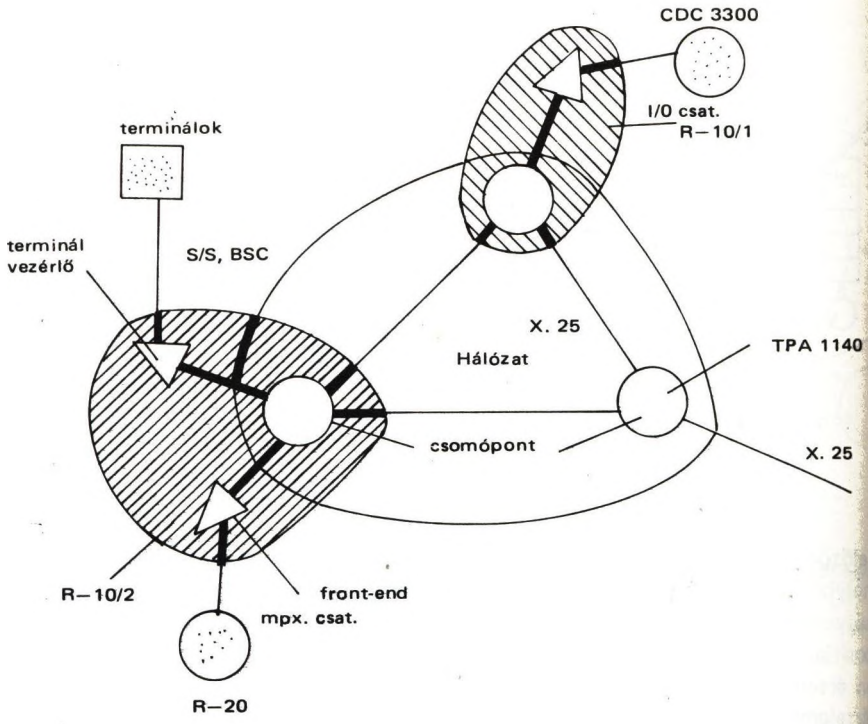
Természetesen a központi gép- és a terminál-oldal intelligens működést biztosító elemeket is tartalmaz, és így multiplexerként tovább tehermentesíti a központi erőforrásgépet. Azonban multiplexer-jellegét nem változtathatjuk meg, mivel a központi gépek rendszere belső változtatás nélkül ezt nem teszi lehetővé.

A terminálok hálózati terminállá válása azok kezelésében bizonyos módosításokat tett szükségessé (pl. bejelentkezés, azonosítás, stb.). Erre egy egyszerű konvenciót definiáltunk.

4. R–10, mint többfunkciós kommunikációs processzor

Az eddigiekben az R–10 alapú multiplexer továbbfejlesztéséről volt szó. A multipexer-fejlesztés tulajdonképpen melléktermékeként annak az intézeti fejlesztésnek, melynek célja az Akadémiai Számítógép Hálózat kialakítása. Az ASZH csomagkapcsolásos technikával dolgozik, kezdetben a független csomagok, a módosított terv szerint a virtuális kapcsolatok elve szerint működve. (Az utóbbi alapján csak a kapcsolatkialakulás fázisában van önálló útvonalválasztás, a felépülés ezen út fixálását jelenti az összes többi csomag számára egészen a bontásig.)

A kísérleti hálózati konfiguráció három csomópontú (7. ábra). Benne két R–10-gép, mint csomóponti gépek a CDC 3300 illetve egy R–20 előtt. Az R–20 előtt még az a multiplexer-R–10 is van, mely funkcionális működését az előző fejezetekben már ismertettük. A két egymással összeköttetésben álló R–10 valójában egyetlen berendezés, függetlenül működő software-rendszerrel. Így a multiplexer és a csomópont közötti kapcsolat fizikailag a gépen belül realizálódik.



7. ábra Az Akadémiai Számítógép Hálózat alapkonfigurációja

A virtuális áramkörök elvén működő adathálózat belső csomópontok közötti protokoll-ja is X. 25 szerint van definiálva, így minimális software igényt jelent ez a multiplexer-funkcionáláshoz képest. Elkészült még a független csomagok elvén működő datagramm típusú csomóponti egység software is, ez azonban nem került még kipróbálásra.

5. Alapsoftware a kommunikációs processzorhoz (DTM)

A feladathoz önállóan kifejlesztettünk egy multitask supervisort. (Az új fejlesztést a már említett okok indokolták.) A supervisor nem általános, hanem célorientáltan a feladathoz készült. A rendszer csak a memóriában dolgozik, mágneslemezt nem használ.

Kihasználva a gép hardware lehetőségeit a taskok bizonyos csoportját prioritásosztályba sorolva egy tetszőleges megszakítási szintre lehet kapcsolni. Az így meghatározott csoporton belül a kváziparalel futást egy újrahívható módon, tehát más szinteken is hasonlóan működő ütemező irányítja. A prioritás csoportok természetesen függetlenek a taskok funkcionális csoportosításától. A prioritáscsoportok közti kváziparalelitást a mikrogép vezérli. Az egy prioritáscsoportba tartozó taskok egy IDOS-assembler modult alkotnak. A modulokból a szerkesztő alakítja ki a rendszert a megfelelő cím-abszolutizálások elvégzése után.

A taskok egymásnak adatokat közös zónán keresztül adhatnak át egymásnak. A szinkronozódást supervisor szolgáltatások teszik lehetővé; a rendszerben definiálhatóak események. Ezekre „és" vagy „vagy" kapcsolatban lehet várakozni. Esemény lehet egy időzítés lejárta is

Ezen belül a rendszer biztosít:

1. központi memóriagazdálkodást fix hosszúságú, láncolt bufferekkel,
2. időzítés-kezelést,
3. erőforráskezelést,
4. sorkezelést,
5. on-line periféria-használatot.

Nagy segítséget ad a belövéshez a nyomkövető modul, amely a DTM futása alatt az IDOS hasonló moduljával egyenértékű, sőt fejlettebb szolgáltatásokat is nyújt.

A supervisor kb. 6 kbyte hosszúságú tárterületet foglal le a memóriából. Ebben a részben a csatorna és vonal oldal superhandlerek is bent vannak.

6. A fejlesztési módszer

Az ismertetett kommunikációs processzor-software a programmennyiséget és a minőséget – azaz a jelleget — tekintve már nagyobb feladatnak számít. Ez azt jelenti, hogy

1. a funkcionális tervezés illetve a rendszerkészítés, annak különböző konkrét és kebés-bé konkrét fázisaiban nagy tervezői-programozói figyelmet — és persze figyelmet is — igényel,
2. különös jelentősége van az alaprendszer, illetve a részelemek működése elvi problémái tisztázásának,
3. alapvetően fontos az egyes task-programok, mint önálló egységek belső, rendszer-technikai tisztasága,
4. igen fontos a megfelelő eszköz a már megírt programok alapos kipróbálására,
5. rendkívül fontos az együttműködés ellenőrzése lehetőségének kellő biztosítása,
6. elengedhetetlen a részletes, megfelelően illusztrált dokumentáció készítése és aktualizálása.

A feitneknek megfelelően — dokumentatív jellegét is kihasználva — az alapműködések leírását magasszintű nyelven készítettük el először. A realizálásban ez volt az elsődleges útmutató. Ebben a munkában óriási segítséget jelentett Böszörményi Lászlónak, a VEIKI tudományos munkatársának nagy elméleti felkészültsége.

A kész taskokat egyenként, mint IDOS-programokat, az IDOS felügyelete alatt próbálták ki a modullá szervezés előtt. Az összeszerkesztett rendszer a már említett nyomkövetővel igen kényelmesen tesztelhető. (Ezt a program egységet Pollák Tibor, a SZTAKI tudományos munkatársa tervezte és fejlesztette ki, az IDOS-ban újnak számító szerkesztőprogram elkészítése mellett.)

7. Összegezés

Az előadás az R—10 alapú kommunikációs processzor részletes ismertetésére nem térhetett ki; sok más fontos kérdést is kellett menet közben még megoldani. Általában ezek mindegyike – és az egész is — olyan jellegű, hogy önálló tanulmányozásuk sok érdekes és értékes információt adhat a további munkákhoz.

Az eddigi munkát ugyanis nem tekintjük lezártnak, bár reméljük, hogy a befektetett sok emberi energia egy hasznos eszköz kifejlesztéséhez vezetett.

HŐSZOLGÁLTATÁS ÉS KAPCSOLT VILLAMOSENERGIATERMELÉS KÖZÉPTÁVÚ SZÁMÍTÓGÉPES TERMELÉSTERVEZŐ RENDSZERE

Fekete István—Pallinger Ferenc—Schubert Tamás
BÁNKI DONÁT GÉPIPATI MŰSZAKI FŐISKOLA

1. A hazai hőszolgáltatás fejlődése

A hazai villamos erőművek hőt több, mint 20 éve szolgáltatnak, mégpedig különböző nyomásfokozatokban gőz, illetve forróvíz, mint energiahordozó útján. A hőszolgáltatás kezdeti, lassúbb mennyiségi növekedése az 1960-as években ugrásszerűen gyorsult, majd a gyorsulás 1968-ban, amikor az új gazdasági mechanizmusban a hőszolgáltatás központilag pénzügyi támogatásban már nem részesítették, mérséklődött. A Magyar Villamosművek Tröszt azonban az érdekelt hőfogyasztókkal együttesen azóta is további beruházásokat eszközöl, és amint az alábbi táblázat tény - (az utolsó oszlopban terv) számai tanúsítják, a fejlesztést tovább folytatják.

	1960	1965	1970	1975	1980
kiadott hő (Tcal)	1194	3748	7560	10470	15000
összes hőszükséglet (Tcal)	25504	37225	48321	63985	84656
kiadott hő hőszükséglete (Tcal)	1774	5332	10253	13520	19344
(az összes hőszükséglet %-ában)	7,0	14,2	21,4	21,1	22,9

A villamosenergia-termeléssel kapcsolt hőszolgáltatás igazi haszna a vele együttjáró tüzelőanyag-megtakarítás. Ugyanis a turbinákban a generátorok villamosenergiatermelésére már használt gőzt a hőfogyasztóknak fűtésre vagy más ipari célra eladják, és így az utóbbi hőigények újabb tüzelőanyag felhasználása nélkül elégíthetők ki.

A hőszolgáltatás abszolút értékének növekedése mellett — a tervezés jelentőségének szempontjából — különösen figyelemre méltó a kiadott hő hőszükséglete részarányának növekedése az összes hőszükségletből.

A hőszolgáltatást az 1950-es években a régi, kondenzációs erőművekből kezdték, amelyet nagy fajlagos hőfogyasztásuk miatt villamosenergiatermelésre már egyre kevésbé vettek igénybe. Ezen erőművek mellé fokozatosan léptek be a már hőszolgáltatásra tervezett erőművek az úgynevezett ellenyomású program keretében. Ma a hőszolgáltatásban mind kondenzációs végű, mind ellennyomású turbinák részt vesznek. Az egyes turbinatípusok és a hozzájuk rendelt kazántípusok részarányát, illetve az új beruházásokban megvalósuló részesedésüket gazdaságossági és környezetvédelmi szempontokból határozzák meg. Az elmúlt 20 év statisztikája az ellennyomású és az ellennyomású-elvételes turbinák részarányának növekedését mutatja.

2. A termelés tervezése

A hőszolgáltatási terveket 1975-ig az erőművek intézkedési és rövidtávú terveire alapították és a tervezés lényegében az erőművi tervek összegezését és a fejlesztésre vonatkozó műszaki becsléseket jelentette. A becslésekben sem több tervváltozat, sem a feltételezett termelés energiafogyasztási és erőművi kapacitáskihasználási hatásainak részletes számítása nem szerepelt.

A becslésekből álló tervezés megengedhető volt, amíg a hőszolgáltatás a villamos energiaszolgáltatás mellett országos összesítésben mennyiségileg számottevő szerepet nem játszott, illetve az energiagazdálkodás mérlegét jelentősen nem befolyásolta.

Miután azonban az utóbbi években az erőművek hőszolgáltatásának mennyisége és ezen keresztül a kapacitásokra és a tüzelőanyag-gazdálkodásra gyakorolt hatása jelentősen megnőtt, a hőszolgáltatás távlati tervezése is nagyobb alaposságot és a mindenkori lehetőségeket kihasználó, biztosabb előrelátást követel.

3. A számítógépes termeléstervező rendszer

A szerzők az utóbbi három évben a középtávú, mintegy 10—15 évre előretekintő tervezés számára egy számítógépes programrendszert szerveztek és fejlesztettek ki. Ez a Magyar Villamos Művek Tröszt Termelési Igazgatóságának Műszaki Adatbankjára épül és a hőszolgáltatást és kapcsolt villamosenergiatermelést a tervező által választott erőművekre és időtartamra számítani, összegezni és a termeléshez szükséges energiafelhasználást is tervezni képes.

A teljes tervező rendszert(1. ábra) nyolc főprogram alkotja, amely közül kettő a statisztikai alapadatrendszer, három a tervezési alapadatrendszer és három a szűkebb értelemben vett tervezés szféráját alkotja. Az első két szférában végzett feldolgozás a tulajdonképpeni tervezés előkészítésének tekinthető.

A tervezés még további két lépcsőben, a hőszolgáltatási terv készítésében és a hőszolgáltatással már meghatározott villamosenergiatermelés számításában valósul meg. Ez utóbbi termelésszámítási lépcső tartalmazza a tervezett termelési eredmények és energiafelhasználás havonkénti és évenkénti összegezését is.

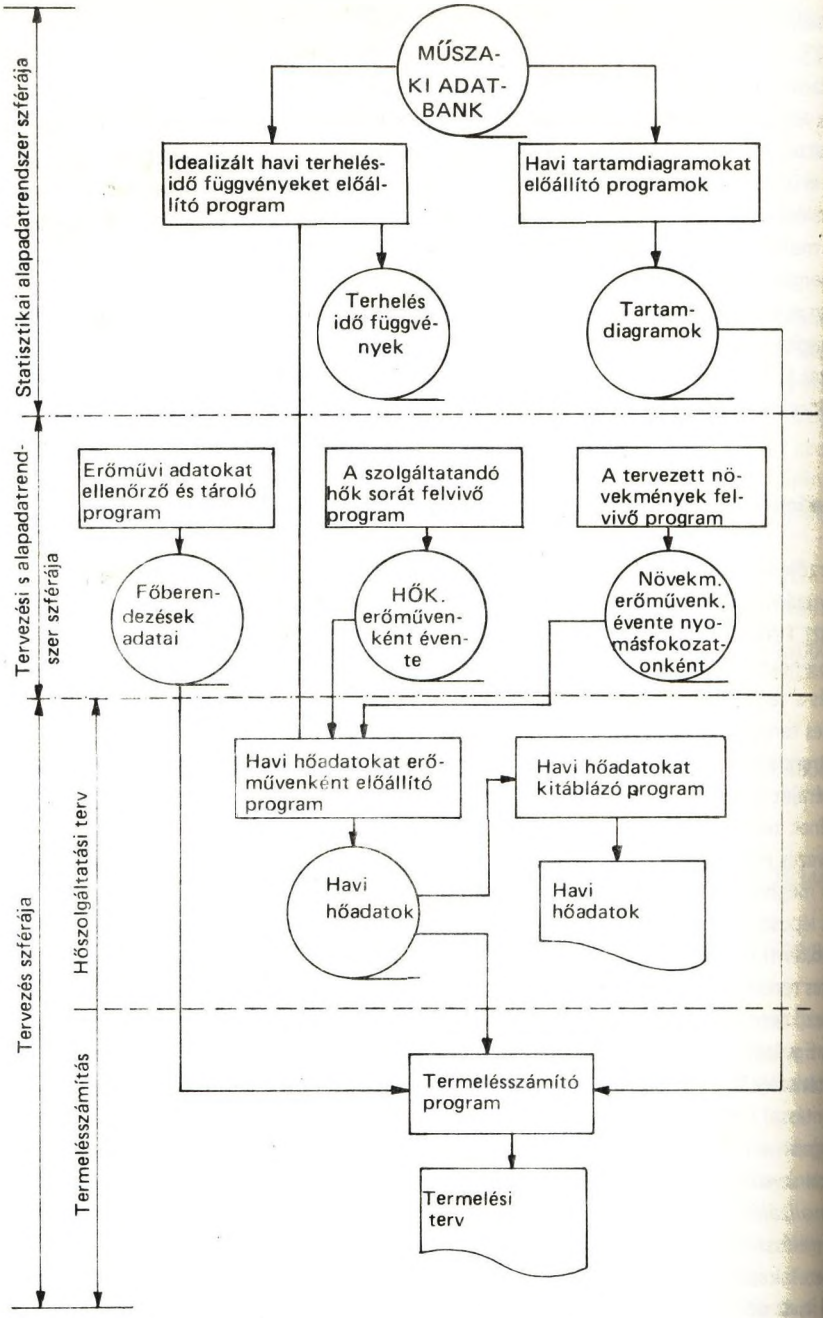
A teljes rendszer legterjedelmesebb és legtőbb eljárást tartalmazó főprogramja az utolsó lépcsőt képező termelésszámító program, amely a tervező által kért erőművekre és évekre tervezi a termelést. Egy-egy erőműre havonként és nyomásfokozatonként tervezett szolgáltatandó hőt – a tárolt statisztikák alapján — napi igényekre is képes bontani és így akár hónapokra, akár napokra bontással képes tervezni.

A program a tervezett havi vagy napi hőmennyiséget az erőmű fő berendezéseire úgy osztja el, hogy azzal az erőmű *a lehető legtöbb villamosenergiát* termelje, azaz a program a tervezéssel együtt *optimalizálási feladatot* is megold.

A termelésszámításhoz a program megkapja az erőmű éppen aktuális felépítését (fő berendezések és ezek kapcsolatát leíró gráf). A villamosenergia maximalizálását a program kétféle algoritmussal is, az ún. logikai szintszámos és egy lineáris programozási feladat megoldásával is elvégzi. A kétféle módszer a termeléstervezési feladat két különböző leírásának felel meg.

A logikai szintszámos módszer esetén a program a maximális villamosenergia-termelést tartja elsődlegesnek és csak ezen belül igyekszik a hőfogyasztók igényeit maradéktalanul kielégíteni. A lineáris program elsődleges célja viszont a hőfogyasztók maradéktalan kielégítése és csak másodlagos cél a maximális villamosenergia-termelés.

A logikai szintszámos eljárás a hagyományos erőművi terhelés kiosztására épül, amely szerint egy-egy hőfogyasztói igényt első-, másod-, . . .n-ed sorban, az ún. *szintszámok* növekvő sorrendjé-



1. ábra

A rendszer adatállománya és főprogramjai

en arról a turbinafokozatról vagy csúcskazánról, illetve hőgyűjtősínről elégítünk ki, amelynek még szabad kapacitása van. Ez a hagyományos terheléskiosztás azonban a kapcsolt villamosenergiatermelés maximumát nem minden esetben biztosítja. Ezért ezt a terheléskiosztást egy *optimalizáló* eljárással egészítettünk ki.

A másik eljárás a maximális villamosenergiatermelést biztosító hőenergiautak meghatározását egy lineáris programozási feladat megoldásával végzi. Ehhez a SIMPLEX-nek nevezett eljáráshoz kidolgoztuk a feladat a paramétereit az adott erőmű alapadatrendszeréből *felépítő* SKETAF eljárást. Ezen termelésszámító programrendszer futását a tervező matematikai előkészítő beavatkozása – célfüggvény és korlátozó feltételek megfogalmazása – nélkül teszi lehetővé.

Mindkét termelésszámító eljárás az erőmű fő berendezéseit és kapcsolataikat feltüntető gráf, mint rendszertechnikai segédlet alapján épül fel. A 2. ábrán egyik hőszolgáltató erőmű gráfja látható.

A fő berendezéseket a gráfpontok, az energiautakat a gráfélek jelképezik. A fő berendezéseket háromjegyű szám jellemzi. Az első számjegy a berendezés fajtáját, a következő kettő pedig az erőműbeli helyszámát jelöli. A százasok jelentése:

100: kazán
200: hőgyűjtősín; gőz vagy forróvízelosztó csomópont a kazánok és a fogyasztók között; erről a pontról nézve a turbinák is fogyasztók, akár a közvetlen hőfogyasztók
300: első turbinafokozat, a legnagyobb nyomású gőzelvétel helye
400: második turbinafokozat, a középnyomású gőzelvétel helye
500: harmadik turbinafokozat, a legkisebb nyomású gőzelvétel helye
600: a kondencáziós turbinák kondenzátor fokozata
700: hőfogyasztócsoport; csak rendszertechnikai szempontból „fő berendezés", mert e gráfpontok valóságbeli megfelelői az erőművön kívüli hőfogyasztók nyomásfokozatonkénti teljesítmény-összegei; így a második két számjegy itt nem erőműbeli helyszámot jelent, hanem a nyomásfokozatot jellemzi
800: villamos generátor

A hőfogyasztók csoportokba sorolása nyomásfokozatonként:

701: 40 ata-nál nagyobb nyomású gőzt fogyasztók
702: 25 ata-nál nagyobb, 40 ata-nál kisebb nyomású gőzt fogyasztók
703: 7 atanál nagyobb, 25 ata-nál kisebb nyomású gőzt fogyasztók
704: 7 ata-nál kisebb nyomású gőzt fogyasztók
705: 150 C -nál forróbb vizet fogyasztók
706: 150 C -nál kevésbé forró vizet fogyasztók

A gráf amellett, hogy a fő berendezéseket jelképezi, helyettesítő vázlat jellegű is. A villamosenergiatermelést ugyanis olyan gráfélek jellemzik, amelyeknek valóságos vezeték nem felel meg. Pl.: a 2. ábra 301–801 gráféle azt a villamos teljesítményt „szállítja", amelyet a turbina a 301 kódú elvételi ponton távozó gőz előzetes entropiaesése révén termel; hasonló szerepet játszik a 401–801 gráfél is, jóllehet a 301–401 kódú turbina a 801 kódú generátort a valóságban természetesen egyetlen tengelykapcsolattal hajtja.

A teljes tervező programrendszer, benne a két optimalizáló eljárás rendelkezésére közös alapadatstruktúra-tömb áll, amely tartalmazza:

a) Nyomásfokozatonként a hőfogyasztókat ellátó tápvezetékek – gráfélek – darabszámát, megterhelésük sorrendjét – szintszámát – és kódjukat.

b) A gráf további csomópontjait is – a 100-as kódszámú kazánok kivételével – fogyasztóknak tekintve, rájuk nézve ugyanazokat az adatokat, amelyeket az a) pontban rögzítettünk.

c) A gráf csomópontjait – a 600-as kódszámú kondenzátorok, a 700-as kódszámú hőfogyasztók és 800-as kódszámú villamos generátorok kivételével – táppontnak tekintve, a belőlük induló fogyasztói vezetékek darabszámát és kódjukat.

d) Vezetékenként táppontjuk kódját.

e) Vezetékenként fogyasztópontjuk kódját.

f) A vezetékek folyamatos sorszámát és sorszámonként a vezeték kódját.

9. Csomópontonként – amelyek a kazánok, kondenzátorok, hőfogyasztók és villamos generátorok kivételével mind a fogyasztók, mind a táppontok szerepét játsszák – a csomópont műszakilag előírt alsó és felső határterhelését, mint rögzített adatokat, továbbá a csomópont által még felvehető terhet, mint a kiosztás közben változó, nem növekvő adatot.

h) Nyomásfokozatonként a fogyasztók hőigényét, mint rögzített adatot, valamint az egy fogyasztók még kielégítetlen hőigényét, mint kiosztás közben változó, nem növekvő adatot.

i) A különböző nyomásfokozatú hőfogyasztócsoportok igénykielégítésének sorrendjét.

Megjegyezzük, hogy a két módszer szerinti kiosztás eredménye az előállított villamosenergiát tekintve csak akkorra fogyasztói hőigények esetén különbözik, amelyeknél a termelt gőz teljes energiatartalmára szükség van a fogyasztók gőzigényének kielégítésére és így a fogyasztókat teljesen vagy részben a turbinák kikerülésével, közvetlenül a gőzgyűjtősínekről látják el. A kiosztás eredményeként megkapjuk, hogy mely fő berendezések működnek, mekkora terhelésel és az erőmű mennyi villamosenergiát termel.

A napi kiosztások eredményét havonként összesítjük, a havi összegekből pedig éves összeget képzünk. A havi és az éves összesítéseket mindkét kiosztás alapján elvégezzük és kinyomtatjuk. A kétféle kiosztás eredményét nemcsak egymás ellenőrzésére, hanem egymás helyettesítésére is használjuk a havi összesítők számításakor abban az esetben, amikor valamelyik módszerrel a kiosztás nem jár sikerrel.

4. A tervező rendszer file-jainak kezelése

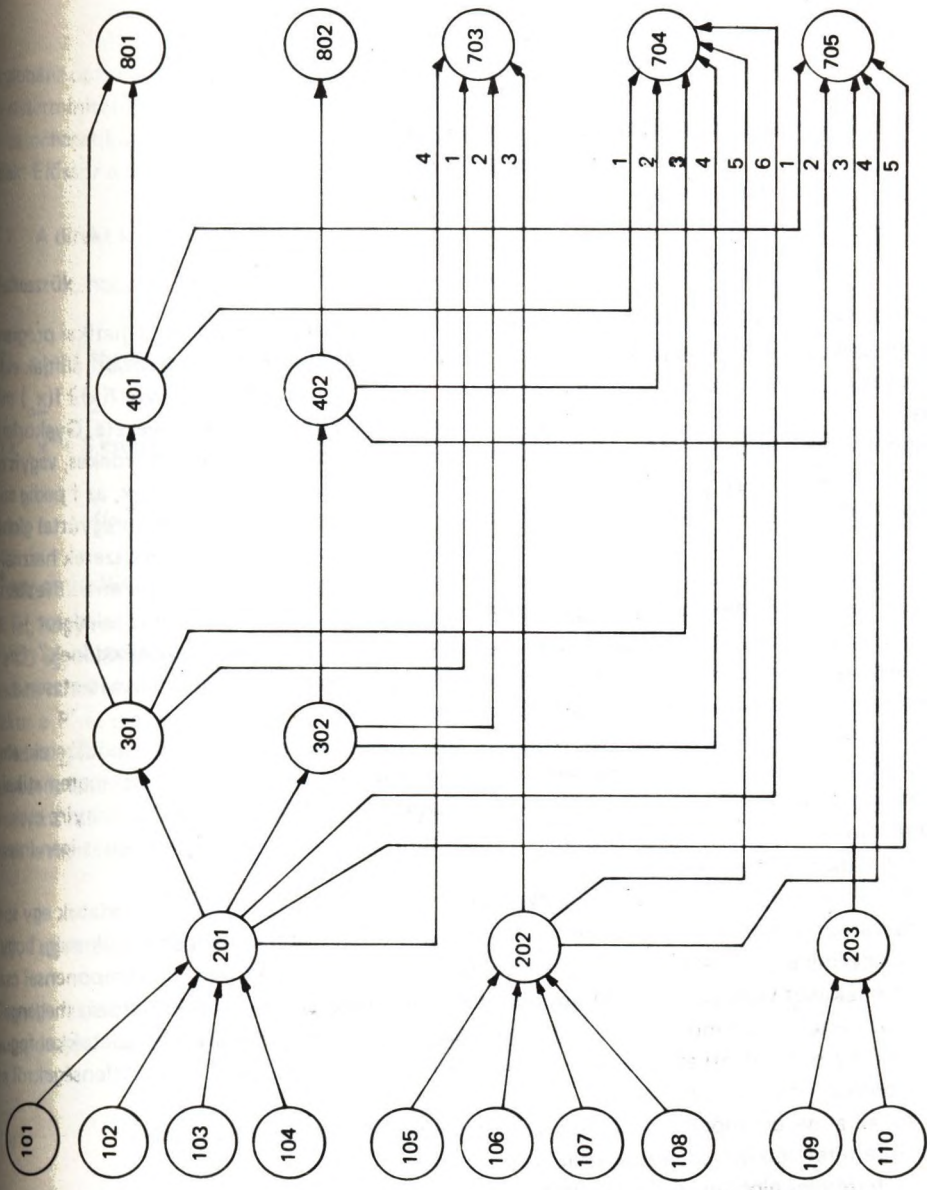
A tervező rendszernek az 1. ábrán bemutatott fő programja a megfelelő input adatállományokkal külön-külön is futtathatók. Egy-egy fő program output adatállománya a továbbiak valamelyikének inputját képezi, illetve a termelésszámító fő program outputja a tervező további – az itt ismertetett rendszeren kívüli – felhasználásra is rendelkezésre áll.

A szerzők jelenleg olyan nagyobb termeléstervező rendszer kifejlesztésén dolgoznak, amely a hazai együttműködő erőműrendszer teljes villamosenergiatermelését és tüzelőanyagfogyasztását tervezi a tüzelőanyagköltség, mint célfüggvény minimalizálásával. A fentebb bemutatott, hőszolgáltatást és kapcsolt villamosenergiatermelést tervező rendszer e nagyobb rendszernek alrendszere lesz.

Az alrendszer adatfile-jainak kezelését tehát úgy kellett megoldanunk, hogy a közvetlen adathozzáférés mind az alrendszer egyes fő programjai számára, mind a nagyobb rendszer számára biztosított legyen.

E követelmények kielégítésére az 1. ábrán bemutatott és azonos tervváltozathoz tartozó adatfile-okat egyetlen, közvetlen hozzáférésű háttértárolón helyeztük el. Az egyes file-ok az adathordozón csak az éppen szükséges helyet foglalják el, több helyet adva más file-oknak, amelyek a feldolgozott erőművek és évek nagyobb darabszáma miatt több helyet igényelnek. Az adatrekordokat olyan szervezésben kezeljük, ami az adatok rendkívül gyors elérését biztosítja. Az adatok biztonságát a mágneslemez tartalmának automatikus szalagra mentésével teremtjük meg.

A tervezés különböző fázisaiban létrejött file-ok archiválhatók (későbbi újrafelhasználáshoz mágnesszalagra menthetők), amivel helyet adunk a mágneslemezen egy más változat adatfilejainak.



2. ábra

Egy hőszolgáltató erőmű fő berendezéseinek gráfja

NEMKONVEX PROGRAMOZÁSI FELADATOK MEGOLDÁSI MÓDSZEREIRŐ

Dr. Forgó Ferenc
MKKE

Tekintsük az alábbi általános programozási feladatot:

$$\begin{matrix} f(x) \to \max \\ (1) \\ x \in S \end{matrix}$$

ahol S az n-dimenziós euklideszi tér (továbbiakban R^n) részhalmaza. A matematikai programozás „klasszikus" módszerei általában az (1) feladat egy *lokális maximumpontját* állítják elő (véges vagy végtelen számú lépésben), vagyis egy olyan $\underline{x}_o \in S$ pontot, amelyre $f(\underline{x}) \geqq f(\underline{x}_o)$ minden $\underline{x} \in N(\underline{x}_o) \cap S$ esetén, ahol $N(\underline{x}_o)$ az $\underline{x}_o$ pont valamilyen pozitív sugarú környezete. Gyakorlati szempontból azonban elsősorban a *globális maximum* ill. maximumpont az érdekes, vagyis egy olyan $\overline{\underline{x}} \in S$ pont, hogy $f(\overline{\underline{x}}) \geqq f(x)$ minden $x \in S$ esetén. Ha az S halmaz konvex, az f pedig szigorúan kvázikonkáv S-en, akkor ismeretes, hogy minden lokális maximumpont egyúttal globális is, s így a „klasszikus", többségükben számítástechnikailag is hatékony módszerek használhatók megoldásukra. Ha viszont csupán az f és S tulajdonságaiból nem tudunk arra következtetni, hogy egy lokális maximumpont globális is, akkor joggal tekinthetjük az (1) feladatot – amelyet *nemkonvex programozási feladatnak* nevezünk ekkor – „rosszul viselkedőnek". Ez azt jelenti, hogy új típusú „globális" módszerek is szükségesek megoldásra, természetesen a „klasszikus", lokális módszerek felhasználása mellett.

A probléma nehézségi fokának megfelelően a jelenleg rendelkezésre álló módszerek, amelyekről rövid áttekintést szeretnénk adni, meglehetősen „tökéletlenek". Számos matematikai tulajdonságuk felderíthetetlen, számítástechnikai viselkedésükről igen kevés, többnyire csak közvetett információ áll rendelkezésre. A kutatómunka azonban ezen a területen is igen intenzív s számos kutatási irány igen sokat ígérő.

Ebben az áttekintésben nem foglalkozunk a nemkonvex programozási feladatok egy speciális osztályával, a diszkrét programozási feladatokkal, helyesebben nem engedjük meg, hogy az S halmaz definíciójában expliciten elő legyen írva az, hogy az $\underline{x}$ vektor egyes komponensei csak diszkrét értékeket vehetnek fel. (Az implicit, folytonos feltételekkel való korlátozás megengedett.) Ilyen értelemben mondhatjuk azt, hogy csak a folytonos nemkonvex feladatokkal foglalkozunk. Ez nem jelenti azt, hogy az f függvényről és az S-et definiáló egyenlőtlenségekről eleve folytonosságot tételezünk fel.

Az alkalmazott módszerek jellege szerint célszerű megkülönböztetnünk a

- feltétel nélküli globális optimalizálást és a
- feltételes globális optimalizálást

noha a két kategória között az átmenetet a SUMT-jellegű módszerek biztosítják.

1. Feltétel nélküli globális optimalizálás

Feltesszük, hogy az (1) feladat globális optimumpontja az S halmaz belső pontja. Általában azt is célszerű feltennünk, hogy S egy n-dimenziós tégla:

A globális optimalizálás módszerei két nagy csoportra oszthatjuk.
- determinisztikus
- sztochasztikus

módszerek. Először a determinisztikus módszerekkel foglalkozunk.

1.1. A direkt keresés módszere

Feltesszük, hogy az f függvény Lipschitz-féle, vagyis van olyan M konstans, hogy

$$|f(\underline{x}^1)-f(\underline{x}^2)| \leq M|\underline{x}^1-\underline{x}^2| \tag{2}$$

minden $\underline{x}^1, \underline{x}^2 \in S$ esetén. Keresünk egy „ε optimális" megoldást, vagyis olyan $\bar{\underline{x}} \in S$ pontot, hogy

$$f(\underline{x}^*) - f(\bar{x}) \leq \varepsilon, \qquad (\varepsilon > 0)$$

ahol $\underline{x}^*$ egy optimális megoldás.

Tegyük fel, hogy rendelkezésünkre áll $\underline{x}^1, \underline{x}^2, \ldots \underline{x}^r \in S$ pont és $p^r = \max\{f(\underline{x}^1), f(\underline{x}^2), \ldots f(\underline{x}^r)\}$. Ekkor a (2) egyenlőtlenséggel meg lehet határozni olyan $\underline{x}^1, \underline{x}^2, \ldots \underline{x}^r$ középpontú kockákat, amelyekben nincs olyan pont, ahol a függvényérték p^r-nél nagyobb. Ha ezek a kockák lefedik S-et, akkor a P^r-et szolgáltató pont ε-optimális, ellenkező esetben választunk egy, a kockákkal le nem fedett $\underline{x}^{r+1}$ pontot és az eljárást megismételjük. Az $\underline{x}^{r+1}$ pont kiválasztásánál lokális módszereket is lehet alkalmazni. A módszer csak kis n esetében gazdaságos, hátránya, hogy a fix M konstans miatt a kockák kicsik, így sok pontot kell meghatározni. Kényelmetlen még a lefedett és le nem fedett részek nyilvántartása.

1.2. Trajektória módszerek

Tegyük fel, hogy a folytonos gradiens módszerrel keressük meg (1) feladat egy $\bar{\underline{x}}_o$ lokális maximumpontját. Ez azt jelenti, hogy kiindulva egy x pontból a

$$\frac{d\underline{x}(t)}{dt} = \frac{1}{1+\|\underline{g}\ [\underline{x}(t)]\|}\ \underline{g}[\underline{x}(t)] \tag{3}$$

differenciálegyenletrendszer által meghatározott trajektória mentén az $\bar{\underline{x}}_o$ pontba érkezünk el, ha $t \to \infty$. $\underline{g}(\underline{x})$ az $f(\underline{x})$ gradiensét jelöli. Mindazon pontok halmazát, amelyből kiindulva a (3) által meghatározott trajektória mentén az $\bar{\underline{x}}_o$ pontba jutunk az $\bar{\underline{x}}_o$ *vonzáskörzetének* nevezzük.

Ha a lokális maximumpontok száma véges, akkor a vonzáskörzetek kompaktak és legfeljebb egy nullmértékű halmaz kivételével lefedik az egész R^n-et. Két vonzáskörzet találkozásán levő nyeregpont megtalálása esetén a nyeregpont környezetében van olyan pont, ahonnan kiindulva egy új lokális maximumpontot tudunk felderíteni. Ezt a pontot úgy is meg lehet határozni, hogy nem állítjuk elő az egész vonzáskörzetet, hanem csak megfelelő kvadratikus függvények nívóhalmazával közelítjük.

Megoldatlan kérdés jelenleg az egész S szisztematikus lefedése (valamennyi lokális maximumpont meghatározása), ha a lokális maximumpontok száma nagy.

A (3) differenciálegyenletrendszer helyett tekinthetjük az alábbit:

$$\frac{d\,\underline{x}\,(t)}{dt} = [\underline{G}\,(\underline{x})\,]^{-1}\;\underline{g}(\underline{x}) \tag{4}$$

ahol $\underline{G}\,(\underline{x})$ a Hesse-féle mátrix. Kiindulva egy $\underline{x}_o$pontból a (4)-et numerikusan integráljuk és így eg folytonos görbét kapunk. Ha az összes (4)-et kielégítő folytonos görbét elő tudnánk állítani, akkor valamennyi stacionárius pontot érintenének és ki tudnánk közülük választani a legnagyo függvényértéket adót. Ezeknek a görbéknek a szisztematikus előállítása még megoldatlan probléma, noha Branin sejtése szerint bizonyos szinguláris esetek kizárása esetén valamennyi (4)-et kielégítő trajektória átmegy az összesstacionáriusponton. Ezt a sejtést még sem bizonyítani, se cáfolni nem sikerült.

1.3. A globális maximumpont explicit előállítása

Tegyük fel, hogy az (1) feladatban az f (x) folytonos és (1)-nek egyetlen optimális megol dása van. Legyen ez $\underline{z}$. Ekkor

$$z_i = \lim_{\lambda\to\infty} \frac{\int_S x_i\,\exp\,[\,\lambda\,f(\underline{x})\,]\,d\underline{x}}{\int_S \exp\,[\lambda f\,(\underline{x})\,]\;d\underline{x}}\,, \qquad (i = 1, \ldots\ldots .m).$$

Ha az f(x) függvény Lipschitz-féle, akkor egy hibakorláthoz meg lehet egy véges λ-t úgy adni hogy a

$$\frac{\int_S x_i \exp\,[\,\lambda\,f\,(\underline{x})\,]\,d\underline{x}}{\int_S \exp\,[\;\lambda\,f\,(\underline{x})\,]\,d\underline{x}} \tag{5}$$

hányados z_i-t előírt pontossággal közelítse. Az (5) hányados kiszámítása nagy n esetén igen nehe általános módszerként csak Monte-Carlo típusú módszereket lehet javasolni. Érdemes lenne me vizsgálni, hogy speciális f (x) függvény és S tartomány esetén ezeknek az integráloknak a kiszá mítása egyszerűsíthető-e.

A sztochasztikus módszerek közül kettőt említünk meg.

1.4. Véletlen keresés módszere

Az egyszerű véletlen keresés abból áll, hogy egyenletes eloszlás szerint véletlen pontot ge nerálunk az S halmazból és mindig az addig elért legjobbat tároljuk. Ha a mintaelemszám tart végtelenhez, akkor 1 valószínűséggel megtaláljuk a globális maximumpontot. Ha valamilyen el zetes elképzelésünk van a globális maximumpont elhelyezkedéséről, akkor az eljárás konverge marad, tetszőleges folytonos, pozitív sűrűségfüggvényű eloszlás alkalmazása esetén. Az elosz reprezentálja a maximumpont elhelyezkedéséről alkotott „véleményünket".

Ha egy függvényérték számítás időigényes, akkor számítási munkát lehet megtakarítani azzal, hogy Lipschitz-típusú f (x) célfüggvény esetén egy véletlen generált pontban csak akkor számítjuk az f (x)-et, ha az nem esik az eddig meghatározott pontok köré írt kockák (gömbök) valamelyikébe. Így a véletlen keresést kombinálni lehet az 1.1 pontban tárgyalt, direkt keresés módszerével. Az 1 valószínűséggel való konvergencia továbbra is megmarad.

A véletlen keresés egy másik stratégiája a következő: Határozzunk meg valamilyen eloszlás szerint egy véletlen indulópontot, majd ebből a pontból valamilyen folyamatos pozitív sűrűségfüggvényű eloszlás szerint véletlen irányt generálunk. Ezen irány mentén egzakt maximalizálást végrehajtva kapjuk a következő interációs pontot, ahol ismét véletlen irányt választunk. Ennek a módszernek az alkalmazásához az kell, hogy hatékony egydimenziós maximalizálási eljárás álljon rendelkezésünkre.

1.5. A multistart módszer

Ennél a sztochasztikus módszernél egy interáció egy véletlen pontkeresésből és aztán ebből a pontból kiindulva egy lokális maximalizálásból áll. Ha a véletlen pontkereséshez használt eloszlás folytonos és pozitív sűrűségfüggvényű, akkor ez az eljárás is 1 valószínűséggel konvergál.

Gyakorlatban ezt a módszert szokták legtöbbször alkalmazni. Hatékonysága nagyban függ a lokális maximumkeresés gyorsaságától.

A sztochasztikus módszerek természetes sajátossága, hogy az optimális célfüggvényértékektől való maximális eltérésre csak valószínűségi ítélet formájában adható becslés. A mintaelem szám növelésénél is csak ilyen jellegű becslések jöhetnek szóba.

A feltétel nélküli globális optimalizálás egyéb módszereiről és a fentiek részleteiről jó áttekintést ad [1].

2. Feltételes (globális) optimalizálás

A következőkben feltesszük, hogy a zárt, korlátos S halmazt véges számú lineáris egyenlőtlenség írja le:

$$S=\left\{\underline{x}\,\middle|\, g_i(\underline{x})\geq 0\,,\qquad (i=1,\ldots\ldots m)\right\}$$

Noha különböző direkt keresési eljárások a feltételes optimalizálás esetében is alkalmazhatók, azokkal önállóan ritkán találkozunk. Egyéb módszerekkel kombinálva, főleg ezek konvergenciájának biztosítására hasznos segédeszköznek bizonyulnak.

2.1. A korlátozás és szétválasztás módszere

A következő, szeparábilis célfüggvényű feladaton mutatjuk be a megoldás elvét:

$$f(\underline{x}) \equiv \sum_{i=1}^{n} f_i(x_i) \longrightarrow \max$$

(6)

$$\underline{x} \in G$$

$$\underline{x} \in C \equiv \left\{ \underline{a} \leq \underline{x} \leq \underline{b} \right\},$$

ahol G zárt, konvex halmaz, f $(\underline{x})$ pedig Lipschitz-típusú függvény a C halmazon, továbbá $G \cap C \neq \emptyset$. Célunk egy „ε optimális" megoldás keresése. Legyen $g^o(\underline{x})$ az f egy felső konkáv burkolója a C halmazon és oldjuk meg (bármilyen lokális módszerrel) az alábbi konvex programozási feladatot:

(7)

$$g^o(\underline{x}) \longrightarrow \max$$

$$\underline{x} \in G \cap C$$

Ha $\underline{x}^o$ a (7)-nak optimális megoldása és $g^o(\underline{x}^o) - f(\underline{x}^o) \leq \varepsilon$, akkor $\underline{x}^o$ „ε optimális". Ellenkező esetben a C halmazt szétválasztjuk egy olyan x_j^o pontban, ahol $g_j^o(x_j) - f_j(x_j) > \varepsilon$ az alábbi módon:

(8)

$$C^1 = \left\{ \underline{x} \in C \mid a_j \leq x_j \leq x_j^o \right\},$$

$$C^2 = \left\{ \underline{x} \in C \quad x_j^o \leq x_j \leq b_j \right\},$$

és a C^1, (C^2) halmazon új $g^1(\underline{x})$, $g^2(\underline{x})$ konkáv burkolókat határozunk meg és megoldjuk a

(9)

$$g^i(\underline{x}) \longrightarrow \max$$

$$\underline{x} \in G \cap C^i \qquad (i = 1, 2)$$

feladatokat, melyeknek optimális célfüggvényértéke a z^1, (z^2) korlátokat adja. A korlátozás és szétválasztás elvének megfelelően a maximális korláthoz tartozó feladatnál ellenőrizzük f (x) és a burkoló eltérést az optimumban, majd szükség esetén további (8)-al analóg szétválasztást hajtunk végre.

A módszer konvergens, egy „ε optimális" megoldáshoz véges számú lépésben konvergál. Alkalmazása főleg akkor kecsegtet sikerrel, ha az f_j függvények „közel" konkávok, ill. csak kevés van közöttük, amelyik nem konkáv. Előnyös még, hogy állandóan van egy felső korlátunk a célfüggvényre.

2.2. Metszési módszerek

A metszési módszerek alapgondolata az, hogy az S lehetséges tartomány bizonyos részeit, amelyek további figyelmet már nem érdemelnek „levágjuk", vagyis a további vizsgálódásokból kizárjuk. Direkt és indirekt metszéseket fogunk megkülönböztetni.

a) *Direkt metszés*

Legyen $F_z = \{\underline{x} \mid f(\underline{x}) \leqq z\}$ az f függvény *alsó z-nívó halmaza.* Egy zárt C halmazt z szinten érvényes metszésnek nevezünk, ha

(i) $$C \cap S \subseteq F_z$$

(ii) $$\overline{S - C \cap S} \subseteq S$$ [1]

Ha tehát van egy olyan $\underline{y} \in S$ vektorunk, amelyre $f(\underline{y}) \geqq z$, akkor a $C \cap S$ halmazt nem érdemes tovább vizsgálnunk. Így felépíthetünk egy általános algoritmust, amely direkt metszéseken alapul:

1. lépés. Keressünk egy $\underline{x}_o \in S$ vektort. Ha ilyen nincs, akkor az (1)-nek nincs megoldása. Legyen $S_1 = S$, $z_o = f(\underline{x}_o)$.

k+1. lépés. Határozzunk meg egy z_{k-1} – szinten érvényes C_k metszést. Legyen $S_k = \overline{S_{k-1} - C_k}$. Keressünk egy $\underline{x}_k \in S_k$ pontot, legyen $z_k = \max \{z_{k-1}, f(\underline{x}_k)\}$ és menjünk a k+1-ik lépésre.

Ha egy $p > 0$ esetén $S_p = \emptyset$, akkor $z_p = \max_{\underline{x} \in S} f(\underline{x})$ és $\underline{y}$ egy optimális megoldása az (1) feladatnak, ha $f(\underline{y}) = z_p$.

A véges *konvergencia egy elégséges feltétele:* Van olyan $\delta > 0$ szám, hogy az $\underline{x}_{k-1}$ távolsága S_k-tól legalább δ minden k-ra.

A metszést célszerű úgy megkonstruálni, hogy $\underline{x}_{k-1} \in S_k$. Ha C_k egy zárt féltér, vagyis $= \{\underline{x} \mid d_k \underline{x} \leqq d_{k_o}\}$, akkor $S_k = S_{k-1} - C_k = \{\underline{x} \; \underline{x} \in S_{k-1}, \; d_k \underline{x} \geqq d_{k_o}\}$. Ha minden metszés ilyen, akkor *metszősík algoritmusról* beszélünk.

b) *Indirekt metszés*

A következő feladatot az (1) probléma egy lazításának nevezzük:

$$f(\underline{x}) \to \max$$
$$\underline{x} \in P \qquad (10)$$

ahol $S \leqq P \leqq R^n$. Ha $\underline{x}_o$ a (10) optimális megoldása és $\underline{x}_o \in L$, akkor $\underline{x}_o$ az (1) feladatnak is optimális megoldása.

Tegyük fel, hogy P zárt és korlátos.

A zárt, nemüres C halmazt a P és S halmazokra vonatkozó érvényes indirekt metszésnek nevezzük, ha

(i) $S \subseteq \overline{P - C}$

(ii) $\overline{P - C} \subset P.$

A következő általános algoritmus indirekt metszéseken alapszik.

O. lépés. Legyen $P_0 = P$. Oldjuk meg a (10) feladatot. Ha $\underline{x}_0$ a (10) optimális megoldása és $\underline{x}_0 \in S$, akkor $\underline{x}_0$ az (1) optimális megoldása. Ha $\underline{x}_0 \notin S$, akkor menjünk az 1. lépésre.

k. lépés. Határozzunk meg egy a P_{k-1} és S halmazokra vonatkozó C_k érvényes indirekt metszést.

Legyen $P_k = \overline{P_{k-1} - C_k}$ és oldjuk meg a

$$(11) \qquad \begin{array}{l} f(\underline{x}) \to \max \\ \underline{x} \in P_k \end{array}$$

feladatot. Ha ennek optimális megoldása, $\underline{x}_k \in S$, akkor $\underline{x}_k$ az (1) optimális megoldása. Ellenkező esetben menjünk a k+1-ik lépésre. (Azt mindig feltesszük, hogy (11)-nek minden k-ra van optimális megoldása.)

Általában a C_k metszés olyan szokott lenni, hogy $\underline{x}_{k-1} \notin P_k$.

A *véges konvergencia két egyszerű elégséges feltétele:*

(i) Van olyan $\delta > 0$, hogy $\underline{x}_{k-1}$ és P_k távolsága nagyobb δ-nál minden k-ra.
(ii) Van olyan $\delta > 0$, hogy $f(\underline{x}_k) - f(\underline{x}_{k-1}) \geq \delta$ minden k-ra.

Most is, ha C_k egy zárt féltér, akkor *indirekt metszősík módszerről* beszélünk. Konkrét metszősík módszerekről az [5] munkában lehet részletesen olvasni.

3. Multiplikátorok és a nemkonvex programozás

Az utóbbi időkben néhány kísérlet történt arra, hogy a nemkonvex programozást is valamilyen „elméleti alapra" helyezzék természetesen szem előtt tartva az elméleti eredmények potenciális vagy közvetlen hasznosíthatóságát. Ebben a vonatkozásban a dualitási és nyeregpont-típusú tételektől lehet eredményt várni. Ezek az eredmények a Lagrange-függvény vagy annak általánosításaival, a bővített Lagrange függvénnyel kapcsolatosak. Tekintsük az alábbi, egyenlőségekkel korlátozott nemkonvex feladatot:

$$(12) \qquad \begin{array}{ll} f(\underline{x}) \longrightarrow \max & \\ g_i(\underline{x}) = 0, & (i = 1, \ldots .m) \end{array}$$

ahol valamennyi függvény folyamatosan deriválható és a feltételi függvények gradiensei lineárisan függetlenek. A

(13) $$\emptyset\ (\underline{x},\ \underline{u}) = f(\underline{x}) - \sum_{i=1}^{m} u_i\ \ g_i(\underline{x}) - \frac{1}{2}\beta \sum_{i=1}^{m} [g_i(x)]^2 \beta > 0$$

függvényt, ahol β fix konstans *bővített Lagrange függvénynek* nevezzük. Bizonyítható, hogy ha β elég nagy, akkor ennek a függvénynek van nyeregpontja, s így a

(14) $$\max_{\underline{x}} \quad \min_{\underline{u}} \emptyset\ (\underline{x}, \underline{u})$$

primál feladat helyett a

(15) $$\min_{\underline{u}} \quad \max_{\underline{x}} \emptyset\ (\underline{x}, \underline{u})$$

duál problémát is megoldhatjuk. A $\max \emptyset\ (\underline{x}, \underline{u})$ függvény elég nagy β esetén szigorúan konvex, s így a (15)$^{\underline{x}}$ egy konvex programozási feladat. Nemkonvex f és g_i függvények esetében azonban egy konkrét $\underline{u}$ esetén a $\max_{\underline{x}} \emptyset\ (\underline{x}, \underline{u})$ meghatározása egy feltétel nélküli nemkonvex probléma megoldását igényli, amely, mint láttuk az előzőekben ugyanolyan nehézségi fokú, mint az eredeti (12) probléma. Ezért a (15) feladat megoldására olyan jellegű feltétel nélküli minimalizálási algoritmusra lenne szükség, amely a $\emptyset(\underline{x}, \underline{u})$ függvény $\underline{x}$ szerinti maximalizálását nem igényli.

A multiplikátor módszert egy más, sajátos módon is lehet alkalmazni szeparábilis nemkonvex programozási feladatok megoldására. Tekintsük az alábbi problémát:

(16)
$$\sum_{j=1}^{n} f_j(x_j) \longrightarrow \max$$

$$\sum_{j=1}^{n} g_{ij}(x_j) \leqq b_i \qquad (i = 1, \ldots\ldots m)$$

$$0 \leqq x_j \leqq d_j, \qquad (j = 1, \ldots\ldots n)$$

ahol a $g_{ij}(x_j)$ függvények nemnegatívok és folytonosak a $D = \left\{ \underline{x}\ \ 0 \leqq \underline{x}_j \leqq d_j,\ \ (j = 1, \ldots . n) \right\}$ halmazon. Válasszunk egy k racionális pozitív számot és tekintsük a (16) feladat egy lazítását:

(17) $$\sum_{j=1}^{n} f_j(x_j) \longrightarrow \max$$

$$\sum_{j=1}^{n}\left[\frac{g_{ij}(x_j)}{k}\right] \leq \left[\frac{b_i}{k}\right] \quad , \qquad (i = 1, \ldots\ldots\ldots m)$$

$$\underline{x} \in D$$

ahol [r] az r szám egész részét jelöli. Az y_i nemnegatív egészértékű kiegészítő változók bevezetésével (ezek szintén korlátosak) a (17)-et átalakíthatjuk:

$$\sum_{j=1}^{n} f_j(x_j) \longrightarrow \max$$

(18)

$$\sum_{j=1}^{n}\left[\frac{g_{ij}(x_j)}{k}\right] + y_i = \left[\frac{b_i}{k}\right] \qquad (i = 1, \ldots\ldots m)$$

$$\underline{x} \in D$$

$$\underline{y} \in E, \quad \underline{y} = \text{egész},$$

ahol E egy korlátos tartomány. Az u_i egészértékű multiplikátorokkal (18)-at egy hátizsákfeladat alakítjuk:

$$\sum_{j=1}^{n} f_j(x_j) \longrightarrow \max$$

(19)

$$\sum_{i=1}^{m}\sum_{j=1}^{n} u_i\left[\frac{g_{ij}(x_j)}{k}\right] + u_i y_i = \sum_{i=1}^{m} u_i\left[\frac{b_i}{k}\right]$$

$$\underline{x} \in D, \quad \underline{y} \in E, \quad \underline{y} = \text{egész}.$$

Bizonyítható, hogy van olyan $\underline{u}^o(k)$, hogy (19) és (18) optimális megoldásai egybeesnek és amellett (19) jobboldala minimális. Legyen $\underline{x}^o(k)$ a (19) optimális megoldása $\underline{u}^o(k)$ mellett. Ha $\underline{x}^o(k)$ kielégíti a (16) feltételeit, akkor ez a (16)-nak is optimális megoldása. Bizonyítható, hogy az $\underline{x}^o(k)$ sorozat minden torlódási pontja, ha $k \longrightarrow O$, a (16) optimális megoldása. Az

$$\left\{\frac{1}{\|\underline{u}^o(k)\|}\,\underline{u}^o(k)\right. \qquad \text{sorozat torlódási pontjait pedig tekinthetjük a (16) feltételeihez tartozó}$$

„árnyékáraknak".

Kiépíthető, az (19) feladatot többször megoldó, az u_i szorzókat egyidejűleg meghatározó
an algoritmus, amely a (16) egy tetszőleges pontosságú közelítő megoldását véges számú lé-
den meghatározza. Az algoritmus a korlátozás és szétválasztás elvén alapszik.
Szintén a multiplikátor elvet használja és praktikus, heurisztikus algoritmusok kiépítésénél
n hasznos EVERET általánosított Lagrange multiplikátor módszere [6].

odalom

1] Szegő, G. P.; Dixon, L. C. W.: Toward global optimisation. Proceedings of a Workshop at the University of Cagliari, Italy, Oct. 1974. North–Holland, 1975.
2] Pincus, M.: A closed form solution of certain programming problems. Operations Research, 16. 1968. 690–694. o.
3] Devroye, L. P.: Progressive global random search of continuous functions. Mathematical Programming 15 (1978) Wo. 3. 330–342. o.
4] Falk, J. E.: SOLAND, R. M.: An algorithm for separable nonconvex programming problems. Management Science, 15 (1969) No. 9. 550–569. o.
5] Forgó, F.: Nemkonvex és diszkrét programozás. Közgazdasági és Jogi Könyvkiadó, Budapest, 1978.
6] Everett, H.: Generalized Lagrange multiplier method for solving problems of optimum allocation of resources. Operations Research, 11 (1963) 399–417. o.
7] Zoutendijk, G.: Mathematical Programming Methods. North–Holland Publ. Comp. Amsterdam, 1976.

PROPHET ÚJELVŰ NYELV ÉS ALKALMAZÁSA

Futó Iván, Szeredi János, Rédei János
SZKI

A PROPHET (PROgramming Prolog with HELp of Time) egy logikai alapú, párhuzamos és adatmegoldási lehetőségekkel is rendelkező újnyelvű nyelv, melyben központi szerepet tölt be a *belső idő* fogalma.

Bevezetés

Az 1970-es évek elejétől egyre inkább felmerült annak igénye, hogy olyan újelvű (mesterségesintelligencia-) nyelveket hozzanak létre, melyekben a feladatok megoldása nem szekvenciálisan, hanem egyidejűleg, párhuzamosan történik [3, 4].

A párhuzamos feladatmegoldás egyfajta megvalósítását tükrözi a Magyarországon is elterjedt PROLOG nyelvnek az Imperial College-ban (London) készült IC–PROLOG elnevezésű változata is, melynek első leírása 1978-ban jelent meg [5].

A gyakorlati életből vett modellek egy részénél azonban a párhuzamos feladatmegoldó tevékenységeknél figyelembe kell venni, hogy az egyes tevékenységekhez időpont és időtartam is tartozik.

Az általunk megvalósított újelvű PROPHET nyelvben lehetőséget kívántunk biztosítani az egidejűleg történő párhuzamos feladatmegoldáson kívül a tevékenységek időben történő koordinálására is.

Ennek megfelelően a PROPHET a hagyományos mechanikus tételbizonyítón kívül rendelkezik egy belsőóra-mechanizmussal is, amely hasonló a hagyományos szimulációs nyelvekéhez (SIMULA, GPSS) [6, 7].

Ellentétben azonban ezekkel a szimulációs nyelvekkel, a PROPHET a futás során az időben visszalépésre is képes, ha a modell dead-lockba vagy zsákutcába kerül, azaz nem képes megvalósítani az eléje kitűzött célt.

A célok elérése érdekében szükség esetén képes az eredeti modell automatikus átstrukturálására, azaz úgy kibővíteni bizonyos keretek között, hogy ebből a kiindulási állapotból a kitűzött célok már elérhetők legyenek.

Első alkalmazásaként a Városépítési Tudományos Intézettel közösen a hosszú- és nagytávú regionális modelleknél kívánjuk használni.

1. A PROPHET általános jellemzése

A PROPHET-programokat logikai definíciók halmaza alkotja, melyeket Horn-formulákkal – az elsőrendű predikátumkalkulus egy megszorításával – adunk meg.

A logikai definíciók a nyelv dinamikus elemeinek, a sejteknek a tevékenységére, illetve lehetséges fejlődési alternatíváira vonatkoznak, vagy pedig ezek előfeltételeinek a definíciói.

A rendszerben egyidejűleg meghatározatlan számú aktív sejt tevékenykedhet. Minden létrehozott sejthez születése pillanatában hozzárendelünk egy célt, amelyet élete során kell érnie, majd ezután meghal, vagyis kikerül a rendszerből.

Sejtek dinamikusan futás közben bármikor létrehozhatók, sőt meg is szüntethetők, tekintet nélkül arra, hogy sikerült-e elérniük a születésükkor hozzájuk rendelt célt. A sejtek tevékenységét céljuk elérése érdekében egy beépített mechanikus tételbizonyító vezérli egy belsőóra-mechanizmus segítségével.

Bizonyos tevékenységekhez időtartam vagy időpont rendelhető.

A rendszerben tevékenykedő sejtek egymás között háromféleképpen kommunikálhatnak, illetve befolyásolhatják egymás fejlődését, illetve egymás céljainak elérését.

A sejtek közötti kommunikáció a közös változókon keresztül, a közös adatbázis módon keresztül, illetve egy egyszerű „démon-mechanizmuson" keresztül röerénik.

Egy PROPHET-program sikeres lefutását az *összes* sejt sikeres fejlődése, azaz az eléjük tűzött célok teljesülése jelenti.

Megjegyezzük, hogy kitűzhetők olyan célok is, melyek teljesülése közben bizonyos sejtek meghalhatnak, de a program futása sikeres lesz annak ellenére, hogy az így eltűnt sejtek céljai nem teljesültek.

A sejtek fejlődésük, illetve céljaik elérése közben zsákutcába kerülhetnek, vagy egymást blokkolyozva dead-lock-ba eshetnek. Ilyenkor a beépített deduktív mechanizmus a teljes megoldási folyamatot visszalépteti a rendszer egy korábbi állapotába, ahol új alternatívát választ valamelyik sejt számára (esetleg többnek is), és újból a célok irányába tudnak haladni a sejtek, illetve a teljes sejtrendszer (rendszer). A deduktív mechanizmus ezt mindaddig megteszi, amíg vagy sikerül elérni a rendszer céljait, vagy pedig a teljes keresési teret (összes lehetséges alternatívát) bejárva nem talál megoldást a feladatnak, és ekkor hibaüzenetet ad.

A PROPHET-programok működésére alább közlünk egy kis példát.

2. Egy PROPHET-mintapélda

Pali és Panni sejtek közös szórakozásként moziba akarnak menni. A Pali sejt csak úgy szórakozhat, vagyis részére a közös szórakozás csak akkor jön létre, ha valamely moziba olyan filmre van jegy, amely film megfelel a Panni sejtnek, továbbá Pali odaér az illető moziba az előadás kezdetéig.

A Panni sejt pedig csak akkor szórakozhat, ha a Pali sejt által javasolt film jó film és ő is odaér az előadás kezdetéig a moziba.

Nézzük meg a fenti példának egy lehetséges PROPHET-beli megfogalmazását!

```
(1) +szórakozhat (Pali, *mozi, *film, *időpont)
      –van_jegy (*mozi, * film, *időpont)
      –odaér (Pali, *mozi,*időpont)
      –wait_for (megfelel_Panninak (*film) )!

(2) +szórakozhat (Panni, *mozi, * film, *időpont)
      –wait (Pali_talált_jegyet (*film) )
      –jó_film (*film)
      –odaér (Panni, *mozi, *időpont)
      –send (megfelel_Panninak (*film) )!

(3) +Pali_talált_jegyet (*film)
      –identifier (*film) !

(4) +odaér (*valaki, *mozi, *időpont)
      –utazik (*valaki, *mozi)
      –systemtime_is_less_than (*időpont) !
```

(5) +van_jegy (Corvin, Rosszemberek, 30)!
(6) +van_jegy (Urania, Magyarok, 45)!
(7) +van_jegy (Puskin, K. O., 30)!
(8) +utazik (Pali, Corvin) During 20!
(9) +utazik (Panni Corvin) During 10!
(10) +utazik (Pali, Urania) During 20!
(11) +utazik (Panni, Urania) During 25!
(12) +utazik (Pali, Puskin) During 15!
(13) +utazik (Panni, Puskin) During 10!
(14) +jó_film (Magyarok)!
(15) +jó_film (Apa)!
+Fin!
(16) —create (Pali, szórakozhat (Pali, *mozi, *film, *időpont))
—create (Panni, szórakozhat (Panni, *mozi, *film, *időpont))!

Az egyes állítások értelmezése röviden:

(1): Pali akkor szórakozhat a *mozi-ban a *film-en az *időpont-ban, ha van jegy a *moziba a *film-hez az *időpont-ra, és Pali odaér a *mozi-ba az *időpont-ra, továbbá tudnia kell, hogy Panninak megfelel-e a *film.

(2): szórakozhat Panni a *mozi-ban a *film-en az *időpontban, ha Pali talált jegyet a *mozi-ba a *film-hez az *időpont-ra. Ebben az esetben megüzeni Palinak, hogy megfelel Panninak a *film.

(3): Pali talált jegyet a *film-hez, ha a *film változó már kapott értéket.

(4): odaér *valaki a *mozi-ba az *időpont-ra, ennek a *valaki-nek a *mozi-ig tartó utazása végén a rendszeridő kisebb, mint a film kezdési időpontja.

(5)—(7): értelemszerűen.

(8)—(13): ki melyik moziba mennyi időegység alatt tud eljutni.

(14)—(15): értelemszerűen.

(16): kitűzi a feladatot, létrehoz két sejtet Palit és Pannit a szórakozhat Pali (Panni) ugyanabban a *mozi-ban ugyanazon a *film-en ugyanabban az *időpont-ban célokkal,

Az egyes sejtek közötti kommunikáció a példában részben a közös *mozi, *film, *időpont változókon keresztül, részben pedig a démon-mechanizmusok keresztül (wait_for és send járások) történik (1), (2). A közös logikai változókon e keresztül történő kommunikáció lényege, hogy ha egy logikai változónak valamelyik sejt tevékenysége során értéket ad, ez automatikusan megjelenik a többi sejtnél is (egyesítési algoritmus).

A démon-mechanizmus [3] lényege, hogy egyes sejtek üzenetre várhatnak, melyeket más sejtek küldenek a számukra. Mikor egy sejt elküld egy üzenetet a figyelő démonok azonnal elindítják az üzenetre váró sejteket sorban egymás után (egyprocesszoros gép), míg az üzenetet küldő sejt tevékenysége pillanatnyilag felfüggesztődik.

3. A példaprogram futási diagramja

Várakozási lista: O: Pali. O: Panni. NIL (A O: az előjegyzési időpontot jelenti)
Blokkolt lista: NIL
Démonok sejtjei: NIL
Aktív sejt: Pali
Rendszeridő: O

(1) —szórakozhat (Pali, *mozi, *film, *időpont)!
*mozi:=*mozi, *film: = *film, *időpont: = *időpont
—van_jegy (*mozi, *film, *időpont)
—odaér (Pali, *film, *időpont)
—wait_for (megfelel_Panninak (*film))!
(6) *mozi: = Corvin, *film: = Rosszemberek, *időpont: = 30
—odaér (Pali, Corvin 30)
—wait_for (megfelel_Panninak (Rosszemberek))!
(4) *valaki: = Pali, *mozi: = Corvin, *időpont: = 30
—utazik (Pali, Corvin)
—system_time_is_less_then (30)
—wait_for (megfelel_Panninak (Rosszemberek))!

(8) A Pali sejt tevékenysége felfüggesztődik, legközelebb 20 időegység után kell vele foglalkozni

(Pali 29 időegységig utazik a Corvin moziba)

Várakozási lista: O : Panni. 20: Pali. NIL
Blokkolt lista: NIL
Démonok sejtjei: NIL
Aktív sejt: Panni
Rendszeridő: O

(2) —szórakozhat (Panni, Corvin, Rosszemberek, 30)!
*mozi: = Corvin, *film: = Rosszemberek, *időpont: = 30
—wait (Pali_talált_jegyet (Rosszemberek))
—jó_film (Rosszemberek)
—odaér (Panni, Corvin, 30)
—send (megfelel_Panninak (Rosszemberek))!
(3) —Pali_talált_jegyet (Rosszemberek) !
*film: = Rosszemberek

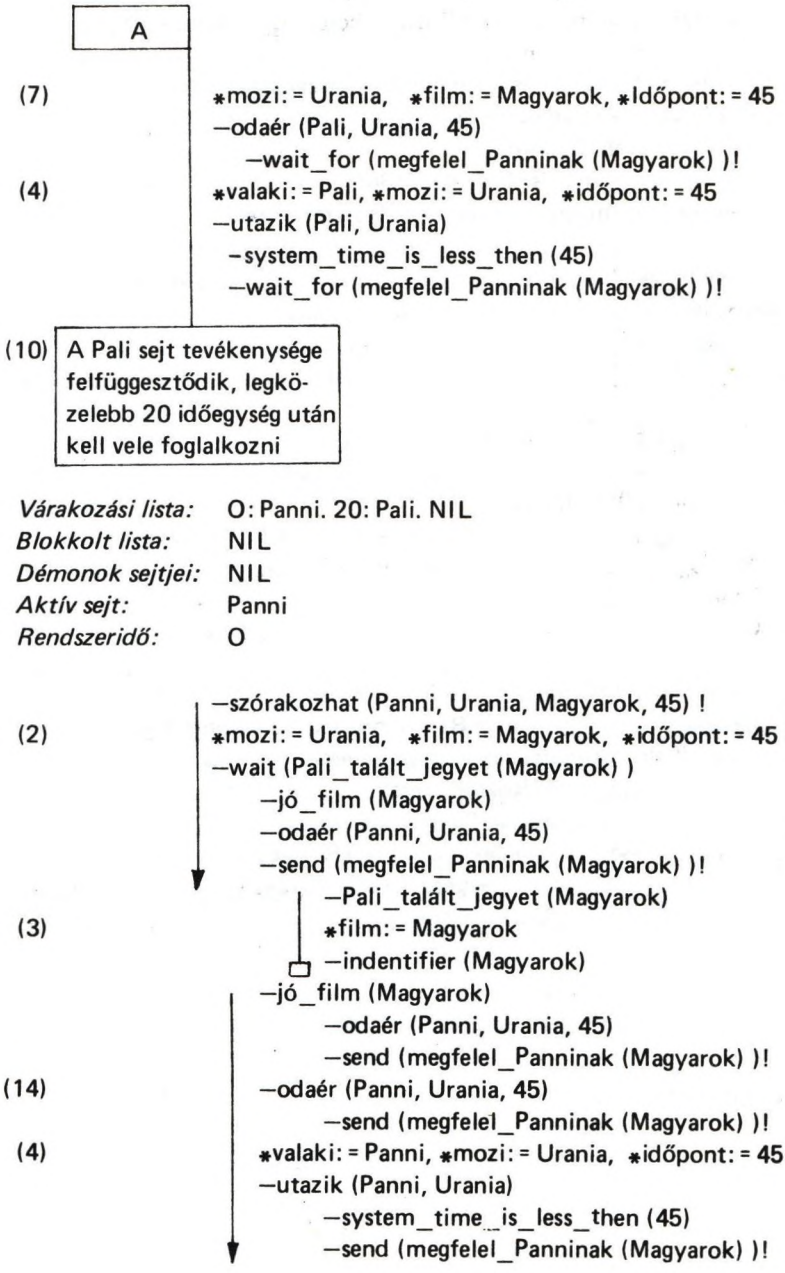
Beépített eljárás

—identifier (Rosszemberek)

—jó_film (Rosszemberek)
—odaér (Panni, Corvin, 30)
—send (megfelel_Panninak (Rosszemberek))

Zsákutca

A Rosszemberek-ről nem szerepel az adatbázisban, hogy jó film, így a feladat megoldásában visszalépés történik, mindaddig, amíg új alternatívát nem lehet választani valamelyik sejt számára. Természetesen ilyenkor a rendszeridő is visszalép. Jelen esetben a visszalépés a Pali sejt filmválasztásáig tart, mivel csak ott lehet új alternatíva kiválasztásával a megoldást folytani (A állapot).



(11) A Panni sejt tevékenysége felfüggesztődik, legközelebb 25 időegység után kell vele foglalkozni

Várakozási lista: 20: Pali. 25: Panni. NIL
Blokkolt lista: NIL
Démonok sejtjei: NIL
Aktív sejt: Pali
Rendszeridő: 20

mivel a O időpontra több esemény nem lett előjegyezve, a rendszeridő a legközelebbi esemény előjegyzési időpontjára áll (20).

—system_time_is_then (45)
—wait_for (megfelel_Panninak (Magyarok)) !
mivel 20 < 45
—wait_for (megfelel_Panninak (Magyarok)) !

Pali sejt tevékenysége felfüggesztődik mindaddig, amíg egy megfelel_Panninak (Magyarok) üzenetet nem kap

Várakozási lista: 25: Panni. NIL
Blokkolt lista: NIL
Démonok sejtjei: Pali. NIL
Aktív sejt: Panni
Rendszeridő: 25

—system_time_is_less_then (45)
send (megfelel_Panninak (Magyarok))
mivel 25 < 45
—send (megfelel_Panninak (Magyarok))

Megszakad a Panni sejt tevékenysége, folytatja a tevékenységét az üzenetre váró Pali sejt.

Aktív lista: 25: Pali. 25: Panni. NIL
Blokkolt lista: NIL
Démonok sejtjei: NIL
Aktív sejt: Pali
Rendszeridő: 25

Pali elérte célját

Aktív lista: 25. Panni. NIL
Blokkolt lista: NIL
Démonok sejtjei: NIL
Aktív sejt: Panni
Rendszeridő: 25

Panni elérte célját

Aktív lista: NIL
Blokkolt lista: NIL
Démonok sejtjei: NIL
Aktív sejt: O
Rendszeridő: 25

A feladatot tehát Pali és Panni szórakozhat az Urániában a Magyarok-on 45 perc múlva megoldással megoldottuk.

4. A PROPHET belsőóra-mechanizmusa

Az idő kezelését a belsőóra-mechanizmus végzi, melynek vázlatos működését az alábbiakba közöljük.

1. Induláskor az óramechanizmus megnézi, hogy a sejtek születéséhez hozzárendelt kezdeti időpontok közül melyik a legkisebb, és az órát erre az időpontra állítja. (Ha egyik sejthez sem rendelünk kezdeti időpontot, akkor O-nak tekinti.)
2. Ezután elindítja az első erre az időpontra előjegyzett sejt tevékenységét, és mindaddig futtatja, amíg az lefut vagy felfüggesztésre nem kerül (feltételtől függően vagy feltétel nélküli módon), vagy olyan elemi esemény végrehajtása (hivása) nem következik, mely időtartammal rendelkezik. (Ha nem adunk meg explicit időtartamot egy elemi eseményt reprezentáló tényállításnak, akkor azt O-nak tekinti az óramechanizmus.)
3. Ha a sejt felfüggesztésre kerül, akkor az óramechanizmus a felfüggesztett sejttel azonos időpontra előjegyzett következő sejtet indítja el. Ha ilyen nincs, akkor a belső órát a legközelebbi időpontra előjegyzett esemény idejére állítja, és elindítja az első erre az időpontra előjegyzett sejtet.
4. Amennyiben a sejt időtartammal rendelkező elemei eseményt hajt végre, akkor annak lefutása után az óramechanizmus az esemény időtartamát hozzáadja a belső óra pillanatnyi idejéhez, a sejtet felfüggeszti és előjegyzi az így kapott időpontra, majd rátér a 3. lépés végrehajtására.
5. Ezt mindaddig csinálja, amíg az összes sejt tevékenysége le nem fut (természetesen visszalépés esetén az idő is visszalép).

Minden esetben, amikor valamelyik sejt új időtartamra előjegyződik, az óramechanizmus ellenőrzi, hogy ez az időpont nem haladja-e meg a sejt céljának elérésére előírt végső időpontot, mert ha igen, akkor visszalépés történik.

4. Az idő megadása PROPHET-ben

Az egyes sejtek tevékenységének megkezdési és legkésőbbi befejezési időpontját a sejt létrehozásakor lehet megadni az alábbi módon.

—create (Sejt, Cél) {Begin T1} {End T2}

A kapcsos zárójelek közötti részek opcionálisak.

Az így meghatározott célállítások jelentése értelemszerű. Ahol nincs Begin ott a kezdés időpontja (T1) 0. Ahol nincs End, ott a befejezési időpont (T2) a jelenlegi megvalósításnál 100 000 időegység, amit a felhasználó saját célja szerint értelmezhet (perc, óra stb.).

A sejtekkel történő eseményekhez időtartamot az alábbi módon lehet hozzárendelni:

+<event> During T!

Jelentése: Az <event> esemény időtartama T időegység, vagyis a sejt T időegységig felfüggesztődik.

+<event> For T!

Jelentése: Az <event> esemény befejezésének előírt időpontja T, amennyiben a belső idő < T, a sejt tevékenysége felfüggesztődik T időpontig, ellenkező esetben visszalépés.

+<event> Before T!

Jelentése: Az <event> eseménynek a T időpont előtt kell befejeződnie. Amennyiben a belső idő < T-nél, a sejt futása folytatódik, ellenkező esetben visszalépés.

+<event> After T!

Jelentése: Az <event> eseménynek a T időpont után kell bekövetkeznie. Amennyiben a belső idő > T a sejt futása folytatódik, ellenkező esetben visszalépés.

Lehetőség van továbbá a sleep (T) beépített eljárás hívásával egy sejtet „T" pillanatnyi értékének megfelelő időtartamra felfüggeszteni.

5. Alkalmazás

A PROPHET alkalmazását először a VÁTI-val közösen hosszú- és nagytávú regionális tervezés modelljein kívánjuk kipróbálni. A vizsgálandó területi egységeknek rendszerünkben sejteket feleltetünk meg.

[1] Futó I., Szeredi J., Rédei J.:
Egy párhuzamos programozási lehetőségekkel rendelkező újelvű nyelv SZKI 1978. november.

[2] Futó I., Szeredi J., Rédei J.:
A PAPLAN újelvű nyelv
SZKI 1979. március.

[3] J. P. Rulifson, J. A. Derksen, R. J. Waldinger:
OA4 a procedural calculus for reasoning,
Artificial Intelligence Center technical note 73 SR I 1972.

[4] C. Hewitt:
Viewing Control Structures as Patterns of Passing Messages,
MIT Memo no 410. 1976.

[5] R. Kowalski:
IC–PROLOG,
Imperial College London 1978.

[6] GPSS/360 User's Manual,
IBM 420–03261.

[7] G. M. Birtmisth, O. J. Dahl, K. Nygovid:
SIMULA Begin,
Auerbach Publishers Inc 1973.

AZ INTUÍCIÓ SZEREPE ÉS ESZKÖZEI A PROBLÉMAMEGOLDÁSBAN

Füle Károly
SZÁMKI

1. Módszeres és intuitív problémamegoldás

A *problémamegoldás* a legjellegzetesebb emberi tevékenység. Éppen ezért az ember szá-ára legtermészetesebb és a legérdekesebb probléma éppen az, hogy magát a problémamegoldási ékenységet megértse. A legizgatóbb és egyben a legreménytelenebb cél az ember számára az, gy olyan egyetemes, tökéletes módszert találjon, amellyel minden problémát akadálytalanul g tud oldani. A megismerés mai szintjén, ilyen egyetemes módszer nincs. Mégis sikerült és yre inkább sikerül egy ilyesféle módszer bizonyos elemeit kikristályosítani és megfogalmazni. zeket az ún. heurisztikus eljárásokat és szabályokat következetesen alkalmazva igen sok prob-ma megoldható és hatékonyabban oldható meg.

A *módszeres problémamegoldás* lényegében egy heurisztikus felismerésen alapul, amelyet ég Descartes fogalmazott meg. Ez a felismerés abban áll, hogy a megoldásra váró problémát le ell bontani, vissza kell vezetni már megoldott problémákra, illetve az adott probléma megoldását ás problémák ismert megoldásából kell fölépíteni, kikövetkeztetni. A problémamegoldásnak t a módszere ma már szinte kötelező és kizárólagos módszerré, majdhogynem megfellebbezhe-elen hozzáállássá vált. Nem is alaptalanul. Igen kevés az olyan tudományos eredmény, sőt chnikai vívmány is, amelyet alapvetően nem az ilyen módszeres problémamegoldási techniká-ok köszönhetünk. A feltalálás, a megoldásra való rájövés művészetét az emberiség egyre inkább esterséggé változtatja. A művészet végül is megfoghatatlan, a mesterség viszont megtanulható, egyakorolható dolog.

A tudományos és a technika fejlődés problémainváziójával szemben csak a módszeres prob-émamegoldási technika nézhet szembe némi magabiztossággal. A módszeres problémamegoldás élkülözhetetlen kellékei a széles körű szakmai tájékozottság, az alapos tárgyi tudás és a követke-tes gondolkodási képesség.

Az *intuitív problémamegoldás* az ember olyan titokzatos és mégis természetes képességén apul, amely a probléma megoldásához pusztán belső szemlélődés útján vezet el, minden tudatos ondolkodás és célratörő logika nélkül. Az ember képes ösztönösen és hirtelen ráeszmélni a meg-idásra, a dolog mélyére látni, ráérezni a probléma nyitjára. Az intuitív problémamegoldási fo-yamat természetéből kifolyólag nem tanulmányozható, és így nem is tanulható meg, nem gya-orolhatóbe. Ilyen értelemben az intuíció a módszeres problémamegoldásnak mintegy ellentéte, a úgy tetszik, riválisa. Úgy tűnik, hogy manapság a problémamegoldásban egyre kisebb szerep ut az intuíciónak. Munkájuk során egyre kevesebben, és egyre kisebb mértékben hagyatkoznak ermészetes intuíciós képességükre. Tulajdonképpen az intuíciónak ma már csak valamiféle át-idaló, összekötő szerep jut a módszeres gondolkodáson belül. Mégiscsak ösztönös felismerések ellenek ahhoz, hogy a megoldásra váró problémában meglássuk azt, hogy az mely már megol-ott problémákra és miképpen vezethető vissza. Az intuíciónak ennél magasabb szintjét elvárni, agy akár szorgalmazni, sürgetni, előírni ma még nem lehet. Mégis megkockáztatjuk azt, hogy az ntuíció ilyennyire háttérbe szorulása nem a legszerencsésebb dolog. Tudván tudjuk, hogy a tu-omány jóideje szinte reménytelenül keresi számos életbevágó probléma megoldását. Ösztönösen éha azt is érezzük, hogy igen sok probléma megtalált és elkönyvelt megoldása nem tekinthető

igazi és végleges megoldásnak. Valamiképpen felemás az olyan megoldás, amely az adott problémát úgy oldja meg, hogy közben újabb problémákat vet fel. Valahogyan erőszakolt az a megoldás, amely túlontúl bonyolult a probléma egyszerűbbnek érzett természetéhez képest. Valamiért nehézkes megoldás az olyan, amely túlontúl nagy apparátust foglal magában a probléma kisebbnek érzett jelentőségéhez képest.

Az *intuíció eszközeiről, módszereiről* beszélni szinte képtelenségnek tűnik. Az intuíciót viszont erősen motiválhatja az a meggyőződés, hogy a probléma belső természetéhez a puszta szemlélődéssel történő közelférkőzés is igen sokat segíthet a probléma megoldásában. Az ilyen hozzáállás voltaképpen határos az ún. rendszerszemléletű közelítéssel. Az intuíció eszközein konkrétan csupán szempontokat értünk, olyanokat, amelyeket figyelembe véve ez a belső szemlélődés körültekintővé, mélyrehatóvá, vagyis alapossá tehető. Az ilyen modell a problémát egészében, részleteiben és belső viszonyaiban úgy igyekszik leírni, amint az önmagában létezik, vagy amint az önmagától létrejött, lehetőleg tartózkodva minden szubjektív állítástól, problémán kívüli asszociációtól. Aligha vitatható, hogy aki mindig módszeresen gondolkozik a problémák megoldáshoz, az alig képes így szemlélődni, ilyen ontológiai modellféleséget konstruálni. Az intuíciós képesség az emberben a módszeres problémamegoldás gyakorlata során egyre inkább meddővé válik. Az intuíció leghatásosabb ellenszere ezért sajnos éppen a merev ragaszkodás a módszeres problémamegoldás kellékeihez, a széles körű szakmai tájékozottsághoz, az alapos tárgyi tudáshoz és a logikus gondolkodáshoz.

Az előadás olyan megfigyeléseket, tapasztalatokat és következtetéseket vázol fel, amelyeket a szerző zömmel közvetlenül a saját munkája során szűrt le, részben közvetve munkatársa és általában a szakmabeliek munkája nyomán ismert fel. Az előadás egzaktságra egyáltalán nem kíván törekedni, inkább csupán elgondolkoztató, kérdésfelvető szerepre vállalkozik. Munkán itt olyan problémák megoldását kell érteni, amely mindig gyakorlati, viszonylag nagylélegzetű volt és mindig számítógépes megoldásra irányult, konkrétan vállalatok megrendeléséből adódó, nagyméretű és komplex alkalmazási programrendszerek kidolgozását. Éppen az ilyen természetű munkákkal való végeláthatatlan küzdelem, a gyakorlati félsiker, a nem ritka kudarc adta és adja fel a leckét mindannyiunk számára és tereli figyelmünket egyre inkább az intuíció felé. Annál is inkább, mert már a számítástechnika ezen szűkebb területén belül is igaz az, hogy ma már sok munkahelyen szinte reménytelen dolog (és egyre inkább az lesz) egyszerre produkálni is, és szakmai tájékozottságban, tárgyi tudásban a fejlődéssel is lépést tartani.

2. Problémaabsztrakció és technológia

A problémamegoldáson a számítógépes gyakorlatban mindig csak elvont, matematikai probléma megoldását értjük. Az eredeti problémát mindig elvonatkoztatjuk a konkrétumokból és absztrakt problémává alakítjuk át. Az eredetileg is matematikai problémák esetében ez az absztrakció természetszerűleg triválisan adódik. A vállalati alkalmazások területén a valódi probléma ilyen abszrakciója korántsem triviális. Hagyományosan az ún. rendszerszervezés feladata ennek a problémaabsztrakciónak az elvégzése.

A számítástudományban újabban igen sokat foglalkoznak azzal, hogy technológiát dolgozzanak ki a problémamegoldásra. Lévén a számítástudomány mindig is erősen matematikai beállítottságú, szinte kizárólag csak az absztrakt problémák megoldására nézve iparkodik technológiával szolgálni. Mintha a problémaabsztrakció a legegyszerűbb dolog volna! Figyelemreméltó paradoxon, hogy a rendszerszervezők munkáját nem becsüljük valami sokra, a félsikerű vagy éppen balsikerű vállalkozásokért mégis teljes mértékben őket okoljuk. Egyszerűen nincs mit tanítanunk nekik. A rendszerszervező szinte teljesen magára van utalva; vagy ösztönösen jól dol-

ik, vagy csapnivalóan. Ezért is hallatszanak újabban olyan hangok, amelyek szerint a rend-rszervezés ilyen elkülönítése a teljes problémamegoldási folyamatban nem volt szerencsés nyzat. Az érem másik oldala viszont úgy fest, hogy a szinte csak konkrétan gondolkozni tudó lati szakemberek és a szinte csak absztraktan gondolkozni tudó számítástechnikai szakem-k között az érdemi kommunikáció majdnem lehetetlen.

Úgy tűnik, hogy az absztrakt problémák megoldására kidolgozott technológiák – struktú-t programozás, Jackson-féle programtervezési módszer stb. – egyre inkább csődöt mondanak yakorlati alkalmazások területén. Voltaképpen csak azért, mert többet várunk el tőlük, mint t nyújtani tudnak. Ha bármely életből vett problémát elvileg többféleképpen is lehet absztrakt blémává alakítani, akkor ezek az absztrakt problémák nyilván nem egyenértékűek az eredeti blémához viszonyítva. Ugyan akkor, tulajdonképpen alig ismerjük az eredeti problémát, vala-képpen viszolygunk is az eredeti probléma alapos megismerésétől. Csak a valódi probléma ható ismeretei igazít el bennünket abban, hogyan kell azt jól absztrakt problémává átfogal-zni. A közismert technológiák csak a problémaabsztrakció utáni problémamegoldás segédesz-közei. Ugyancsak a valódi probléma körültekintő ismerete igazít el bennünket abban is, hogyan el ezeket a korszerű technológiákat jól alkalmazni.

A vállalati alkalmazások területén szinte kizárólagos az a módszer, hogy a valódi proble-mákat minden további nélkül ún. típusfeladatoknak feleltetjük meg, több kevesebb változtatás-sal. Az így adódó absztrakt problémát azután mindenféle korszerű technológiával igyekszünk megoldani. A megoldásként előállt számítógépes rendszert korszerűnek hirdetjük, és általában nem értjük, miért a gyakori félsiker és kudarc. Ezért erőfeszítéseinket arra összpontosítjuk, hogy típusfeladatok minél teljesebb és tökéletesebb készletét fejlesszük ki. Minden ilyen erőfeszíté-sünk ellenére a helyzet alapvetően nem változik. Fel kellene végre ismernünk, hogy így lényegé-ben mindig valamilyen absztrakt feladatsémát erőszakolunk rá a valódi problémára, a valódi probléma alapos ismerete nélkül. Holott fordítva kellene eljárnunk, behatóan meg kellene ismer-nünk a valódi problémát és – helyes absztakció útján – visszavezetnünk valamilyen absztrakt prob-lémára. Ha az így adódó absztrakt problémát azután valamely ismert típusfeladat valamilyen vál-tozataként kézenfekvően meg tudjuk oldani, akkor megtehetjük, ha nem, akkor ne is kísérel-jük meg. Megkockáztatjuk azt az állítást is, hogy a típusfeladatok alkalmazhatóságának növe-lését célzó erőfeszítéseink a visszás helyzetet csak fokozzák. Még inkább elterelődik a figyelmünk arról, hogy a valódi problémát kellene behatóan ismerni. Ugyanakkor ezek a korszerű típusfel-adatok egyre nehézkesebben alkalmazható és egyre improduktívabb megoldást eredményező eszközökké válik.

3. A probléma megismerése

A számítástudomány és technika a gyakorlati életben fejtegetéseink szerint pusztán azért könyvelhet el magának félsikereket és kudarcokat, azért néz szembe a megoldásra váró gyakorla-ti problémák tömegével szinte már reménytelenül, mert csupán az absztrakt problémák megol-dására koncentrál és csupán az ilyen problémák formális megoldására koncentrál. Sokkal nagyobb sikereket könyvelhetnénk el és sokkal kevesebb munkánk adódna, ha ugyanennyire koncentrál-nánk a valódi probléma megismerésére is, és az így megismert probléma tulajdonságaitól vezet-tetve hajtanánk végre a probléma absztrakcióját is és alkalmaznánk a rendelkezésünkre álló kor-szerű technológiai eszközöket is.

A probléma és a feladat között általában nem szokás különbséget tenni. Úgy tűnik, hogy köztük mégis van különbség és nem kicsi.

A probléma általánosságban nem más, mint diszharmónia az ember céljai (törekvései, vágyai, álmai) és adottságai (lehetőségei) között. A probléma akkor és úgy születik, amikor és ahogyan az ember ezt a diszharmóniát felismeri, és egyszersmind megszüntetendőnek érzi. A probléma leglényegesebb eleme az akadály, a célok és az adottságok között, a kényelmetlen alapérzés megszűnik. És ez nemcsak az ember magánéletében van így, pontosan ez igaz a vállalati szervezetek életére nézve is.

Az igazi probléma azért ismerhető meg nehezen, mert benne csupán a kényelmetlen alapérzés biztos, de már a célokat, az adottságokat teljességükben nem ismerjük. Az ember sokszor sem az adottságait nem érzékeli, sem a céljait nem tűzi ki reálisan. Valójában csak annyit akar, hogy szabaduljon a kényelmetlen alapérzéstől, lehetőleg gyorsan. Ez a törekvése idővel feladattá fogalmazódik benne, amelyben már mind a céljait, mind az adottságait objektívnek tartja. A feladatot azután valamiképpen megoldja, végrehajtja, kitűzött célját eléri. Hány esetben fordul elő az, hogy mégis úgy érzi, a probléma részben vagy egészben megmaradt, a kényelmetlen alapérzés alapjában nem változott. És ez lényegében így van a vállalati szervezetek életében is.

A probléma és a feladat ilyen konfliktusa jogossá teszi, hogy a két fogalmat élesen megkülönböztessük, egyszersmind azt is, hogy az alábbi következtetéseket levonjuk:

- Mindenkor a problémát kell megoldani, nem a feladatot. A probléma valamiképpen objektív, a feladat már nem szükségképpen az. Az objektívnek vehető kényelmetlen alapérzést kell megszüntetni, nem a szubjektívnek is felfogható feladatot kell minden további nélkül végrehajtani.
- A probléma tisztázásához úgy juthatunk el, hogy az adottságokat és a célokat megkérdőjelezzük és megtisztítjuk az irreális vonásoktól.
- Mennél jobban megismertük a valódi problémát, annál közelebb vagyunk az igazi feladathoz. Az igazi feladat és csakis az igazi feladat megoldása vezet el a probléma megszűnéséhez, a diszharmónia feloldásához. Végtére is ez a cél, nem más.

Egy ismert amerikai példa: egy szálloda tulajdonosa azt a megbízást adta egy vállalkozónak, hogy szereltessen be hozzá még egy liftet, mert a vendégei sokat méltatlankodnak a hosszas várakozás miatt. A vállalkozó megoldása, ami a vendégek méltatlankodását valóban meg is szüntette, a következő volt: újabb lift beszerelése helyett szereltessen fel a szállodatulajdonos szinteként a liftajtó közelében egy-egy hatalmas tükröt.

4. Ontológiai problémamodell

Az ontológiai problémamodell (vagy feladatmodell) lényegében az ún. rendszerkoncepció vagy az ún. feladatspecifikáció azon része, amely a probléma megértését, tisztázását célozza. Az ontológiai problémamodell megfogalmazása iránt tartalmi és formai követelményeket lehet támasztani.

A formai követelmények alapvetően a modell egészére vonatkoznak és az ún. modellfilozófia jellegét adják.

- A modellnek elsősorban azt kell tükröznie, hogy a probléma a gyakorlati életben keletkezett. A problémát úgy érzékeljük, hogy beszélni kezdünk róla. A modell legfontosabb követelménye a szöveges megfogalmazás, olyan szöveggel, amellyel ott beszéltek és beszélnek róla, ahol keletkezett. Az alkalmazási rendszerek területén különösen igaz, hogy a probléma nem képletek, ábrák, sémák, absztrakciók stb. formájában születik. Fontos kellék az eredeti szóhasználat, a műszavak (terminus technikusok) átmentése, jelentéskörének rögzítése. Mindaddig nem értjük igazán a problémát, amíg nem tudunk róla úgy beszélni a megrendelővel, hogy a szavainkat és szófűzésünket egészen termé-

szetesnek vegye. Azt a bizonytalanságot (redundanciát, inkonzisztenciát) is le kell rögzítenünk, amelyet a probléma meghallgatásakor észlelünk. Mindaddig könnyen félreérthetjük a problémát, amíg ezeket a bizonyta lanságokat a megrendelőtől függetlenül, magunk akarjuk kiküszöbölni. A probléma első meghallgatásakor aligha lehet rögtön különbséget tenni lényeges és lényegtelen vonások között, különben könnyen félremagyarázhatjuk a problémát. Megkockáztatjuk azt a — már misztikába hajló — állítást is, hogy a bizonytalan problémafelvetés eredeti szókincsében, szókapcsolataiban inkább ott lappang a probléma igazi megoldása, mint a feladat határozott előadásában.

- A modellnek másodsorban azt is vissza kell tükröznie, ahogyan mi az előadott problémát megértettük. A szöveges megfogalmazásban megkockáztathatjuk azt is, hogy új műszavakat vezessünk be és határozzunk meg. Ha ezeket a megrendelő szinte észrevétlenül elfogadja és használja is, és ugyanakkor ezek a probléma szöveges megfogalmazását egyszerűsítik, a probléma tisztázását elősegítik, akkor velük sem idegeníthetjük el a problémát eredeti természetétől. Amint lépésről-lépésre haladunk beljebb a probléma belső világába, azaz vonatkoztatunk el a lényegtelen, a fölösleges vonásoktól küszöböljük ki az ellentmondásos tulajdonságokat, közben észrevétlenül valamiféle fokozatos absztrakciót végzünk. Fontos kellék itt is, hogy az így kölcsönvett absztrakt szókincset és szófűzést mindenkor visszacsatoljuk az eredetihez, életszagú és közérthető megfelelőjüket is kierőszakoljuk magunkból. Igy kisebb a veszélye a probléma torzult absztrakciójának, és nagyobb az esélye a probléma megoldását hozó intuíciónak is.

A modell tartalmi követelményei a modell részeire vonatkoznak és a modellfilozófia eladják. Amolyan alcímek ezek, amelyeket rendre a megfelelő szöveggel kell kiegészíteniük.

- A probléma tárgya. A probléma tárgyát egyetlen mondatban kell megfogalmazni, valahogyan nagyon röviden kell eltalálni a probléma lényegét. Lehet, hogy ez az egyetlen mondat születik meg utolsónak a probléma megfogalmazásában. Csak legvégén jövünk rá, mi a problémában az a 2—3 leglényegesebb dolog, ami egy mondatba belefér. Fontos, hogy a lényegen ne absztrakt dolgot, hanem nagyonis gyakorlati dolgot értsünk.
- Cél. A cél, vagy komplexebb probléma esetében a célok rendszere, mindenképpen a probléma legfontosabb ismérve. A célt sokan összetévesztik az eredménnyel, vagyis azzal a szituációval, ahová el kellene jutni. A célban sokkal inkább a szándékot, a motivációt kell megfogalmazni, amiért egyáltalán valamit tenni kell. Ha egyszerre több célt kell elérni, akkor az egyes célokat egymáshoz képest meg kell vizsgálni.
  A többivel összeférhetetlen célokról tudatosan le kell mondanunk. Az egymást gyengítő, erősítő célokat fontosságuk szerint súlyoznunk kell. Ha kellően nem tisztázzuk a célokat, előfordulhat, hogy előáll az a megoldás, de a probléma részben vagy egészben megmarad, akár még súlyosabb is lehet. Ilyenkor hiába vagy önmagunk ellenére dolgoztunk.
- Jelentőség. A megoldott probléma jelentősége a célok elérésének jutalma, a probléma megszűnésének nyomán remélt harmonikus szituáció. A problémamegoldás jelentősége a célok elérésének jutalma, a probléma megszűnésének nyomán remélt harmonikus szituáció. A problémamegoldás jelentősége is igencsak konkrét dolog, tulajdonképpen ez amolyan gazdaságossági vizsgálat. Számba kell vennünk a mérleg egyik oldalán a beruházás (ráfordítás, befektetés), másik oldalán a haszon tényezőit, és latolgatnunk kell az egyensúlyt (a várható nyereséget). Mindkét oldalon figyelembe kell vennünk a nem mérhető (elvi) és a mérhető (számszerűsíthető) tényezőket, az egyszeri és ismétlődő tényezőket, a megrendelő és a vállalkozó részéről jelentkező tényezőket. Ha az ilyen mérlegkészítést nem hanyagoljuk el, gyakran kiderülhet, hogy a probléma nem is probléma, vagyis megrendelő eláll céljától, vagy esetleg a probléma más, azaz a megrendelő más célt tűz ki.

– Szerep. A – bennünket érdeklő – vállalati alkalmazások területén a megoldott probléma mindenkor egy számítógépes információs rendszerben testesül meg. Ez az információs rendszer nem azonos a kidolgozott programrendszerrel, hanem a programrendszer és az azt üzembeállító, üzemeltető vállalati környezet együttesét jelenti. Nagyon formálisan ez az együttes úgy működik, hogy a vállalati környezet információt ad a programrendszernek, a programrendszer ebből újabb információt állít elő, és átadja a vállalati környezetnek, a vállalati környezet ebből és más információból megint újabb információt ad a programrendszernek. Ebben a kölcsönös kapcsolatban fontos tisztázni a programrendszer szerepét és a vállalati környezet szerepét. Mennyi és milyen munkát igényel a programrendszer a vállalati környezettől ahhoz, hogy működhessen, mennyi és milyen munkát tételez fel a vállalati környezettől ahhoz, hogy működésének eredményeit az valóban hasznosítsa. Fontos tisztázni ezeket a stratégiai kérdéseket a beindulás (üzembeállítás) és a rendszeres üzemeltetés időszakában egyaránt. Sokszor a vállalkozás félsikere vagy kudarca éppen abban keresendő, hogy a megadott probléma várható szerepét előre nem tisztáztuk.
– Problémakörnyezet. A probléma környezete egyrészt maga a megrendelő, úgy amint van. Fontos tudnunk a vállalatról is mindazt, amit az előbbiekben vázoltak szerint a problémáról is tudnunk kell, nevezetesen a vállalat profilját, célját, jelentőségét, szerepét stb. Fontos ismernünk másrészt a probléma szűkebb környezetét is, nevezetesen a kidolgozandó információs rendszer környezetében vagy éppen vele összekapcsoltan működő információs rendszereket is, ugyancsak a fenti szempontok szerint. Mindezen ismeretek birtokában mintegy madártávlatból láthatjuk az adott problémát, rálátásunk lesz rá. Nem békaperspektívából szemléljük, így szemléletünkben a probléma arányai kevésbé torzulhatnak el.
– Krónika. A probléma létrejöttének krónikája, történelmi háttere is fontos szempontokat adhat a valódi probléma megismeréséhez. A probléma soha nem célirányosan születik, hanem az érzékelés ellentmondásos síkján. Az értelem az igazi problémát gyakran eltorzíthatja, és az intelligencia sem védtelen a divatok hatásától. Ugyanezek az ismeretek a problémamegoldást is célirányosabbá tehetik. Megkockáztathatjuk azt az – ismét kissé misztikus – állítást is, hogy a probléma szinte együtt születik a megoldással, csak az utóbbira mindig később eszmélünk rá.
– Perspektíva. A probléma kitűzött célként születik. Az ember az elért célok szintjére föllépve újabb célokat tűz ki. Szűkös lehetőségei miatt egyelőre gyakran redukálja céljait, mennyiségben és minőségben is, talán nem is mindig tudatosan. Messze előre belátni egy probléma perspektíváját persze nem lehet, de annyira szükséges, amennyire azt a megrendelő már ma is magától látja, vagy érzi, sejti. A megrendelő vágyai-álmai nyomán felrajzolható probléma nem szükségszerűen nagyobb igényű, mint a kitűzött céljai nyomán adódó. Néha van olyan érzése az embernek, hogy valamilyen algoritmikus megoldás csak azért olyan bonyolult, mert számos helyen mintegy vissza kell tartani a szabad célratörésben, amint a megrendelő is visszatartotta magát a célkitűzéseiben.
– Esély-veszély. A probléma megoldása végülis egy információs programrendszerben testesül meg. Ezt az objektumot létre kell hozni, és a vállalat információs környezetébe kell szervesen beiktatni. Egy ilyen vállalkozásnak vannak esélyei és veszélyei is, ezeket is előre tisztázni kell. A létrehozás esélyeit a vállalkozó oldalán rendelkezésre álló szellemi és gépi kapacitás, a feladat bonyolultsága, a feladat térbeli és időbeli megoszthatósága, a rendelkezésre álló idő igencsak meghatározza. A megrendelő esetleg csökkenti igényeit, vagy elállhat tőlük, ha elegendő és reális esélyt nem lát a programrendszer létrehozására. A beiktatás esélyeit a megrendelő oldalán ugyanezek a tényezők

határozzák meg. A vállalkozó esetleg visszaléphet a kidolgozástól, ha elegendő és reális esélyt nem lát arra, hogy a programrendszer a vállalatnál valóban működni is fog. A veszélyek fölmérése mindkét oldalon sokkal nehezebb, A létrehozás-beiktatás pontosan megtervezett folyamatában a várható rendkívüli körülményekre is gondolni kell. A létrehozási periódusban ilyen veszély lehet az ember vagy a gép tartós kiesése. Ellene szerepcserés feladatmegosztással, tartalékkapacitással lehet és kell védekezni. A beiktatási periódusban ilyen veszély lehet ezenkívül a meglevő vállalati szervezet váratlan ellenállása az „idegen test" iránt, a megváltozás vagy éppen megnövekedett feladatokból kifolyólag. A veszélyekkel nem számoló vállalakozás rosszabb, mint egy el sem kezdett vállalkozás.

- Előny-hátrány. A vállalati szervezet valahogyan funkcionált a programrendszer üzembeállítása előtt is, utánna is valahogyan (nyilván kicsit másképpen) funkcionálni fog.
  Az adott probléma viszonylatában az előző és a jövendő helyzet összehasonlítása elengedhetetlen, mindkettőnek természetszerűleg vannak konkrét előnyei is és konkrét hátrányai is.
  Gyakran egy vállalkozás kudarcát az okozza, hogy a vállalat vezetősége az új rendszertől egyszerűen csodát remél, hozzá illuzórikus reményeket fűz. A problémák még számítógépes megoldások ellenére sem változtatják meg azt az alapvető természetüket, hogy bármely probléma megoldása nyomán újabb problémák születnek. A problémákat végülis az ember maga csinálja magának, mert ilyen a természete. A megoldott probléma soha nem jelent számára igazi harmóniát, bár mindig azt reméli. A vállalkozás ilyen előnyeit és hátrányait előre be lehet látni, és tudatosítani is kell.
- Input-output. Gyakori az olyan problémafölvetés a vállalati alkalmazások területén is, hogy az adott információkból újabb információkat kell előállítani. A helyes problémafölvetés során a célból következik a szükséges eredmény (output) és az outputból a szükséges input. Az input és az output részletes leírása rendszerint csak az absztrakt adatok leírására korlátozódik, és ritkán tér ki az adatok absztrakciójának a leírására. Az absztrakció eredményét részletesen leírják, de szinte semmit sem foglalkoznak a konkrét adatok leírásával és az absztrakciós folyamat leírásával. A konkrét adatok jellemezhetők az egyes értéksorozatok változékonyságával, értékhatáraival, az egyes értéksorozatok (pl. rekordok) várható számosságával, a számosság változékonyságával, a tilos értékkombinációkkal, az egyes értékrendszerek (pl. gráfok) átlagos és szélsőséges kiterjedési méreteivel, a tipikus és szélsőséges adatkonstellációk körének eseteivel, az egyes esetek jellegzetességeivel stb. Az absztrakciós folyamat jellegzetességeivel stb. Az absztrakciós folyamat jellemezhető az objektíve létező és a szubjektíve létrehozott kapcsolatok jelzésével, több lehetséges adatabsztrakció mérlegelésével stb. Kis túlzással azt mondhatnók, hogy mielőtt döntünk a legkézreesőbb absztrakt séma mellett, azután már mindenképpen csak benne gondolkozunk, sőt már a céljainkat is rajta keresztül fogalmazzuk meg.
- Eljárás. Gyakori az olyan problémamegfogalmazás is, amely szinte semmi közelebbit nem árul el a lehetséges eljárásból, legföljebb csak az eljárás külső-belső feltételeit írja elő. A vállalati szakemberektől pedig gyakran meg lehet tudni, belőlük ki lehet erőszakolni olyan ötleteket, belső összefüggéseket, akár algoritmusvázlatokat is, amelyek sokat könnyítenének a megoldás kitalálásában, az algoritmus tervezésében és egyszersmind hatékonyabb megoldást eredményeznének. Kis túlzással azt is mondhatnók, hogy az igazi megoldás is az utcán sétál, csak fel kell ismerni, akár a körözött tettest a mozaikképe alapján.

5. Konklúzió

Az intuíció szerepét a problémamegoldásban úgy összegezhetnénk, hogy ott, akkor és annyiban van rá szükség, ahol, amikor és amennyiben a mesterségbeli tudás egy kicsit is bizonytalanná válik. Bármelyik emberben eredendően megvan rá a képesség, legföljebb a nevelés-tanítás-kiképzés fokozatosan elnyomja benne, az általános mesterség-szemlélet nem hagyja hinni benne.

Az intuíció kiválasztására igazi eszköz nem létezik. A valódi probléma alapos megismerése azonban az intuíció táptalaja lehet. A probléma ilyen megismerésére vonatkozólag lehet segédeszköz az ontológiai modell.

Az intuíció szerepének fokozása a problémamegoldásban egyrészt a probléma-fölvázolás és a problémamegoldás szellemi erőfeszítéseit csökkentheti, másrészt sikeresebb vállalkozásokat garantálhat, rövid és hosszú távon egyaránt.

A SIMULA 67 ÉS A SZÁMÍTÁSTUDOMÁNY

Gáspár András
MTA SZTAKI

Bevezető

A SIMULA 67 az ALGOL nyelvcsalád tagja, tartalmazza a teljes ALGOL 60 nyelvet *string, own* nélkül). Mégis félrevezető állítás, hogy az ALGOL 60 kiterjesztése, mivel több igen jelentős konstrukciót, illetve modellt vezetett be.

A SIMULA 67 nyelvi eszközökkel kora igen nagyszámú számítástudományi problémájá- kívánt választ adni és egy ma is élő számítástudományi irányzat, a „skandináv iskola" mag- Fogalmainak, eszközeinek alapos ismerete ezért a „számítástudományi általános műveltség" része. A SIMULA 67 azonban nemcsak tudásgyarapító elképzelés-javaslat, hanem egyenként kb. 10–20 emberévnyi munkával már eddig is több géptípusra implementálták (IBM 360/370; SIEMENS BS 2000; ICL SYSTEM–4; CDC 3300/3500, CDC 6000, CDC CYBER; CII–IRIS 80; UNIVAC 1100; UNIVAC 90/80; DEC 10, DEC 20).

A SIMULA 67 nyelv tervezése [1] során (1962–68) a kor valamennyi programozási és szimulációs nyelvét kritikai vizsgálat alá vették.Végülis 1967-ben a Norwegian Computing Center a „Common Base Conference" elé terjesztette az eseményközéppontú szimuláció közös alapnyelvére tett javaslatát. A jelentős nemzetközi résztvevő gárda tulajdonképpen az esemény- középpontú szimulációs nyelvek konstruktőreiből, implementálóiból és fő felhasználóiból tevő- dött össze. A konferencia javaslatára komoly változtatások is születtek egyévi kemény munka során (szövegkezelés, input-output kezelés). A nyelvfejlesztés végeredményét [2] ismerteti.

A tervezők a 60-as évek számítástechnikájának egyik mellékösvényén, a szimulációs nyelvek nyomán jutottak el az eseményközéppontú szimuláció közös alapnyelvéhez, de „túl- lőttek a célon": A közös alapnyelv annyira jól sikerült, hogy nem szimulációs célokra is hasz- nálhatónak bizonyult. Kitűnően illusztrálják ezt az eddig megtartott 7 SIMULA Konferencia és a 8 Workshop anyagai, címei, tematikái; továbbá az 1973 óta negyedévenként megjelenő folyóirat [3] rövid közleményei.

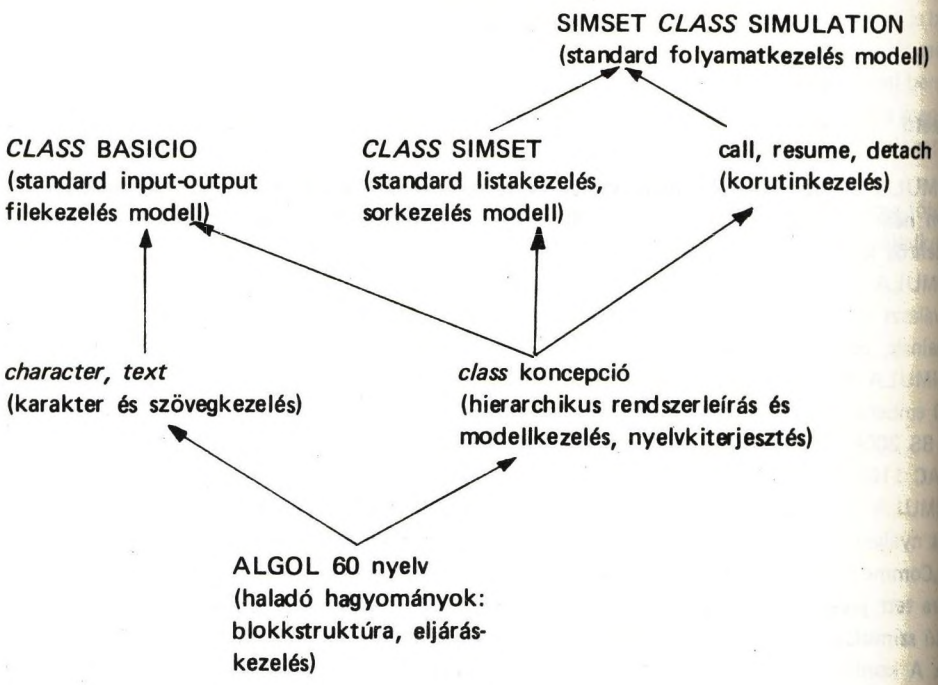
A SIMULA 67 nyelv felhasználóinak nemzetközi társasága az Association of SIMULA Users (ASU). Öt kontinens 32 országából kb. 500 tagja van Kiadványa [3]. Rendezvényei: ASU Conference, ASU Workshop. Bizottságai: a SIMULA Development Group (SDG) és a SIMULA Standards Group (SSG). Az SDG a nyelvfejlesztők tudományos, az SSG pedig első- sorban az implementálók technikai bizottságának tekintendő. Az SDG az „alsóház", az SSG pedig a „felsőház". Az SSG a portabilitás őre.

A továbbiakban gyakran hivatkozunk a SIMULA Newsletterben megjelent közlemények- re. E hivatkozások alakja: (intézmény–évfolyam/szám . oldal)

1. A SIMULA 67 és a szimulációs nyelvek

A SIMULA 67 nem szimulációs nyelv, hanem a szimuláció közös alapnyelve. Kapcsola- tát a szimulációs nyelvekkel ezért legjobban azzal jellemezhetjük, hogy milyen nyelvi eszközö- ket szolgáltatott a sokféle szimulációs nyelv (GPSS, DYNAMO, CSL, ...) kiváltására és milyen nyelvi konstrukciókat ajánl a szimulációs „nyelvecskék" (programcsomagok) elburjánzása helyett.

A SIMULA 67 nyelv szerkezete jól tükrözi eszközeit:



Az alábbiakban a *class* koncepció egy-egy alkotóeleme után zárójelben utalunk annak legértékesebb felhasználási területeire:

- *class* deklaráció = adatstruktúra + eljárás gyűjtemény + forgatókönyv deklaráció (rendszer illetve komponensosztály leírása; modelldefiníció, modellbeágyazás, nyelvkiterjesztés)
- Lokális *class* deklaráció (rendszer-alrendszer hierarchia illetve lokális modelldefiníció)
- *class* deklaráció formális paraméterei (rendszerek illetve komponensek formális paraméterei, modellek formális paraméterei)
- *class* deklarációban deklarált adatstruktúra (rendszerek illetve komponensek állapothatározói, modellek állapothatározói-változói)
- *class* deklarációban deklarált eljárásgyűjtemény (rendszerek illetve komponensek funkciói, modellek funkciói-eljárásai)
- *class* deklarációban deklarált forgatókönyv (rendszerek illetve komponensek életműködése, modellek működése)
- *new* generátor utasítás (új rendszerpéldányok, komponenspéldányok, illetve modellpéldányok létrehozása)
- *ref* típusú változók, *this* hivatkozás (rendszerpéldányra, komponenspéldányra, illetve modellpéldányra történő külső és belső hivatkozás)
- *:–none* utasítás (rendszerpéldányra, komponenspéldányra, illetve modellpéldányra vonatkozó hivatkozás megszüntetése. A hivatkozásnélkülivé vált objektumokat a garbage collector kisöpri a memóriából és újra felhasználhatóvá teszi az általuk lefoglalt memóriát)

távoli elérés, *inspect* utasítás (rendszerpéldányok, komponenspéldányok illetve modellpéldányok közötti interakciók leírása)
részosztálydeklaráció, *inner, virtual,* prefixesblokk (modell specializálás, modellhierarchia definiálás)
szeparált fordítás, *external class* deklaráció (modell-könyvtárkialakítás)
protected és *hidden* specifikáció (modellek integritásvédelme)

A SIMULA 67 nyelv konstrukcióinak és modelljeinek hatékonyságát a szimulációs beszúrások területén mi sem bizonyítja jobban, minthogy a legelterjedtebb szimulációs nyelvnek, a GPSS nyelvvel egyenértékű *class* implementálása mindössze 700 soros SIMULA programot jelent.

A SIMULA 67 nagyszámú külföldi és hazai szimulációs alkalmazását egy másik írásban ismertettük át [9].

2. A SIMULA 67 és a programozási nyelvek tervezése

A SIMULA 67 nyelv ötletei, konstrukciói, modelljei – de főként a *class* koncepció egy sor programozási nyelv tervezőinek adtak ihletet. Például: ALGOL 68, PASCAL [5], MODULA [6] CLU [7], ALPHARD [8]. E nyelvek tervezése néhány elv következetes végigvitelét jelentette (Techn. Univ. of Denmark 1977/3. 24). Például: ortogonalitás, hatékonyság, rendszerűség, rendszerprogramozási lehetőség, modellintegritás, verifikálhatóság. Sajnos az elvek végigvitele rendszerint más elvek rovására történt. Például az ALGOL 68 tervezése közben figyelmen kívül maradt a SIMULA 67 rendszerleíróképessége és modellkezelése. Nem elvetették, sőt nem csupán észre sem vették ... Pedig a rendszerleírás képessége és a hatékony modellkezelés minden korszerű programnyelvtől joggal elvárható. Később voltak is „szervátültetési" javaslatok [4].

Ez a magyarázata annak, hogyha ma egy bizottságnak lenne bátorsága megalkotni a kor programnyelvét, akkor munkáját célszerű lenne az ALGOL 68 és a SIMULA 67 keresztezésével kezdeni, és a többi felsorolt nyelvet, illetve a szakirodalomból ismert konstrukció ötleteket csupán az így született új nyelv bírálatának tekinteni.

A SIMULA 67 tervezési értékei közül mindig kiemelik a kiterjeszthetőséget (Univ. of Oxford 1976/2. 17), a kifejezőerőt (Univ. of Minnesota 1975/2. 8), a megbízhatóságot [10].

Tervezési hibái között említik a defenzív programozást nehezítő *inspect* utasítást, illetve a szintenkénti tervezést nehezítő „S" szabályt.

Több ma is vitatott/kutatott SIMULA nyelvfejlesztési elképzelésről tudunk:
- objektumok standard inputja-outputja
- standard tasking [11], save-restore-kommunikáció (1975/4.8; 1977/1.16; Norwegian Computing Center 1978/2.10; MTA SZTAKI 1978/2.3)
- a context-koncepció (SDG 1979/3.3)

3. A SIMULA 67 és az erőforrásokért versengő folyamatok

A folyamat szinkronizálást gyakorlati szempontból is jól megoldó Hoare-féle monitor definíciója még alakjában is hasonlít a SIMULA 67 osztálydeklarációhoz.

A Petri-hálók gyakorlati szempontból fontos kiterjesztései egyszerűen vizsgálhatók SIMULA nyelven írt szimulátorok segítségével (Gesellschaft für Konforschung-Karlsruhe 1976/3.13)

A DEMOS osztály erőforrásokért versengő folyamatok szimulációjának oktatására k
Közvetlen előzményei voltak az erőforrásdiagramok (1976/1.4) és a SIMON '75 (1976/4.6
Négy fajta folyamatszinkronizálást kezel (Univ. of Bradford 1976/3.12; 1979/1.4):

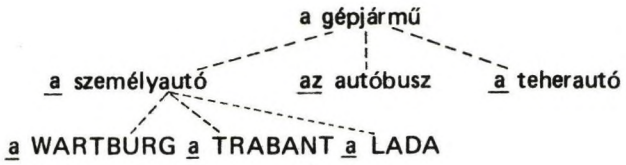
- kölcsönös kizárás (mutual exclusion): az erőforrást legfeljebb egy folyamat használhatja
- termelő-fogyasztó (producer-consumer): amit az egyik folyamat termel, azt a másik folyamat elfogyasztja
- hívó-hívott (master-slave): a hívó a hívás kibocsájtása után addig várakozik, míg a hívott szabaddá nem válik. Ezután a hívott alárendelt viszonyba kerül az együttműködés idejére
- feltételes várakozás (wait until): a folyamat felfüggesztődik addig a pillanatig, ameddig az adott feltétel nem teljesül.

4. A SIMULA 67 és a struktúrált programozás

Nem térünk ki az olyan közismert vonatkozásokra, melyet magyar nyelven hozzáférhető könyv ismertet [13]. Viszont ki kell emelnünk, hogy a SIMULA 67 programozó 3 fajta hierarchiával találkozik:

- *A rendszer-alrendszer hierarchia*

- *A modellhierarchia*



- *A nyelv belső fogalomhierarchiája* (lásd a nyelv szerkezete). [14] szerint a struktúrált programozás elveinek legjobban megfelelő nyelv a SIMULA 67, mivel:
- Az adatok és programok struktúrálása ugyanazon koncepció szerint történhet (*class* koncepció). Az adatok és rájuk vonatkozó műveletek együtt definiálhatók.
- A program modulokra osztható, világos, jól definiált és jól ellenőrzött modulok közötti kapcsolatokkal.
- Lehetővé teszi a koncepciók szintenkénti részletezését.
- Teljesen biztonságos. Nincs olyan program, melynek futása ne a nyelv definícióját követné.
- Hatékony és kompatibilis SIMULA-implementációk léteznek a legjelentősebb géptípusokra.

5. A SIMULA 67 és a számítógépes grafika

A szövegkezelés, filekezelés és a modellkezelés felhasználásával elért eredmények:
Perspektíva rajzolórendszer építészek számára
..of Trondheim 1973/3.6), *class* GRAPHIC (Academic Center Utrecht 1977/1.9;
..4.19), Moving pictures show simulation to user (Swedish National Defense Research
..1973/3.15), ICGL—Interactive Computer Grapics Language, DEC GT—42 graphic display
..onal Lab. of Civil Eng. Lisbon 1978/3.9).

6. A SIMULA 67 és a nyelvek, compilerek, interpreterek

A témakörről 1978-ban Workshop volt „SIMULA and Compiler Writing" címmel
..wegian Computing Center 1978/1.18). További közlemények:
SIMULÁ-ban tervezett compiler (1976/3.11; Texas Univ. 1978/1.3), rendszerprogramo-
.. nyelv (Univ. of Aarhus 1978/1.18)
SIMULÁ-ban implementált SIMULA (Univ. of Oslo 1977/3.24), LL (1) -compiler gene-
.. (Univ. of Simon Bloivar 1978/1.13), CDL (INFELOR 1976/3.11), preprocessor (Univ. of
..dheim 1977/2.4), makroprocessor (Univ. of Karlsruhe 1975/1.6).

7. A SIMULA 67 és az adatbáziskezelőrendszerek, információs-rendszerek

A témakörről 1977-ben Workshop volt „SIMULA and Data Bases" címmel (Univ. Pierre
..arie Curie 1977/3.11). További közlemények:
SIMULÁ-ban információs rendszer tervezés (Univ. of Groningen 1977/3.21; 1977/3.23)
Adatbáziskezelő szimuláció (Univ. of Calgary 1977/2.10; Royal Ins. of Technology,
..ckholm 1977/1.5)
SIMULÁ-ban implementált adatbáziskezelő rendszerek: CODASYL típusú — SIMDBM
..edish Nat. Def. Res. Ins. 1975/3.9), relációs — ASTRA (Univ. of Trondheim 1977/3.23),
..gráf alapú (Univ. Pierre et Marie Curie 1976/3.14)
SIMULÁ-ban implementált egyedi adatbázisok: utak geom. leírása a Norvég Útügyi
..gatóság számára (NCC 1974/3.19)
SIMULÁ-ban implementált információvisszakereső rendszer könyvtárak számára, illetve
..bványok kezelésére — SARI (NCC 1973/2.19)

8. A SIMULA 67 és az operációs rendszerek

A témakörről 1976-ban Workshop volt „DESIGN AND TUNING OF OPERATING
..STEMS" címmel (Technical Univ. of Denmark 1976/3.4). További közlemények:
Tervezés SIMULÁ-ban (DEC 1975/4.21; OLIVETTI 1977/3.20)
Implementálás SIMULÁ-ban (DEC 1975/4.21)
Szimuláció és tuning SIMULÁ-ban (Univ. Pierre et Marie Curie 1975/3.7; Univ. of Cal-
..y 1975/3.8; 1977/2.10; DEC 1975/3.8; Univ. of Bonn 1976/3.4; NCC 1976/3.4; Univ.
..re de Bruxelles 1977/3.7)
SIMON—75 (1975/3.11; 1976/4.6; 1977/2.4),
DEMOS (Univ. of Bradford 1979/1.4)

A témakörről 1976-ban Workshop volt „SIMULA and Real Time" címmel (Swedish Def. Res. Ins. 1977/1.5). További közlemények:

SIMULA real-time kiterjesztése (Swedish Nat. Def. Res. Ins. 1974/3.16; 1976/2.9; 1977/1.5; Univ. of Oslo 1978/3.16)

Tervezés, szimuláció, tuning SIMULÁ-ban (Swedish Nat. Def. Res. Ins. 1977/3.12)

Implementálás SIMULÁ-ban (Univ. of Oslo 1978/3.13)

SIMULÁ-val generált környezet tesztelés céljára (NCC 1974/4.3; Univ. of Oslo 1978

10. A SIMULA 67 és az interaktivitás

A témakörről megjelent legfontosabb közlemények:

Beszélgetős szimulációról (Swedish Nat. Def. Res. Ins. 1975/3.10), interaktív input-output-ról (SHAPE Technical Centre 1974/2.10), hibatűrésről (Swedish Nat. Def. Res. Ins. 1975), interaktív hibajavítás – SIMDDT (Swedish Nat. Def. Res. Ins. 1977/2.13), Lásd még 9.és 11. pont alatt.

11. A SIMULA 67 és a terminálhálózatok, számítógéphálózatok

A témakörről 1978-ban könyv [15] jelent meg. További közlemények:

Tervezés, szimuláció, implementálás (Karlsruhe Nuclear Res. Center 1974/2.7; 1974/2.8; Univ. of Calgary 1976/3.13); Royal Ins. of Technology, Stockholm 1977/1.5; Univ. of Oslo 1978/3.13)

Az MTA tervezett számítógéphálózatával kapcsolatos szimulációs vizsgálatokról szól [16, 17] és [18].

12. A SIMULA 67 fordító és futtató programok minősége

Valamennyi SIMULA 67 fordító és futtató program szerkezetét, algoritmusát a „SIMULA 67 Implementation Guide" definiálja. A különböző gépeken megírt programokra ezért nagyfokú portabilitás jellemző.

A SIMULA 67 rendszerek minőségvizsgálata természetes igényként jelentkezett a korábbi PL/1 tapasztalatok nyomán, de azért is, mert a nyelv definíciója igen nagy tudást vár el a kísérőprogramtól (rutime system) és a hulladékgyűjtőtől (garbage collector).

Az IBM 360/370 SIMULA benchmark kiértékelését J. Palme végezte el és táblázatokat közölt a FORTRAN G, a PL/1–F, a PL/1 optimizer, az ALGOL W, a COBOL F és az ALGOL F fordító és futtató programokkal történt összehasonlításról (Swedish Nat. Def. Res, Ins. 1974/2.3). A mérések eredményeit magyar nyelven részletesen közli [12]. A kiértékelés eredménye: a SIMULA 67 nemcsak jól tervezett, értékes konstrukciókat tartalmazó nyelv, de rendkívül megbízható és rendkívül hatékony is.

Minden gépen a forrásprogramsor sorszámára hivatkozik az egyébként jól érthető – pontán a SIMULA fogalmaira támaszkodó – futás közbeni hibajelzés. A futás mindig zárójelentéssel végződik, melynek tartalma: időfelhasználás, memóriakihasználtság, file-állapotok.

Több minőségjavító fejlesztésről/kutatásról tudunk:

memóriahiány elkerülése [11]:
integer procedure freemem;
A hulladékgyűjtő meghívása nélkül rendelkezésre álló szabad memóriaterület mérete.
integer procedure compress;
A hulladékgyűjtőt meghívja, majd a hulladékgyűjtés utáni freemem értékét adja
memória hiány lekezelése:
procedure nospace (label);
Ha helyfoglalás közben nincs memória, akkor a vezérlés a kijelölt cimkére kerül. (SHAPE Technical Centre 1974/2.10)
TASK és RUN TIME LIBRARY kezelés [11] (MTA SZTAKI 1978/2.3)
hibatűrő input-output kezelés (SHAPE Technical Centre 1974/2.10), interaktív célra - SAFEIO (Swedish Nat. Def. Res. Ins. 1975/3.13)
nyomkövetés (Univ. of Oslo 1974/3.14; State Univ. of Groningen 1978/1.7)
interaktív poloskairtó (debugging) rendszer-SIMDDT (Swedish Nat. Def. Res. Ins. 1977/2.13).

A SIMULA nyelv alkalmazása megsokszorozza a software gyártás termelékenységét. Szerepet játszik ebben a kiterjeszthetőségén, kifejezőerején, megbízhatóságán, hatékonyságán és hibakezelésén túl az a hatás is, amelyet a felhasználók gondolkodásának átalakításával ér el. Sajnos a termelékenységnövelő hatásról nem állnak rendelkezésre objektív mért adatok, csupán szubjektív becsült adatok, mivel nagyméretű programrendszereket nem szokás csupán a mérés kedvéért két nyelven is megírni. Adataink csupán olyan átírásokról vannak, ahol a feladat módosított a korábbi tapasztalatok alapján és az átírást végző programozók már tapasztaltak vagy éppen tapasztalatlanok. A termelékenységnövekedés szorzótényezője függ a feladattól, a programozó szakértelmétől, gyakorlottságától, személyiségétől stb.

[1] K. Nygaard – O.J.Dahl: The Development of the SIMULA Languages. ACM SIGPL Notices. Vol. 13. No. 8. August 1978. pp. 245-272.
[2] O.J.Dahl – B.Myhrhaug – K.Nygaard: The SIMULA 67 Common Base Language. Norwegian Computing Center, 1970. No. S–22.
[3] SIMULA Newsletter. Norwegian Computing Center, 1973-1979.
[4] C.H.Lindsey: Modals. ALGOL BULLETIN, 37. 4. 3.
[5] K.Jensen – N. Wirth: PASCAL User Manual and Report. Springer Verlag, 1974.
[6] N.Wirth: MODULA: A Language for Modular Programming. Software Practice and Experience, Vol. 7. 1977. pp 3-35.
[7] B.Liskov – S.Zilles: An approach to Abstraction Proc. of a Symposium on Very Hi Level Languages, ACM SIGPLAN Notices, Vol. 9. No. 4. April 1974.
[8] W.A.Wulf – R.L.London – M.Shaw: Abstraction and Verification in ALPHARD: In duction to Language and methodology. Techn. Report Carnegic – Mellon Univ., Pittsburgh, 1976.
[9] Gáspár A. – Csáki P. – Visontay Gy.: A szimulációs módszer és a SIMULA 67 nyel Rendszerelmélet '79 Konferencia.
[10] J.Palme: SIMULA–A Language for reliable programs. Swedish Nat. Def. Res. Ins.
[11] Gáspár A.– Visontay Gy. – Csáki P.: A CDC 3300 SIMULA/MASTER új eszközei fel nálói taskrendszerek számára. MTA SZTAKI, CDC 3300 Felhasználói Ismertetők 11,
[12] Dervaderics K. – Sághy A. – Soós K. – Széplaki Á.: Operációsrendszerek és fordítóp ramok minőségvizsgálata. SZÁMKI Közlemények, 1978. 18.
[13] Dahl – Dijkstra – Hoare: Struktúrált programozás. Műszaki Könyvkiadó, 1978.
[14] DEC SYSTEM–10 SIMULA GAZETTE. Editor: J. Palme. Swedish Nat. Def. Res. Ins. Vol. 2. No. 4. April 1976.
[15] Computer Notworks and Simulation. Editor: Schoemaker. North-Holland Publishing Company, 1978.
[16] Gáspár A. – Kocsis J. – Lamm P. – Visontay Gy.: Az MTA tervezett számítógéphálóz nak minőségvizsgálata, Információ Elektronika 1978/4. 261-265. o.
[17] Gáspár A. – Kocsis J. – Lamm P. – Visontay Gy.: Az MTA tervezett számítógéphálóz val kapcsolatos tervezési szempontokról. MTA SZTAKI, Tanulmányok 1978/87. 179-2
[18] A.Gárpás – P.Lamm: Simulation of packet-switched comminication network. ACM SIGCOMM, Computer Communication Review. Vol. 8. No. 4. 1978. pp. 19-29.
[19] G.Birtwistle, O.J.Dahl, B.Myhrhaug, K.Nygaard: SIMULA *begin,* Auerbach – Student litteratur, 1973.
[20] W.R.Franta. The Process View of simulation. North-Holland, The Computer Science Library, 1977.

VIDEOTON RPT 80 FOLYAMATTERMINÁL DECENTRALIZÁLT FOLYAMATIRÁNYÍTÓ RENDSZEREKBEN

Gáti Rudolf

VIDEOTON FEJLESZTÉSI INTÉZET

A mikroprocesszorok megjelenése a számítástechnika szinte minden területén érezteti hatását, de alkalmazása különösen azokon a területeken járt együtt mélyreható változásokkal, lehetőségileg új alternatívák megjelenésével, ahol a számítógépek hagyományos architektúrája és üzemeltetési feltételei korlátozó kényszerűen figyelembe veendő tényezőt jelentettek.

A számítógépek magas ára miatt a számítógépes folyamatirányítás csak ott lépett az olcsó, egyedi szabályozóberendezések helyébe, ahol a nagyszámú folyamatjel kezelése ill. az ehhez kapcsolódó bonyolult feldolgozás a számítógép alkalmazását gazdaságossá tette. Az ilyen, centralizált irányítási rendszereknél nehézséget jelent a számítógép meghibásodásával járó üzemzavar, amelynek hatását csak jelentős költségnövelő – pl. kétgépes rendszer – vagy az irányítási folyamat hatásfokát csökkentő intézkedésekkel lehet enyhíteni.

A mikroprocesszora alapozott rendszerek megjelenése lehetővé tette a korábbi irányítási struktúra több szintre való tagolását, ily módon a korábbi komplex rendszer elemei egy decentralizált – osztott intelligenciájú – irányítási rendszer különböző szintjein jelennek meg, ahol a mikroprocesszor az irányítás legalsó, a folyamattal közvetlen kapcsolatban levő szintjén, esetleg az e fölött levő szinten kap szerepet. Az így kialakított rendszer számos előnyös tulajdonságokkal rendelkezik. A decentralizált folyamatirányító rendszerek alsó szintjén jelentkező feladatok megoldásának hatékony eszköze a VIDEOTON RPT 80 folyamatterminál.

A berendezés az INTEL 8080 mikroprocesszorra, ill, az ehhez tartozó LSI áramkörcsaládra épül. Moduláris felépítése, széles körű elemválasztéka révén sok feladat megoldásához optimálisan illeszthető. Alkalmas önálló feladat megoldására, de folyamatterminálként nagy területen szétszórt folyamat jeleit gyűjtő, feldolgozó, beavatkozó jeleket kiadó terminálhálózat elemeként, fölérendelt számítógéppel együttműködve is; utóbbi esetben a számítógép mentesül a folyamatperiféria fizikai kezelésétől és az irányítás magasabb szintű feladataival foglalkozhat. A terminál használatával – azaz a folyamathoz való közeledéssel jelentősen csökken a kábelezési költség és nem utolsó sorban növekszik a rendszer megbízhatósága, mert a központi számítógép hibája esetén az egyes részterületek irányítását végző terminálok működésében nincs fennakadás.

A készülék felépítése

Alapegység: Felépítését tekintve a berendezés alapegysége a mikroprocesszoros központi egységet, valamint a megszakítás-, és DMA kezelő egységet tartalmazó CPU kártyát, a real-time órát, (RTC), a tápegységet és a további egységek befogadására alkalmas MIKROBUS-t jelenti, amely fizikailag egy csatlakozókkal ellátott hátlap.

A tápegység a működtetéshez szükséges feszültségeket két lépésben állítja elő. A hálózati feszültségből egy 24 V-os egyenfeszültségű sínt hoz létre, erre csatlakoznak a logikai feszültségeket (±5 V, ±12 V) előállító modulok. Ezáltal egyrészt a készülék külső 24 V-os egyenfeszültségű tápforrásról is működhet, másrészt a készülékbe opcióként behelyezhető akkumulátor néhány perces hálózatkimaradás esetén meg biztosítja a terminál egészének zavartalan működését.

Memória: A berendezés félvezető memóriával rendelkezik. A programok fix részét REPROM-ok tartalmazzák, melyek a REPROM bővítő egység(ek)-en levő foglalatokban helyezkednek el.
Az operatív (írható, olvasható) memória RAM modulokkal bővíthető, melyek normál vagy kisfogyasztású CMOS elemeket tartalmaznak.
Utóbbi változatban akkumulátoros egység biztosítja az információ megőrzését a tápfeszültség kimaradásakor.

Adatátvitel: A készülék terminál jellegének megfelelően kiemelkedő fontosságú az irányítás felsőbb szintjével való kapcsolattartás; ennek lehetőségét soros vonalon keresztül szinkron/asszinkron adatátviteli csatoló(k) (SACI) biztosítják, melyek helyi mikroprocesszoros vezérlőegységet tartalmaznak. Ez kettős előnnyel jár: egyrészt tehermentesíti a CPU egységet, elvégezve az adatátviteli vonal egyébként igen bonyolult kezelésével kapcsolatos műveleteket, így a CPU foglaltsága minimális, mert foglaltság csupán a DMA művelettel kapcsolatos buszhálózat formájában jelentkezik, ez pedig a berendezés real-time jellegű működésénél döntő jelentőségű.
Másrészt leegyszerűsíti a kommunikáció kezelését, hiszen az adatátviteli algoritmus megváltoztatása csupán a csatolókártyán levő REPROM cseréjét jelenti.
Standard változatként az illesztőegység a jelenleg legelterjedtebb adatátviteli protokollt BSC-t tartalmazza, a hardware interface pedig CCITT V.24. Ugyancsak soros, de galvanikusan leválasztott interface-t biztosít a távíró vonali adapter (TGLA), amely fizikai vagy bérelt telexvonalakhoz biztosít csatlakozási lehetőséget.

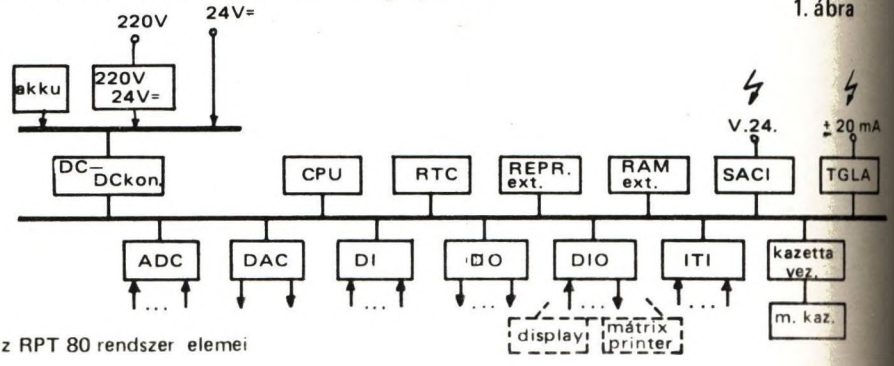
Hagyományos perifériák: A berendezéshez perifériák közül display, mátrixnyomtató, mágnes-kazetta illeszthető. A perifériaválaszték lehetővé teszi az RPT alkalmazását az irányítási rendszer magasabb (ún. diszpécserközpont) szintjén is.

Folyamatperifériák: A terminál és a folyamat között a folyamatperifériák biztosítják a kapcsolatot, az egységek a szokványos interface felületeket biztosítják.

- integráló típusú analóg bemenet (ADC), tízvonalas relés multiplexerrel
- analóg kimenet (DAC), galvanikusan leválasztott két kimenő vonallal
- digitális bemenet (DI), galvanikusan leválasztott 2x8 bites bemenő vonallal
- bistabil relés kimenet (DO), egy 8 bites vonallal
- digitális ki/bemeneti egység (DIO), 2x8+8 vonalat TTL szintű univerzális (BSI) csatlakozási lehetőséggel
- megszakítás-kezelő egység, galvanikusan leválasztott 8 multiplex megszakítási bemenettel.

Az RPT 80 hardware elemeit az 1. ábra foglalja össze.

1. ábra



Az RPT 80 rendszer elemei

Software rendszer

Az RPT 80 software rendszere magába foglalja a programok előállításához, belövéséhez, betéséhez szükséges programfejlesztő eszközöket és a folyamatterminálban futó ún. terminálprogramokat.

A programfejlesztő eszközök R 10 számítógépen használhatók az IDOS rendszer alatt, szoltatásaira építve.

A programírás lehetséges szintjei:

- PLM 80 magasszintű makronyelv
- assembler
- gépi kód.

A 8080-as mikroprocesszora írt programok belövésére nyújt lehetőséget a szimulátor, mely szintén R 10-es számítógépen futtatható.

A programok lehetnek:

- assembler fordító által generált file-ok
- lyukszalagról (mágnesszalagról) származó file-ok
- klaviatúráról bevitt bináris programok.

A szimulátor segítségével belőtt programok lyukszalagra vihetők, ill. REPROM-ba tölthetők.

A terminálprogramok lényeges része az alapsoftware jellegű szolgáltatásokat nyújtó RPS multitask rendszer, amely az RPT real-time környezetben történő alkalmazását teszi lehetővé; real-time rendszerek megfogalmazásához, programozásához nyújt eszközöket és módszereket,

- a feladat párhuzamos folyamatokban történő megfogalmazhatósága
- eseménykezelés
- erőforráskezelés
- időkezelés.

Az RPS szolgáltatásaira épül a fix és lebegőpontos műveleteket tartalmazó aritmetikai programcsomag és a standard perifériákat kezelő I/O programcsomag.

Főbb alkalmazási területek

Terminálként:

— Bányászat, kőolaj-, földgázkitermelés. A kitermelt anyag mennyiségi, minőségi adatainak folyamatos mérése, kiértékelése, regisztrálása. A mért, illetve számított adatok továbbítása a központ felé. A lényeges adatok helyi kiértékelése a vezeték mentén elhelyezett állomásokon. Adatok továbbítása a központ felé. Önállóan vagy központi parancsra vezérlőjelek kiadása szivattyúk, tolózárak, zsilipek kapcsolók stb. működtetéséhez.

— Nagykiterjedésű ipari üzemek
Hierarchikus adatgyűjtő, folyamatirányító rendszerben kisebb folyamatrészek ellenőrzése, szabályozása a fölérendelt számítógéppel együttműködve.

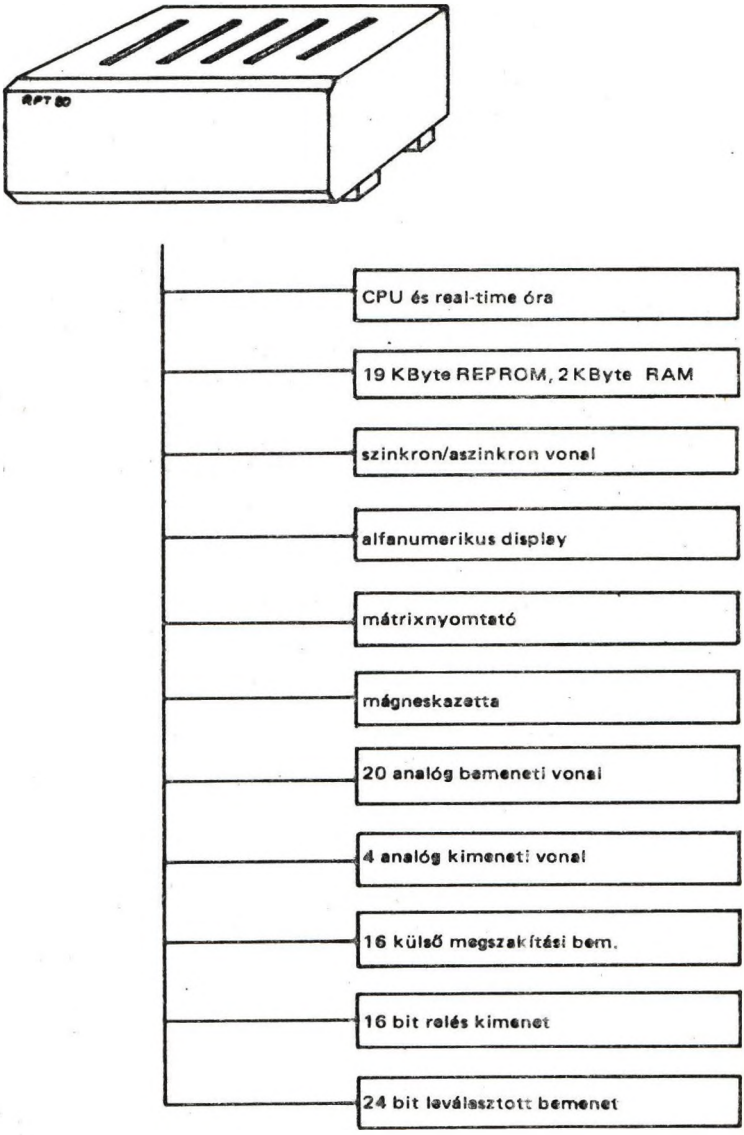
— Vízügy, környezetvédelem, meteorológia
Adott körzet adatainak (talaj-, víz-, levegő-, csapadékjellemzők) mérése és továbbítása önálló, bérelt vagy kapcsolt vonalon át a központ felé.
Önállóan:

— Laboratórium
Mérés, analízálás, vezérlés, szabályozás.

— Ipari üzem
Szerszámgép vezérlése, bonyolult termék automatikus végellenőrzése, kisebb folyamatok önálló szabályozása.

Egy lehetséges kiépítésre mutat példát a 2. ábra.

Az RPT 80 folyamatterminál a decentralizált irányítási rendszerek alsó szintjén jelentkező feladatok megoldásának eszköze, így az ilyen típusú rendszerek eszközbázisának csak egy részét képezi. Az irányítási rendszer magasabb szintjein jelentkező feladatok megoldására alkalmas eszközök is rendelkezésre állnak: az R 10 M és R11 számítógépek különböző kiépítésű változatai és az ezeket összekapcsoló adatátviteli berendezések lehetővé teszik komplex irányítási rendszerek kialakítását egy egységes koncepció szellemében.



2. ábra

Az RPT 80 egy lehetséges kiépítése

STATISZTIKAI METAADATBÁZIS

Györki Ildikó
KSH SZIG

A metaadatbázis általában

Az a körülmény, hogy az adatbázisokban tárolt adatoknak több gazdája és felhasználója van, akik egymás adatait is használhatják, napjainkban sürgetően vetette fel az igényt az adatbázis-környezet információinak dokumentálására.

A problémát a legkülönbözőbb nevek és definíciók alatt említik

- fogalmi séma (conceptual schema)
- metaadatbázis
- adatszótár (útmutató) data dictionary (directory)

A lényeg azonban az, hogy nem elég az eddig szokásos adatbázis leírás, hanem ezen kívül olyan eszközöket kell biztosítani, amelyek lehetővé teszik

- az adatbázisban tárolt információk egységes leírását, elnevezését, dokumentálását,
- más információkkal való kapcsolatuk leírását,
- az adatokhoz való hozzáférés jogosultságának dokumentálását,
- az adatok eredetére, érvényesség ellenőrzésre vonatkozó információk tárolását,
- több adatgazda esetén az adatok aktualizálására szolgáló programok, felelősök jegyzékének kezelését stb.

A metaadatbázis nemcsak az adatbázisban tárolt, illetve annak megfeleltethető információkat írja le, hanem a teljes információ rendszert. Az adatbázisban tároltakra a dokumentációs igényen túl még biztosítani kell az információk és adatok (adatstruktúra elemek: mezők, file-ok stb.) és a tárolók közti leképzést is, és egy interface-szel lehetővé kell tenni, hogy az adatbázis elemei csak a metaadatbázison keresztül legyenek elérhetők.

A statisztikai metaadatbázis

A metaadatbázisok igénye legelőször statisztikai adatbázisoknál merült fel. Ennek oka abban rejlik, hogy itt nagy információtömeget tárolnak (a megfigyelt egységek sokezer tulajdonságát), és az adatok gazdáinak és felhasználóinak köre jelentősen eltér.

Le kellett írni az adatbázisban, illetve a statisztikai információrendszerben létező információkat:

- hogy a felhasználót tájékoztassa a statisztikai táblákról, elemzésekről, kiadványokról, adatgyűjtésekről
- hogy lehetővé tegye az adatgyűjtések összevetését a redundancia elkerülése érdekében
- felfedje a különböző témák közti koordinációs problémákat (eltérő definíciók, nómenklatúrák, módszertan)
- az adatok visszanyeréséhez részletes információt adjon, illetve tároljon (az adatokról és kapcsolatairól)
- lehetővé tegye a leggyakrabban használt eljárások programozás nélküli használatát a programok tárolásával

– az adatvédelemmel, az adatok titkosságával kapcsolatos információk tárolását.

Ahhoz, pedig, hogy ne csak egy dokumentációs, elemző rendszer legyen, hanem a már működő adatbázisokból nyerhető információk ténylegesen elérhetők legyenek rajta keresztül, biztosítani kellett az információ (adat) tároló közti összefüggést,

Igy a metaadatbázisok:

– a fenti tartalmú információleíró részen kívül
– adatleíró részből (logikai és fizikai szerkezetükről)
– az információ (adat) összefüggést leíró részből
– és programrészből (amely az adatok kezelésére szolgáló programokat tárolja)

állnak.

Több fejlett statisztikai hivatalnál indult fejlesztési program a metaadatbázis vagy másutt adatszótár/útmutató néven nevezett rendszer és software-je tervezésére pl. a kanadai, svéd, belga, düsseldorfi statisztikai hivatalokban, a pozsonyi kutatóintézetben. A kezelő rendszerhez minden esetben saját software-t írnak, nem használnak meglevő általános adatbáziskezelő rendszereket.

A rendszerek felépítése, tartalma nem egységes, a metaadatbázisok értelmezésében, funkcióiban definíciós különbségek vannak. De az a tendencia, hogy a rendelkezésünkre álló információtömeget ismerjük és kezelni tudjuk, az világosan látható.

3. A magyar statisztikai adatbázis, STAR rendszer

A magyar statisztikai hivatalban 1974 óta működik adatbázis egyre szélesedő adatkörre mind a tárolt időszakok, mind a témák tekintetében.

Jelenleg iparstatisztikai (I–STAR)
külkereskedelemstatisztikai (K–Star)
munkaügyi (L–STAR)
beruházási (B–STAR)

témákban mintegy 90 millió adatot tárolnak, témától függően 3–9 éves időszakra. A nagy adattömeg annak a következménye, hogy a viszonylag kis számú 1600–8500 adatszolgáltatót nagyon sok szempont szerint figyelik meg. A tárolás időszakonként, témánként és megfigyelési egységtípusonként (megfigyelési szintenként) külön fizikai file-okban történik. A file-ok között adatstruktúrában leírt és fizikailag létező kapcsolat nincs. Ennek oka, hogy a file-ok közt nagyon bonyolult kapcsolatokat kellene kezelni, amelyeknek csak kis és változó részét használnánk. Az alkalmazó rendszer a kapcsolatokat a feldolgozás során valósítja meg, de a kapcsolat létrehozása nagyon egyszerűen történik. Ez a software a MARK IV fejlett file-kezelő rendszer. A rendszer IBM 370/155-ös gépen működik 8 db 30 milliós és 4 db 100 milliós lemezzel.

A file-ok egyszerű összekapcsolhatóságához szervezéssel teremtjük meg a feltételeket. A koordináció kulcsokat a kapcsolódó file-ok egységesen tartalmazzák.

Eddig a rendszer batch környezetben működött, mind a karbantartó, mind a lekérdező feladatokat programozók oldották meg.

A kérések és felhasználók egyre növekvő száma azonban szükségessé tette, hogy ne csak a programozók férhessenek hozzá az adatbázishoz, hanem a statisztikusok közvetlenül fogalmazhassák meg igényeiket, és kis feladat esetén azonnal (on-line környezetben) futhasson.

Amíg azonban a programozó ismerte az adatok file-okba tagolását, kapcsolási lehetőségeit és programozási ismeretei alapján könnyen megoldotta a file-ok összekapcsolását és a szükséges

tok leválasztását, ugyanez a végfelhasználótól (pl. statisztikusoktól) nem várható el. A felhasz-nak úgy kell rendelkezésére bocsátani az adatokat, hogy a végfelhasználói nyelven csak a kséges adatokat, kiválasztási kritériumokat és az adatokkal végzendő tényleges számításokat jen leírni. A csak adatfeldolgozás jellegű feladatokat át kell venni a felhasználótól, automati-ssá kell tenni.

A metaadatbázis igénye a STAR-ban

szempont: a végfelhasználói nyelv következménye

Ahhoz, hogy a végfelhasználó a fenti elvek szerint fogalmazhassa meg igényét le kell írni tárolni kell

- az adatbázisban tárolt és a felhasználó által hozzáférhető, kezelhető statisztikai információkat
- és mindazokat az adatbázis logikai, fizikai szerkezetével kapcsolatos információkat, amelyek lehetővé teszik a végfelhasználó lekérdezéséhez az adatfeldolgozó rész automatikus generálását.

szempont: az információs rendszer dokumentációs igénye

Az adatbázisban tárolt információkon kívül, adatgyűjtések kérdőíven, kiadványokban, unkatáblákon, archivált szalagokon sok millió információt őriznek a statisztikai hivatalban. ezeknek a helyét, az előkeresés módját megismerni nem kis feladat. Ezért olyan dokumentációs ndszert kell kidolgozni, amelyből témánként elő lehet keresni a rendelkezésre álló mutatók lyét, azonosítóját. És ha a kívánt mutató az adatbázisban van, akkor magát az adatot vagy a le végzett számítások eredményét is megkaphatja a felhasználó. Ez a rendszer nemcsak az in-rmáció helyéről adna felvilágosítást, hanem az információrendszer konzisztencia, redundancia mzésére is felhasználható lenne.

Ez a két szempont vezetett el a statisztikai metaadatbázis fejlesztéséhez. Mindkét rész szekapcsolt kifejlesztése hosszútávú feladat. Elsőként az adatbázis végfelhasználói lekérdezésé-z és gépi dokumentációjához szükséges részek készülnek el, és csak ezután kerül sor az adat-ázison kívüli részek bevonására.

A metaadatbázis, a végfelhasználói nyelv és az on-line lekérdezések kifejlesztése 1978-ban dult el. Jelenleg a tervezés fázisában tart.

Az első tervezési szakaszban csak lekérdezés orientált lesz a metaadatbázis, nem tartal-zza az adatbázis karbantartáshoz szükséges információkat (sem logikai ellenőrzési szempon-kat, sem programjellegű információkat).

A metaadatbázis globális felépítése

A metaadatbázis a következő logikai, adatfüggetlenségi és aktualizálási szempontból jól határolható elemekből áll.

1. adatszótás 1.1. információleíró rész (a felhasználó részére)
1.2. adatleíró rész (a software részére)
2. útmutató 2.1. adattárolást leíró rész (MARK IV definíciók)
2.2. adatállománykatalógus
3. programszótár 3.1. a felhasználó számára
3.2. a software számára

5.1. *Az információleíró rész* (adatszótár fejezetek a felhasználó számára

5.1.1. *A statisztikai mutatók tartalmi leírása:* a mutatók azonosítója mellett tartalmazza azok megnevezését, vonatkozási körét, idejét, a tárolás legmélyebb szintjét, összefüggését más mutatókkal, a rendelkezésre álló időintervallumot stb.

5.1.2. *A statisztikai nómenklatúrák és értékkészletük:* a megnevezésen és kódon kívül az egyes értékek időintervallumát, a nómenklatúra elemek összefüggését stb.
A fenti két rész eddig is a felhasználó rendelkezésére állt, de csak nyomtatott formában. Most on-line módon lekérdezhető lesz és téma szerinti visszakeresést is lehetővé tesz.

5.1.3. Leírja azokat a *tartalmi metszeteket* (mutatóhalmazt), amelyeken egy lekérdezés logikusan, statisztikailag értelmes módon végrehajtható.
Mivel ez a végfelhasználói nyelv egy alapeleme érdemes egy kicsit bővebben megnézni, mit tartalmat.
Tartalmi szempontból kijelölhetők a teljes adatbázisrendszerből olyan mutatóhalmazok, amelyek azonos szinten, azonos körre bocsáthatók rendelkezésre. Bármilyen más mutatók az adott halmazba bevonva nem biztosítható a feldolgozás helyessége. Ezeket adatbázis metszeteknek hívjuk. Egy mutató több metszetnek a része is lehet. Ilyen pl. 1. az éves munkaügyi adatok összessége gazdasági egység szinten az egész népgazdaságra 1977-78 évre
2. az iparvállalatok éves telepi adatai
1971–78. évre
3. a gazdálkodó egységek összes éves adata az ipar népgazdasági ágra gazdálkodó egység szinten 1978–78. évre (ami am munkaügyi adatokat és a telepi szinten megfigyelt adatokat is tartalmazza gazdálkodó egység szinten).
(Jegyezzük meg, hogy a metszet fogalom nem tárolási hanem információs kapcsolatot jelent.)

A metszetek leírása tartalmazza: a metszet nevét; kódját; vonatkozási kört; időt; adatszintet; homogenizáltságot; témák (mutatócsoportok) listáját, amelyekből a metszet felépül; azokat a nómenklatúrákat, amelyek szerint szelektálhatjuk, aggregálhatjuk, csoportosíthatjuk a mutatókat; stb.

5.2. *Az adatleíró rész* (adatszótár fejezet a software számára)
Az adatleíró rész feladata az információleíró részben ismertetett elemek és kapcsolatuk meghatározása adatlogikai szempontból.
Az adatleíró rész szempontjából pl. az adatbázis metszet nem más, mint Codasyl terminológiával élve egy felhasználói szempontú alséma. A file-ok és kapcsolataik teljes halmazából egy részhalmazt automatikusan kell generálni és létrehozni. Hangsúlyozni kell, hogy a statisztikai adatbázisokban

– nemcsak hozzárendelő kapcsolatok vannak, hanem
– szelektáló és
– aggregáló kapcsolatok is.

(Ezeket a rendelkezésre álló adatbáziskezelő rendszerek egyike sem oldja meg.)

Az adatleíró rész az adatbázismetszeteket, elemeit és a létrehozásához szükséges kapcsolatokat írja le.

5.2.1. *Az adatbázismetszet leírása*

magát az adatbázismetszetet jellemző adatokat tartalmaz, azonosítót, típust, biztonsági kódot stb.

5.2.2. *Az időszakok leírása*

Az időszak az adatbázismetszeten belül egy logikai és tartalmi részmetszetet jelent. Az elemek és kapcsolatuk időszakonként eltérőek lehetnek, és lekérdezési szempontból is önálló egységet képviselnek. Az időszak rész a vonatkozási idő és homogenizáltság meghatározását foglalja magában.

5.2.3. *A kulcsmezők leírása*

Egy on-line adatbázis egy időszakán belül lehetséges kulcsmezőket definiáljuk itt: a kulcselem azonosítóját; a file-definíció nevét, amiben benne van; a kulcs típusát; nem létező elemét a helyettesítési értéket; biztonsági kódot; számlálóelemzőt a lekérdezések számának gyűjtéséhez; a kulcsok dekódolásához szükséges információt stb.

5.2.4. *Az adatmezők leírása*

Tartalmazza: a file-t lefedő file-definíció nevét;
a DSN-t és az adatállománykatalógus nevét;
a file szerepét az adatbázismetszeten belül;
a szülőfile-jának az azonosítóját;
a kapcsolómező nevét az adott és a szülőfile-ban;
az olvasás típusát;
a file-kapcsolat létrehozásához esetleg szükséges szelektáló műveleteket;
a file kapcsolat létrehozásához esetleg szükséges rendező és aggregáló műveleteket;

5.3. *Az útmutató*

Az útmutató a file-ok fizikai szerekezetére és az adatállományok helyére vonatkozóan tartalmat információt. Ezek a rendszer korábban is létező elemei.

5.3.1. Az adatbázis adatleíró nyelve az ún. *file- és tábla-definíció* (MARK IV-ben), amely megadja a file-szervezés típusát; a rekord típusát; hosszát; blokkolását; a file rendezési sorrendjét; a rekord elsődleges kulcsát a mező típusát, helyét, nevét stb.

Ezek segítségével meghatározható az adatok fizikai elhelyezkedése a *tárolás módja.*

5.3.2. *Az adatállománykatalógusok* a file nevét és a tárolóeszköz azonosítóját tartalmazzák.

5.4. *Programszótár*

– egyrészt a felhasználó számára segítőinformációt tartalmaz a végfelhasználói nyelv használatához
– másrészt futásvezérlő információkat a software számára.

5.4.1. *Programszótár a felhasználó számára*

A végfelhasználó lekérdezéséhez ad segítséget.

– a felhasználó gyakorlottságának megfelelően különböző részletességű leírást a rendszer lehetőségeiről, az utasítások használatáról,
– hibaüzenetekhez kapcsolódóan automatikusan behívja a teendők leírását.

5.4.2. *Futásvezérlő információk a software számára*

A futásvezérlő információk általános adatvédelmi eljárásokat, file-okat és az on-line rendszerben aktuálisan érvényben levő korlátozó a paramétereket tartalmazzák.
(Pl. az egyszerre kezelhető lemezek száma, a display-en leírható maximális táblaméret, maximális futásidő, maximális munkaterület igény stb.)
Ezeket az információkat felhasználja a lekérdező rendszer és ha adott program paraméterei meghaladják a programszótárban tárolt értékeket, akkor hibajelzést ad és tájékoztatja a felhasználót a futás sorsáról (pl. batchben hajtódik végre, a tábla sornyomtatóra megy stb.)

6. A metaadatbázis kezelő rendszer

A metadatbázis, ha nem tartalmi szempontból nézzük, akkor egy általános adatbázisnak tekinthető meg.

Adatstruktúráját tekintve egy bonyolult hálós kapcsolatokkal rendelkező rendszer

Leírása adatdefiníciós nyelvet igényel, a struktúrája, tartalma ugyanúgy változhat mint bármely más adatbázisé, ezért az adatfüggetlenség érdekében el kell különíteni a definíciókat a programoktól

Kezelése: három fő funkciója van a metaadatbázis kezelő rendszernek:

a) a metaadatbázis kreálása, módosítása és közben összetett ellenőrzési eljárások a metaadatbázis konzisztenciájának biztosítására
b) a felhasználói lekérdezések támogatása, ahhoz információ-biztosítás és a megfogalmazott igények ellenőrzése
c) dokumentációkészítés az adatbázis (később az információrendszer) tartalmáról és struktúrájáról

A kezelő rendszer software-jét a tervek szerint saját programozással készítjük el.

A KONCEPCIONÁLIS ADATMODELL EGY LEHETSÉGES MEGFOGALMAZÁSA

Dr. Halassy Béla
SZÁMOK

1. Bevezetés

Az utóbbi időben egyre nagyobb teret kap a szakirodalomban a valóságot lehető legjobban ...öző ún. koncepcionális adatmodell vizsgálata. Külön lendítőerőt adott a kutatásoknak az ANSI SPARC illetve az újabb CODASYL javaslat háromsémás megoldása. A kétségtelenül jelentős ...előrelépések ellenére az eredményekkel három szempontból sem lehetünk teljesen elégedettek.

– A szabványosítási törekvések fontosságának hangsúlyozása ellenére sem mehetünk el ...ert szemmel ama tény mellett, hogy CODASYL, relációs és egyéb modellezési alapokon működő ...endszerek tucatjai már működnek. A kutatók nem szenteltek kellő figyelmet annak vizsgálatára, ...gy a meglevő rendszer-koncepciók beillesztését egy szabványos koncepcióba, illetve ahhoz ...ző kapcsolódását, mely módon célszerű megvalósítani. Az osztott adatbázisok témája és a kü...öző típusú alkalmazások integrálása felveti ezen létező rendszerek együttes működésének ...ndjait is.

– Az irodalom viszonylag keveset foglalkozik azzal a gonddal, hogy a koncepcionális adat...odell voltaképpen a valóság valmilyen tényeit és összefüggéseit, milyen szinten és összefüggé...t, milyen szinten és milyen módon modellezze.

– Végül felmerül a kérdés, hogy a koncepcionális modell lehet-e ténylegesen koncepcioná..., vagy pedig valamilyen mértékű kompromisszumot a külső és belső sémáktól függetlenül is ...talmaznia kell. Ilyen kompromisszumokat tartalmazott a hagyományos CODASYL-séma.

2. A TEK* modell alapjai

Az alábbiakban bemutatásra kerülő TEK modellt a fenti kérdésekre adott meghatározott ...iszok alapozták meg. A modell sarkalatos pontjait az alábbiak szerint lehet összefoglalni:

– Nem szabad eltekinteni attól, hogy létező rendszerek együttműködésére még hosszú ...ig szükség lesz. Olyan koncepcionális modellt kell tehát kifejleszteni, amely támogatja a meg...ő rendszerek filozófiáját is.

– Megtervezhető és megalkotható egy rendszer lehető legjobb, koncepcionális adatmo...ja. Azonban nem célszerű, ha maga az adatbázis-terv logikai szinten ezzel teljesen megegye...k és csak a külső sémák tartalmaznak kompromisszumokat.

A koncepcionális modell különben is a rendszer teljes adatrendszerét leírja, függetlenül ...ttól, hogy annak bizonyos elemeit nem is kezeljük számítógépen. Ezért a koncepcionális séma ...a belső séma közé célszerű beiktatni egy olyan adatmodell szintet, amely a kompromisszumo...t tartalmazza. Ezt a szintet logikai adatmodellnek nevezzük.

A TEK modell, mint neve is mutatja, tulajdonság, egyed és kapcsolat fogalmakra épül. A ...i modellezési koncepciók egy része nem ismeri el ezen fogalmak valamelyikét (ld. bináris és ...ációs modell). Azáltal, hogy a koncepcionális szinten mindhárom fogalmat támogatjuk, to...ább a koncepcionális és logikai szintet szétválasztjuk, lehetővé válik egy probléma koncepcio...ális szintű megfogalmazása úgy, hogy azt logikai szinten már hálós, bináris vagy relációs kon...ciókra képezzük le kompromisszumok alkalmazásával.

*TEK = Tulajdonság, Egyed, Kapcsolat

3. A TEK modell alapfogalmai

A TEK modell a nevében található alapfogalmakkal dolgozik két szinten: típus és előfordulás szinten. (A típus és előfordulás fogalmakat ismertnek feltételezzük.) Az egyed fogalmát hagyományosan értelmezzük, vagyis egyednek nevezünk minden olyan dolgot (eseményt, tárgyat, személyt, tervet stb.), amit tulajdonságokkal kívánunk leírni. Az egyed és a tulajdonság fogalma egymással szorosan összefügg. Az egyedek egyedtípusokat alkotnak. Két konkrét egyed azért tartozik egy típusba, mert azonos tulajdonságtípusokkal írjuk le azokat. Azért képviselnek külön előfordulásokat, mert egy vagy több tulajdonságtípusokra nézve eltérő tulajdonság értékkel rendelkeznek.

A TEK modellben egyébként az egyedtípus fogalmát ugyanúgy határozzuk meg, mint ahogyan a relációs modellben a reláció fogalmát. Az eddigieket tekintve a TEK modell könnyen megfeleltethető a hálós és relációs modellnek, de már jelentős eltérést mutat a bináris modellel szemben. Azonban egy-egy fontos sajátosságban különbözik a hálós és relációs modelltől is. Ez a különbség a kapcsolat megfogalmazásában rejlik.

A relációs modell nem ismeri el a kapcsolat (relationship) fogalmát. A hálós modellben a kapcsolat szintaktikai jellegű, mert nem érdemi tulajdonságtípushoz, nem szemantikus tartalomhoz, hanem mutatókhoz kötődik.

A TEK modellben a kapcsolattípus és – előfordulás fogalmát tulajdonságtípushoz és – értékhez kötjük és így fokozzuk az adatmodell elemei közötti összefüggések szorosságát.

A relációs koncepciónak megfelelően, de a hálós modellel szemben a TEK modellben koncepcionális szinten minden egyedtípus kell, hogy rendelkezzen egyértelmű azonosító tulajdonságtípussal és minden egyed-előfordulás rendelkezik egyértelmű azonosító tulajdonságértékkel. Könnyű belátni, hogy hozzáadás és módosítás esetén az azonosító használata elkerülhetetlen. Implementációs kérdésnek tekintjük és ezért a logikai modell kompromisszumai közé soroljuk, hogy az azonosító tárolásra kerül-e, avagy sem. Ennek előrebocsátása után a kapcsolat fogalma már bevezethető. A kapcsolattípus fogalmát néhány kiegészítő fogalom segítségével mutatjuk be.

Két egyedtípust logikailag redundánsnak nevezünk akkor, ha létezik olyan közös tulajdonságtípusuk, amely a két egyedtípus előfordulásaiban azonos értéket is felvehet vagy felvesz. A logikai redundanciának két típusát különböztetjük meg:

– Erős logikai redundanciáról beszélünk akkor, ha redundáns tulajdonságtípus egyik egyedtípusban sem egyezik meg az azonosítóval illetve annak nem része. Ilyen esetekben belátható, hogy vagy az egyik egyedtípusban felesleges a tulajdonságtípus, vagy csak rossz elnevezéssel állunk szemben és voltaképpen nem ugyanarról a tulajdonságtípusról van szó.

– Gyenge logikai redundanciáról beszélünk akkor, ha a két egyedtípusban redundáns tulajdonságtípus legalább az egyik egyedtípus azonosítója vagy annak része. Ilyen esetekben belátható, hogy ugyanaz a tulajdonságtípus természetszerűen jellemzi mindkét egyedtípust és voltaképpen az egyiknek a másikhoz való tartozását fejezi ki. (Ennek kifejtését a következő pontban adjuk meg.)

Végeredményben a TEK modellben két egyedtípus kapcsolattípust létesít akkor, ha a két egyedtípus gyengén redundáns. A redundáns tulajdonságtípust kapcsoló tulajdonságtípusnak nevezzük. Megkülönböztetünk közvetlen és közvetett kapcsolatot.

– Közvetlen a kapcsolat akkor, ha a redundáns tulajdonságtípus legalább az egyik egyedtípus azonosítója. (A másik egyedtípusban lehet az azonosító része, vagy leíró tulajdonságtípus.)

– Közvetett a kapcsolat akkor, ha a redundáns tulajdonságtípus egyik egyedtípusban sem egyezik meg az azonosítóval.

A TEK modell és a funkcionális függés

Az alapfogalmak meghatározásából látszik, hogy az egyedtípusok és kapcsolattípusok egy- tulajdonságtípusok fogalmához kötődnek. Az egyedtípusok tulajdonságtípus sora az típus belső szerkezetét határozza meg. A TEK modellben feltételezzük, hogy az egyedtípus szerkezetét akkor határozzuk meg helyesen a koncepcionális adatmodellben, ha ez a szerke- harmadik normál formájú.

Mint ismeretes, egy egyedtípus belső szerkezete akkor harmadik normál formájú, ha:
- minden tulajdonságtípusa funkcionálisan függ az azonosítótól,
- a teljes azonosítótól,
- és csakis az azonosítótól.

Azáltal, hogy az egyedtípus olyan tulajdonságtípust is tartalmaz, amely más egyedtípust ismez (gyenge redudancia) az egyedtípus meghatározott kapcsolattípusokkal rendelkezik. Ezek a közvetlen kapcsolattípusok adják az egyedtípusok külső szerkezetét, vagyis az egyed- sok közötti szerkezetet.

Könnyen belátható, hogy egy egyedtípus külső szerkezete mögött is funkcionális függőségi vany áll. A redundáns tulajdonságtípus ugyanis az egyik egyedtípusnak azonosítója, a másik- vagy leíró tulajdonsága, vagy azonosítójának része. Mindkét esetben a második egyedtípus- a kapcsoló tulajdonságtípus funkcionálisan függ az azonosítótól, ha az egyedtípus harma- normál formájú. Ebből viszont következik, hogy az első egyedtípus azonosítója is funkcio- san függ a második azonosítójától. Tehát az egyedtípusok közötti közvetlen kapcsolat volta- nem külső, azaz két egyedtípus közötti funkcionális függést takar.

A TEK modell és a kapcsolattípusok foka

A külső funkcionális függés koncepciójából következik, hogy a TEK modell csak bizonyos kapcsolatokat enged meg. Mint ismeretes, a kapcsolattípusokat fokukkal is szokták jelle- ni. A kapcsolattípus foka megadja, hogy az egyik egyedtípus hány előfordulása a másik egyed- hány előfordulásával létesít konkrét kapcsolatot. A kapcsolattípus foka ennélfogva lehet 1:N, M:1, vagy M:N.

Mivel a TEK modellben két egyedtípus között létező kapcsolattípus funkcionális függést ez, ebből következik, hogy a kapcsolattípus foka általában M:1. Kivételes esetben, vagyis kölcsönös funkcionális függés esetén pedig 1:1. Az M:N fokú kapcsolattípust a TEK modell támogatja. Ez ugyanis feltételezné, hogy a kapcsoló tulajdonságtípus valamelyik kapcsolt típusban ismétlődik. Az ismétlődést viszont a harmadik normál forma nem engedi meg.

Az előzőekből következik, hogy M:N fokú kapcsolattípust a TEK modellben csak közve- kapcsolatként lehet megvalósítani, vagyis a két kapcsolt egyedtípus rendelkezik egy közös rendelt vagy alárendelt egyedtípussal.

A jelenlegi modellezési koncepciók nem tartalmaznak olyan megkötést, amelynek értel- ben a koncepcionális adatmodell csak M:1 vagy 1:1 fokú kapcsolatokat tartalmazhat. Azon- a tényleges adatbázis kialakításakor ezt a kompromisszumot minden alkalommal végre kell ni és nem létezik olyan kezelő rendszer, amely az M:N kapcsolattípust kezelni tudná.

A kapcsolattípusokkal összefüggésben meg kell jegyezni, hogy a TEK modellben explicit nem lehet utalni a kapcsolattípusok két sajátosságára. A bináris modellhez hasonlóan meg kell adni a kapcsolat opcionális vagy kötelező jellegét. Ezen túlmenően két vagy több kapcso- lattípus viszonyát is meg lehet határozni. Két kapcsolattípus kizáró viszonyt létesít, ha az egyik előfordulásának jelenléte kizárja a másik előfordulásának megvalósíthatóságát és fordítva. Két kapcsolattípus bennfoglaló viszonyt alkot, ha az egyik kapcsolattípus előfordulásának létreho- tételezi, hogy a másik kapcsolattípusnak már létezik meghatározott előfordulása.

6. Kapcsolattípusok és a modell szerkezeti egységei

A kapcsolattípusok segítségével egy koncepcionális modellben előforduló tipikus szerkezeti egységek építhetők fel. Ezek az egységek a hierarchia, háló, tranzitív szerkezet és az önmagába visszamutató kapcsolat.

Azok az egyedtípusok, amelyek egymással oly módon létesítenek kapcsolatokat, hogy azonosítójukon keresztül az alárendelt egyedtípushoz, egy másik tulajdonságtípuson keresztül viszont a fölérendelt egyedtípushoz kapcsolódnak, hierarchiát alkotnak.

Amennyiben egy egyedtípus kettő vagy több kapcsoló tulajdonságtípussal rendelkezik, amelyekkel kettő vagy több fölérendelt egyedtípushoz kapcsolódik, úgy hálós szerkezetről beszélünk.

Tranzitív a szerkezet akkor, ha egy egyedtípus alárendeltje tartalmazza mind az adott egyedtípus, mind annak fölérendeltjének kapcsoló tulajdonságtípusát.

Az önmagába visszamutató (1:1 vagy M:1) fokú kapcsolatot is meghatározhatjuk oly módon, hogy ilyen kapcsolattal rendelkezik az az egyedtípus, amelyben az azonosító tulajdonságtípus a leíró tulajdonságtípusok között külön is szerepel (természetesen minősített névvel).

A TEK modell nem engedi meg kettő vagy három egyedtípus között ciklus felépítését. Mivel az N:M fokú kapcsolattípusokat nem ismeri el, az önmagába visszamutató kapcsolat sem lehet N:M fokú.

Viszont az ismertetett fogalmak segítségével jól meghatározható másodlagos kulcs egyedtípus, kapcsoló egyedtípus, struktúra egyedtípus – ezek a hagyományos fogalmak megfelelői – továbbá a hagyományos al-file fogalomnak párja, az egyed-altípus. Ez utóbbival kapcsolatosan megjegyezzük, hogy külön gondot jelent az absztrakciós folyamatban a megfelelő absztrakciós szint kiválasztása. Voltaképpen arról van szó, hogy az egyedtípusokat mesterségesen határozzuk meg és emiatt inhomogén egyed alhalmazokból absztraháljuk az egyedtípust. Ezen alhalmazok kezelésére különböző megoldások léteznek (pl. többszörös kapcsolat).

Végül megjegyezzük, hogy az egyes adatmodell-elemek sajátosságainak igen részletes feltárása a megfelelő adatszótár felállítását célozza.

7. A TEK modell hatása az adatbáziskezelő rendszerre

A TEK modell kellő általánossága következtében jelölt lehet arra, hogy különböző adatkezelő rendszerek közötti kommunikációt elősegítse. Alkalmasnak látszik a lényegében hasonló szerepet betöltő adattranszlátorok központi logikai sémájául. Megfelelő illesztésekkel akár a CODASYL, akár a relációs modellre leképezhető egyértelmű szabályok segítségével. Bizonyos mértékig igaz ez a bináris modellel való kapcsolatára is.

A kapcsolattípusoknak tulajdonságtípusként történő definiálása és ebben az értelemben való rögzítése a koncepcionális sémában nemcsak a modell felépítését, helyességének állandó ellenőrzését könnyíti meg. Alkalmat ad egyrészt a kapcsolatok tényleges tartalmi kezelésére, amely magasabb nyelvi szintet tesz lehetővé. Másrészt a hálós kapcsolatokban az egyik fölérendeltről a másikra történő navigálást automatikussá teszi külön adatútmutató használata nélkül. A relációs rendszerhez hasonlóan lehetővé teszi a matematikai értelemben vett halmazműveleteket.

A kapcsolatok viszonyának illetve kötelező vagy opcinális jellegének meghatározása bővíti az automatikus ellenőrzési lehetőségeket. Bővül a rendszer lehetősége alhalmazok kiválasztására az egyed-altípusok különböző meghatározási lehetőségein keresztül.

Ugyanakkor be kell vallanunk, hogy ezeket a lehetőségeket részletesen nem elemeztük. A TEK modell elsődleges célja ugyanis az adatmodellezési folyamat megkönnyítése és ezen a területen hasznosnak is bizonyult.

OPERATÍV TÁVVEZÉRLŐ FUNKCIONÁLIS ÉS STRUKTÚRÁLIS JELLEMZŐINEK MEGHATÁROZÁSA MULTIPROCESSZOROS SZÁMÍTÓGÉPBEN

Harmat László—dr. Reszler Ákos
SZKI

1. Bevezetés

A 70-es évektől kezdődően a kis- és középkategóriájú számítógépek közt markánsan jelentkezik és a mikroprocesszorok központi egységekben való alkalmazásával tovább erősödik az a tendencia, hogy a hagyományos központi egység feladatokat (utasításfeldolgozás, I/O vezéslés, rendszerfelügyelet) a központi egységet alkotó néhány processzor látja el. Itt processzoron önálló feldolgozóképességű, az operatív táron, mint közös erőforráson osztozó, de saját memóriával rendelkező, a számítógép egyéb részeivel közvetett, vagy közvetlen információs kapcsolatban álló egységet értünk.

A fentiekre és egyben az egyes gépekkel kapcsolatos eltérő szóhasználatra néhány példát hozunk.

Az IBM System/370 Model 125-ben a központi egység funkciókat az utasításfeldolgozó processzor (IPU), a csatornákat megvalósító illetve az integrált periféria csatoló input-output processzorok (IOP) és az operátor-gép kapcsolatot és diagnosztizálátst biztosító szervizprocesszor SVP) látja el [1].

A PDP–11/70-ben a központi processzoron (central processor) túlmenően az I/O vezérlők high-speed I/O controller) és a lebegőpontos processzor (floating point processor) esik a processor kategóriába[2].

A Mitra 15/30 max. 4 feldolgozó egységet — vagyis processzort — tartalmaz (unité de traitement), melyek közül egy a központi egység funkciókat realizálja (unité centrale), a többi pedig az igényesebb I/O funkciókat (unité d'échange) [3].

A processzorok (P_1) specifikus feladatuknak és az adott gép architektúrájának megfelelően kommunikálnak az operatív tárral (OT). E kommunikáció minden egyes processor esetében definiálható a (kezdő) címmel, az adathosszal, az átviteli sebességgel, stb. Egyben a kommunikáció feltételezi a cím-, adathossz-, adat- stb. kezelés eszközeinek létét.

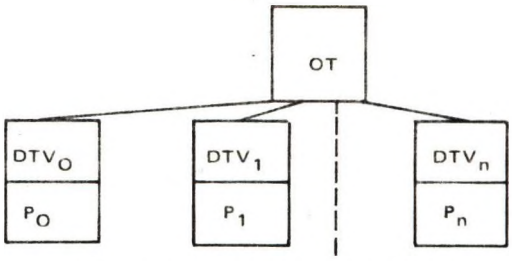
A számítógép cél-kategóriájának megfelelően két szélső megoldás kínálkozik a fejlesztőnek ezen eszközök megvalósítására, így

- az 1. ábra szerinti, maximális sebességet célzó megoldás: processzorként dedikált operatív tárvezérlés (DTV),
- a 2. ábra szerinti, minimális eszköz igényű megoldás: közös operatív tár vezérlő (OTV)

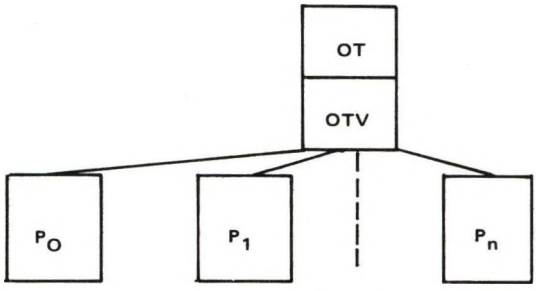
Előadásunk célja, hogy bemutassa egy a 2. változathoz közelálló — kiskategóriájú — multiprocesszoros felépítésű számítógép operatív tár vezérlője tervezésének néhány momentumát.

2. Első megfontolások

A funkciók és struktúra meghatározásánál „a minimális eszközök mellett minél nagyobb teljesítmény" cél eléréséhez a struktúraelemek többszörös kihasználására kell törekedni. Ezzel az elvvel összhangban van a processzorok mikroprogramozott felépítése.



„Max. sebességű" változat struktúrája
1. ábra



„Eszköz takarékos" változat struktúrája
2. ábra

Az OTV alapvető feladata, hogy biztosítsa a P_1 – OT adatforgalmat, a processzorok egyidejű hozzáférési igénye esetén a kiszolgálásukat valamilyen algoritmus szerint sorrendezze. Ezen túlmenően az OTV-hez kell rendelni mindazokat a feladatokat, melyek

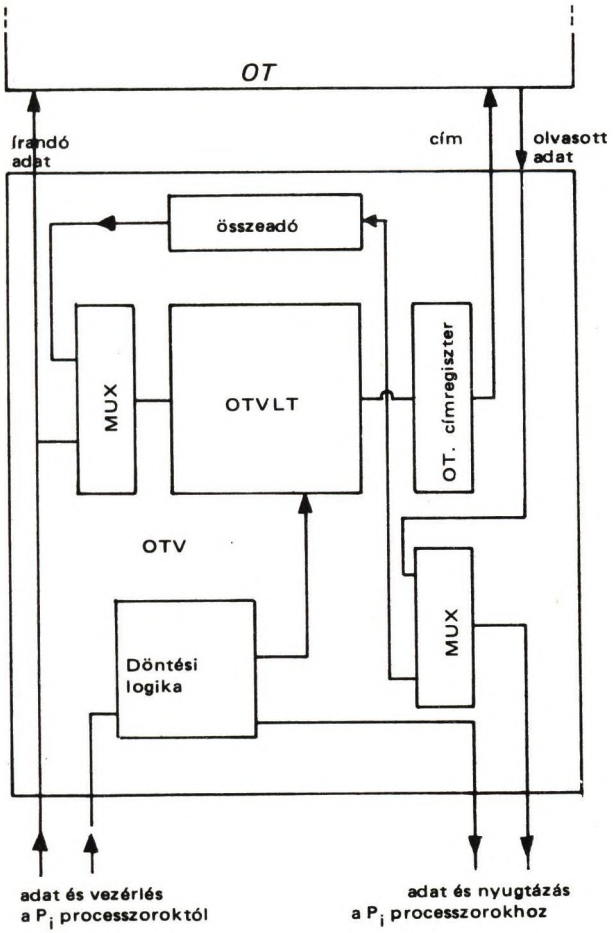
- a P_1 – OT és a P_1 – P_j kapcsolatok során azonos módon, tipizálhatóan hajtódnak végre, és melyeknek
- egy egységben (az OTV-ben) való koncentrálása nem jár időveszteséggel, vagy melyek
- realizálásához a struktúraelemek – más funkciók realizálásából következően – az OTV-ben már rendelkezésre állnak.

A feladatok léte sok tekintetben a fejlesztett gép architektúrájának függvénye (az utasítás- és adathossz fix/változó volta, az I/O műveletek, a megszakításrendszer).

A következőkben gondolatmenetünket egy olyan példán mutatjuk be, amelynél az architektúra meglehetősen gazdag.

Alapvonásai:

– információs egység:	1 byte
– utasítás rendszer :	kétcímű
– utasítás hossz:	változó (2,4,6)
– adathossz:	változó (1–256 byte)
– I/O adathossz:	változó (1–64 Kbyte)
– OT címzési tartomány:	max. 16 Mbyte



Az OTV vázlatos struktúrája
3. ábra

- tárvédelem: van
- I/O műveletek : vezérlő- és állapotinformáció átadása az operatív tárban levő „mail-box"-okon keresztül.

TV struktúra felvázolása

A feltételezett architekturális jellemzőkből következik, hogy a tárhoz forduláskor a pro-oroknak kezelniük kell a címet és az adathosszt. E feladat megfelelő hosszúságú regiszte-l és összeadóval általánosan megoldható. Az előző pontban tárgyalt elv értelmében ezeket közöket összevonjuk egy OTV lokális tár (OTVLT) és összeadó formájában. Ez a megol-megengedi, hogy a P_i – OTV kapcsolat egy kétirányú adatsínnel realizálható legyen. (Azt a st nyitva hagyjuk, hogy e kapcsolat elrendezése soros vagy sugaras.) A konstrukciósan nem us OT – OTV kapcsolat címsínt, írási adatsínt és olvasási adatsínt tartalmaz.

A tárvédelmi mechanizmus triviálisan a címzéshez társul. Prioritási vagy más logika dönt a esszorok egyidejű hozzáférési igényének kiszolgálási rendjéről. Ezek alapján a 3. ábrán fel-juk az OTV struktúráját.

4. A kapcsolatok mennyiségi elemzése

Az OTV vázlatos struktúrát szembesítjük a számítógép kategória követelményeivel és azon belül a tipikus utasításfeldolgozó, periféria-vezérlő feladatokhoz rendelt processzorok által támasztott, elsősorban sebességi követelményekkel.

Részben az architektúrából következően, részben azért, hogy a variációs lehetőségeket illusztráljuk, egyaránt feltételezünk egy- és kétbyte-os OT–OTV–P_i adatút szélességre.

P_i funkciója	Adatátviteli sebesség (Kbyte/s)	Feldolgozási egység (byte)	Átviteli kapcs. frekvenciája ($10^3 s^{-1}$)	Megjegyzés
diszk csatolás	806	2	403	
mágnesszalag csatolás	32	2	16	
multiplex csatorna	14÷112	1, 2	14÷56	adatátvitel + adatláncolás
adatátviteli vonalak csatolása	6÷48	1, 2	6÷24	adatláncolás
konzol csatolás	40	1	40	
Összesen:	1058		599	
Ebből pufferelt	49		45	
pufferelletlen	1009		514	

OT–OTV – P_i kapcsolatok maximális sebesség

1. táblázat

Az 1. és 2. táblázat értékeinek összevetéséből látható, hogy az OTV az PTV–P_i kapcsolat szempontjából az OT vonatkozású kapcsolatok a dominálóak, az OTVLT vonatkozásúak másodlagosak.

Az adatokból az OTV és az OTV–P_i kapcsolat maximális átlagos összadatátviteli sebességére is kaphatunk értéket. A követelményeket képviselje egy olyan perifériás rendszerteszt futtatása, mely az összes nem pufferelt perifériát az előírt maximális programozási esetnek megfelelően – tipikusan 80 byte hosszú I/O adatokat mozgassanak. Ez esetben az I/O műveletek előkészítésével és befejezésével kapcsolatos, az utasításfeldolgozó processzor vonatkozású OT és OTVLT forgalom elhanyagolható mértékű, illetve a forgalom lassulása nincs semmiféle negatív hatással a rendszerteszt futására. Az OTV minimális összadatátviteli sebességére

$$V_1 = 1161 \text{ kbyte/s} \approx 1{,}2 \text{ Mbyte/s},$$

a minimális átviteli kapcsolat frekvenciára (byte-os átvitelt három csatolás viszonylatában feltételezve)

$f_1 = 590 \cdot 10^3\ s^{-1} \approx 600 \cdot 10^3\ s^{-1}$

ag értéket kapunk.

Az átviteli kapcsolat frekvenciára természetesen más és más értékeket kapunk, ha a pro-szorokban más feldolgozási egységet tételezünk fel, és/vagy az OT–OTV–P_i adatút szélessé-t máshogy választjuk meg.

s OTVLT-vel mmunikáló rocesszor funk-ója	Adatátviteli sebesség (Kbyte/s)		Feldol-gozási egys. (byte)	Átviteli kapcs. frekvenciája ($10^3 s^{-1}$)		Megj.
	puffe-relt	puffere-letlen		puffe-relt	puffe-releten	
ck csatolás	–	X	2	–	X	
ágnesszalag atolás	–	X	2	–	X	
ultiplex csatorna	8	104	2	4	52	
átviteli alak csatolása	–	48	2	–	24	
zol csatolás	–	X	2	–	X	
zesen:	8	152		4	76	

elhanyagolható

Az OT kapcsolatokat kísérő OTVLT kapcsolatok maximális sebessége

2. táblázat

Az így kapott adatok alapján előzetesen kiválasztható a leginkább eszköztakarékos, de a ességi követelményeket már kielégítő megoldás, meghatározható a minimális megengedett -OTV–P_i adatútszélesség.

Hangsúlyozzuk, a kapott adatok átlag értékek, melyekkel szemben a valóságos viszo-kra a csúcsértékeket a tényleges mikroprogramok, várakozási idők, pufferméretek, priori-k ismeretében végzett részletes időelemzéssel kell kiszámítani. (Az ismert módszereket itt n tárgyaljuk.)

A kapcsolatok minőségi elemzése

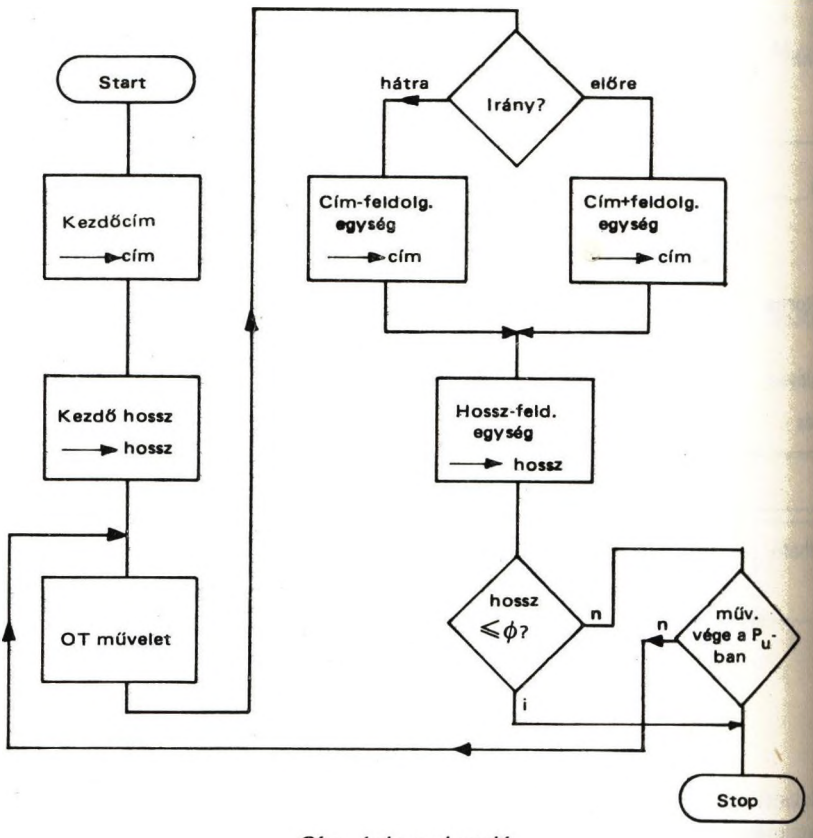
Az előző pontok alapján kapott eredmények további feldolgozásához jellemezzük az OTV csolatokat az átvitelre kerülő információval.

Itt példaképpen az utasításfeldolgozó processzor (P_u) és az OT–OTV közötti adatátvitelt gáljuk.

Az OT–OTV és P_i között átvitelre kerülő információ:

- cím és kulcs (4 byte)
- 1. operandus (1–256 byte)
- 2. operandus (0–256 byte)
- utasítás (2,4,6 byte)

Az utasítás és az operandusok feldolgozása a P_u-ban utasításfüggő, azonban a címüket, hosszukat egységesen a 4. ábra szerinti módon kezeljük.



Cím- és hosszkezelés

4. ábra

A 4. ábra szerinti cím és hossz aktualizálása mindig azonosan kerül végrehajtásra és mindig csak annak az egy P_i-nek a vonatkozásában, mely éppen az OT-val – egyben az OTV-vel – kapcsolatban áll. Az aktualizálásnak a processzorokban való végrehajtása (kb. 4 mikroutasítás) egy állandóan ható, az egész modell működését lassító tényező.

Kézenfekvő a következő megoldás:

Az OTV lokális tárában rekeszeket biztosítunk a címeken túlmenően a hosszok tárolására is. Az OTV-ben helyezzük el az összeadó-művet, mely elvégzi mind a cím, mind a hossz aktualizálását. (A tipikus OT időviszonyok lehetővé teszik, hogy az OT művelettel egyidőben akár az OTVLT-ben tárolt cím, akár a hossz, akár sorosan mindkettő aktualizálásra kerüljön.)

Ellenőrzés átlagos adathosszra

A processzor és az OTV terhelését három alapvető esetre vizsgáljuk meg:

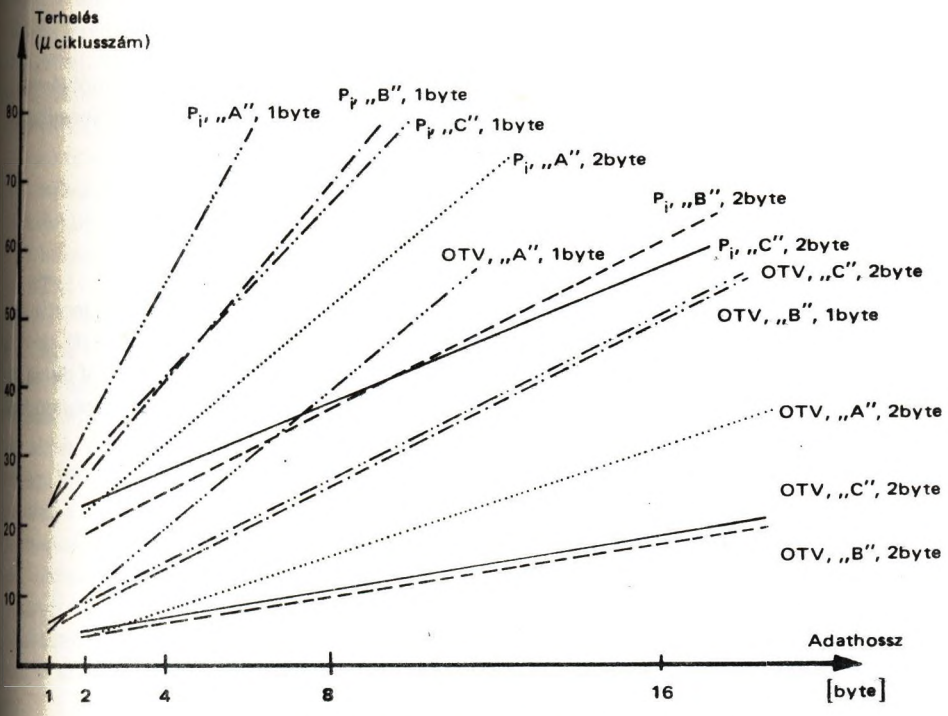
A – a processzor minden feldolgozási egység OT olvasása és írása esetén címet küld, a cím és hossz aktualizálása a processzorban történik;

B – a cím aktualizálását az OTV végzi az OT művelet alatt., a hossz aktualizálását pedig a processzor az OT–OTV művelettel szorosan;

C – a cím és hossz aktualizálását az OTV végzi az OT művelet alatt.

Az egyes P_i processzorok mikroarchitektúrája, belső adatútjainak szélessége (feldolgozási egység) és az átlagos adathossz függvényében minősül az A, B vagy C eset előnyösnek.

E függvénykapcsolat grafikus ábrázolását az 5. ábrán mutatjuk be egy példaképpeni utasításfeldolgozó processzor mikroarchitektúra, 1 és 2 byte-os feldolgozási egységre.



Az OTV és P_i terhelése az adathossz függvényében
5. ábra

Az ábrából következően az OTV terhelése szempontjából a nagyobb adatútszélesség (itt 2byte) és a B vagy C változat az előnyös. A P_u terhelése szempontjából az adathossz függvényében a B vagy C változat az előnyösebb.

Az átlagos adathosszra adatot a központi egységek teljesítményének meghatározásához használt – tipikus programokat reprezentáló – mixeknek az adott architektúrára vonatkozó megvalósításából nyerhetünk. Ez azt eredményezheti, hogy az 5. pont végén tett kijelentéstől eltérünk: a hossz kezelése előnyösebb a processzorban. (Ez a helyzet az 5. ábrával illusztrált példákban, ha 5–10 bytenál rövidebb átlaghosszt adnak a mixek.)

7. Processzorok közötti kapcsolatok eszköztakarékos megvalósítása

A processzorok közötti feladatmegosztás és az architektúra meghatározása az átadandó vezérlő- és állapotinformációt. Ennek az eszköze példánkban az operatív tárban levő „mail box".

A processzor-processzor kapcsolat lényegében e helyek azonosítóinak (OT címeknek) a kölcsönös átadását jelenti.

E folyamat működtetése néhány vezérlő és jelzővonallal megoldható. Tekintve a folyamat kritikusságát (különböző processzorokban párhuzamosan zajló különböző időtartamú és sebességű procedúrák használnak azonos tár-rekeszt) és a viszonylag csekély eszközigényt e jelvezeték-készlet kiépítése indokolt.

Az adatkapcsolathoz felhasználható a meglevő P_i—OT (OTV) adatkapcsolat. Átmeneti tárolóhelyül kézenfekvő az OTVLT szolgál.

8. OT—OTV kapcsolat

Az OT és OTV eszköztakarékos tervezése nem választható el egymástól. A problémák, melyek felölelik a hibajavító kódok alkalmazásának, az OT ciklusszervezésnek, a dinamikus tárolóelemek működtetésének kérdését, túlmutatnak jelen előadás keretein.

9. Összefoglalás

Kiskategóriájú multiprocesszoros felépítésű számítógép központi eleme, az operatív tár vezérlő tervezését, mint közelítő lépések sorozatát mutattuk be. Az architekturális előírások és a műszaki-gazdasági szempontok mérlegelésével fokozatosan meghatároztuk az OTV durva struktúráját és egyes strukturális rész-jellemzőit, illetve pontosítottuk az OTV-ben realizálandó funkciók körét.

Irodalom

[1] IBM System/370 model 125 Functional Characteristics (GA 33—1506—1)

[2] PDP 11/70 Processor Handbook, DEC 1976.

[3] Mitra 15 Manuel de présentation (4029 PI/FR), CII 1972.

A HARDWARE TERVEZÉS ÉS DOKUMENTÁLÁS GÉPI SEGÍTÉSE

Hinsenkamp Alfréd--Dénes György--Váradi Tiborné--Csernó János
SZKI

...vezetés

A hardware fejlesztés és tervezés nagyámú elemből álló, összetett struktúrákat kezel. Az ...-esetenként bonyolult — tervezési lépések számítógépes támogatására, végrehajtására sok ...szer, algoritmus ismert és terjedt el a gyakorlatban is.

Kevésbé kidolgozott viszont a tervezés menete során előálló, folyamatosan bővülő informá...fogó tárolásának, ellenőrzésének, dokumentálásának gépi segítése. Ez ugyanis nem első...an algoritmus-kérdés, hanem a hardware fejlesztés információ struktúrájának gépi kezelését ...(mind a fejlesztett eszköz, mind a fejlesztési tevékenység vonatkozásában).

Utóbbi téren biztosan kevésbé látványosak az eredmények, mint a másik kategóriában, vi...a közvetlen, élőmunka-megtakarításban jelentkező haszon túlmenően a pontosság növekedése ...a közvetett, forintban csak nehezen kalkulálható haszon rendkívül jelentős.

- a fejlesztés átfutási ideje csökken a szubjektív hibák egy jelentős részének idejekorán való felismerése és kijavítása következtében,
- a fejlesztés eredménye emiatt minőségében is javul,
- a dokumentáció nem tartalmazza a kézi dokumentálás esetén elkerülhetetlen elírásokat, hibákat.

Az SZKI-ban az aktuális fejlesztésekből adódó sürgető igények kielégítésére kezdetben az ...ritmikus megközelítést választottuk, és elsősorban a tervezési lépések gépi megvalósítását ...ük ki célul. A későbbiekben egyre inkább felmerült az igény az interaktív módszerek, meg...ok alkalmazására. A két megközelítésből két önálló, de egymással szoros kapcsolatban álló ...szer alakult ki.

Az alább ismertetett Interaktív Tervezési és Dokumentációs Rendszer (ITDR) az SZKI-...mintegy két éve folyamatosan használatban van, és ezeket az előnyöket a gyakorlatban is ...bizonyította.

A megvalósított tervezői rendszer lényege, magja az az adatbázis, mely a konstrukciós ...ségek, részegységek leírását tartalmazza.

A rendszer sarokpontja az a szabálygyűjtemény, amely meghatározza az adatbázis struktú...t, az adatok elérési módját és értelmezését. Az egyes tervezési és dokumentálási funkciók ...on-külön csatlakoznak — a fenti szabályok által képzett interface-en keresztül — az adat...bázishoz.

...tervezői rendszerrel szembeni főbb követelmények

A megvalósított rendszer teljes mértékben eleget tesz a tervezői rendszerekkel szemben tá...asztott általános követelményeknek (modularitás, rugalmas bővíthetőség, áttekinthetőség stb.), ...ezek révén közvetlenül a hardware fejlesztő számára válik használhatóvá. Ezt tekintjük a ...szer egyik legfőbb előnyének. Az olyan fejlesztői rendszer ugyanis, melynek kezeléséhez spe...ális szaktudás szükséges, és így azt csak egy — a fejlesztéstől elkülönült — betanult szolgáltató ...is eredményesen alkalmazni, a fejlesztési munka hatékonyságát és minőségét nem emeli a ...kívánt mértékben.

A rendszer választott megoldását alátámasztó főbb szempontok:

- A fejlesztő (témafelelős) a munka lezárásáig (mintapéldány, illetve dokumentációs átadása) felelős minden tevékenységért és annak eredményéért, így egy számítógépes rendszer nem menti fel a témafelelőst sem a felelős döntések, sem a tervezés folyamatának, állásának ismerete alól, csupán segíti abban.
- Az alkatrészválaszték a fejlesztő aktív közreműködésével, kezdeményezésére folyamatosan bővül és fejlődik, melynek során nemcsak az új tétellel kell a korábbi katalógust kiegészíteni, hanem gyakran új paraméterekkel is.
- A konstrukció a technológia fejlődésével párhuzamosan állandóan változik, fejlődik.
- Az első néhány példány előállítása során nagyszámú változtatásra kell számítani.
- A fejlesztési munkához elválaszthatatlanul kapcsolódik a dokumentálás is, a nagyszámú egymást gyakran követő változások pontos, naprakész dokumentálásának *meghatározó* jelentősége van.
- A fejlesztés nem egyenesvonalú hatásláncal rendelkezik, hanem elágazásokat és hurkokat is tartalmaz, és ezek száma, helye az ismétlődések száma stb. a fejlesztés tárgyától, körülményeitől erősen függ és a technika fejlődésével is állandóan változik.

Ennek megfelelően a rendszer a következő elvekre alapoztuk:

a) A rendszernek nem egy feltételezett tevékenységi sort kell alapul vennie, illetve megkövetnie (esetleg változatokkal), *hanem az egyes lépések kiinduló adatainak és eredményének formátumát és megtalálási helyét az adatbázisban.*
Így a lépések a fejlesztő belátása szerint sorrendezhetők, párhuzamosíthatók, elágaztathatók, sőt, miután az adatbázis rugalmasan bővíthető, bármikor új tevékenység is beiktatható bárhová. Ezzel bármilyen, már működő gépesített tevékenység is helyettesíthető annak újabb, módosított, korszerűsített változatával.

b) A rendszernek a fontos és ismétlődő tevékenységeket kell elsősorban gépi úton elvégeznie. A többit kivételként célszerű kezelni. Kivételekre mindig kell számítani, mert egyrészt
   - az összes szempont, szabály, stb. csak a fejlesztés végére derül ki, továbbá
   - a fejlesztés közben is folyamatosan változnak a körülmények.

A kivételek gyors, rugalmas, felelős kezelése a rendszer használhatóságának mércéje.

Az ilyen alapelven felépített rendszer, mihelyt annak váza, az adatok tárolásának, visszanyerésének, beírásának módja és formája (adatbázis) és a kivételek kezelésének rendje összeállt, már üzembeállítható és folyamatosan bővíthető azáltal, hogy egyre több olyan tevékenység végezhető gépi úton, amely korábban még kivételnek volt tekinthető. Alapvető követelmény, hogy a bővítés szigorúan additív módon, a korábbi részek változatlan megtartásával legyen lehetséges.

c) A rendszernek biztosítania kell, hogy a fejlesztő (témafelelős) minden lépés után meggyőződhessen az eredmény helyességéről, tehát a tárolt adatoknak egyszerű eszközökkel, rutinszerűen hozzáférhetőnek és olvashatónak kell lenniök.

d) Minden tevékenységi lépés eredménye ezen túlmenően a körülményeknek (előállítás és felhasználás) legjobban megfelelő formában álljon elő. Ez biztosítja a következő művelet szükség esetén kézi úton való elvégzését anélkül, hogy az a rendszertől idegen volna. Az ebből származó információ-redundancia – helyes, körültekintő figyelembevétele esetén – nem okoz semmiféle nehézséget.

e) A rendszernek vezetnie kell, hogy a tervezés, azaz a tárolt adathalmaz a fejlesztés melyik stádiumában tart, milyen tervezési-dokumentálási lépéseket hajtott a fejlesztő eredményesen végre. Ez a státuszinformáció a közbenső módosítások, javítások, ismétlések után is egyértelműen kell, hogy jellemezze a fejlesztés helyzetét.

f) A rendszernek általánosságban csak a legszükségesebb, hosszú távon is állandónak tekinthető paramétereket szabad megkötésként tartalmaznia.
Az ellenőrzés során felismert hibák felelős javítását interaktív módon a fejlesztő számára lehetővé kell tenni.

A modularitás, a funkciók külön-külön hívhatósága nem jelenti azt, hogy – amennyibena a fejlesztő a használt funkciók működésének és hatásának ismeretében úgy dönt – ne lehetne a funkciókat ciklikusan ismételtetni, vagy láncolni. De az automatikus működést mindig lentről, az egyedi lépések összekapcsolásával kell megközelíteni egy hosszabb begyakorlási fázis, próbaüzem után, és sohasem felülről, közvetlen legelső célként.

3. A hardware struktúra összefüggései

A (hardware) eszközök felépítése a hatékony tervezés, dokumentálás és reprodukálás érdekében szabályosan strukturált.

A *struktúra* faág-struktúra, a struktúra *elemei* konstrukciós szempontból befejezett (rész) egységek.

A faág-struktúra azt a rendet hivatott rögzíteni, ahogy a struktúra valamely eleme alacsonyabb szintű *összetevőkből* összeáll. Bármely elem a magasabb szintű elem összetevője egyben. A struktúra szintjei az adott *konstrukciós rendszer* által meghatározott kötött *konstrukciós szintek (= szint).*

A gyakorlatban elterjedt *standard konstrukciós szintek* (emelkedő rendben):

- Alkatrész
- Kártya
- Panel
- Berendezés

Az értelmezett szintek (száma, jellemzői) és a struktúra ágai egyértelműen leírják a struktúrát.

Ahhoz, hogy az elemek minden szinten azonos kategóriákkal legyenek leírhatók, az alábbi kategóriákat, szabályokat következetesen kell alkalmazni.

A leírás fő kategóriái:

a) az elem *azonosítója.* Ennek révén hivatkozik az elemre, mint összetevőre az a magasabb szintű elem, amelynek az részét fogja képezni;

b) az elem *külső specifikációja.* Ide tartozik minden olyan paraméter, mely ahhoz szükséges, hogy az elemet behívó magasabb szintű elem azt helyére tudja illeszteni (össze tudja szerelni);

c) az elem alacsonyabb szintű közvetlen *összetevőinek* megadása azok azonosítóinak *felsorolásával;*

d) az *összetevők kapcsolatának,* összeszerelési módjának megadása az összetevők külső specifikációjának felhasználásával. (=belső kapcsolat)

Belátható, hogy ezen kategóriák következetes alkalmazása esetén a leírások hézag- és átfedés nélkül rögzítik a struktúrát.

A struktúra bármely pontján csak lefelé hivatkozás található, ez biztosítja azt, hogy bármely összetevő több helyen felhasználható legyen, illetve tetszőleges újabb helyre felhasználható legyen anélkül, hogy az összetevő leírását meg kellene változtatni.

A külső specifikáció és az ezzel összhangban levő összeszerelési előírás a gyakorlatban két fő csoportra oszlik:

- az elektromos csatlakoztatási pontok és azok összeköttetéseinek elvi megadása, továbbá
- az illeszkedés mechanikai-konstrukciós leírása.

A két csoport között a kész konstrukcióban egyértelmű összerendelés van.

4. A hardware struktúra gépi leírásának kérdései

A gépi leírás legfontosabb követelményei:

- a leírás a hardware eszköz felépítésével megegyező módon legyen strukturálva,
- a leírás a lehető legegyszerűbb eszközökkel, bármikor olvasható legyen,
- biztosítani lehessen a fejlesztő (témafelelős) felelősségét a bentlevő adatokért.

Az alkalmazott számítógép operációs rendszere által nyújtott szolgáltatások közül ezért különös jelentősége van a kiépített file-kezelési rendszernek (esetünkben ISAM) — Indexelt Szekvenciális elérési mód), a rendszerhez tartozó editáló programnak és a rendszer file-védelmi lehetőségeinek. Az olvashatóság érdekében az adatbázisban az adatokat olvasható (karakter) formában célszerű tárolni. Ezek a jellemzők természetesen nem gép-függetlenek; rendszerünk kialakításánál választanunk kellett egy — lényegében gép-független — külön adatbázis kezelő rendszer adaptálása és a gép-függő, de kész, rendelkezésre álló eszközök között. A körülmények figyelembevételével a választás az utóbbi megoldásra esett.

Az alkatrészeknél magasabb szintű összetevőket külön-külön file írja le, (bázis-file) az alkatrészek közös katalógusban helyezkednek el. A standard konstrukciós típusok adatai közül azok, melyek több összetevőnél ismétlődnek, a felesleges ismételt leírás és tárolás elkerülésére konstrukciós katalógusban tárolódnak.

Annak a szabály-rendszernek, mely interface-t képez az adatbázis és a funkciókat realizáló programok között, konstrukciós szintenként külön-külön az alábbiakat kell tartalmaznia:

- azokat a jellemzőket, megoldásokat, összefüggéseket, melyek a gépi tervezési és dokumentálási funkciój számára kötöttnek tekinthetők, és melyekkel kapcsolatban az adatbázis már nem tartalmaz adatot, továbbá
- az adatbázisban megtalálható kódolt jellemzők (tervezési paraméterek) pontos és helyes értelmezéséhez szükséges megkötéseket, összefüggéseket
  = az értelmezett jellemzők felsorolása (szóhasználat)
  = a jellemzőkre való hivatkozás (megtalálás) módja, összefüggése más jellemzőkkel,
  = a jellemző értékkészlet, az értékek értelmezése a konstrukcióban,

stb.

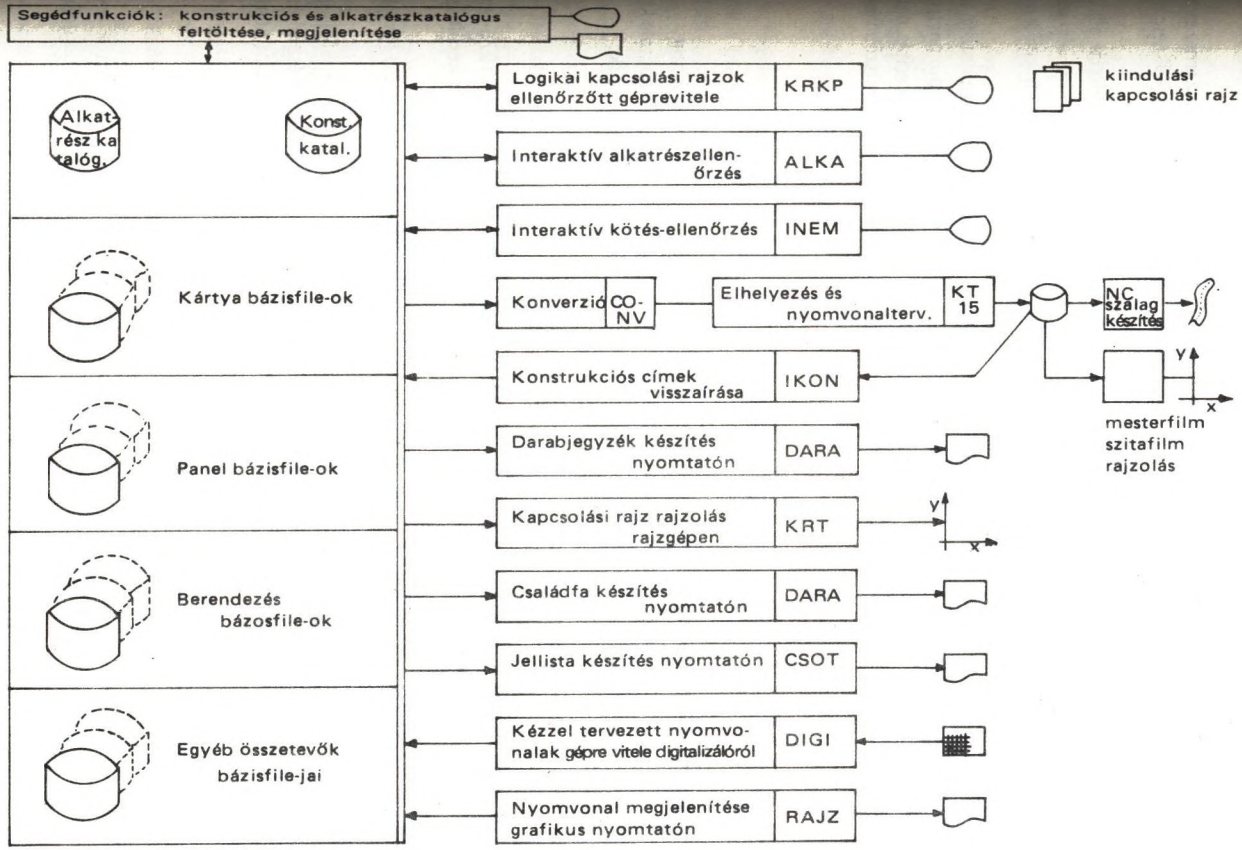
E kér fő kategória sok esetben nem határolódik el egymástól élesen.

5. A megvalósított interaktív rendszer

A rendszer az SZKI Siemens 7755 számítógépen, BS—2000 operációs rendszer alatt működik. A központi géphez telefonvonalon csatlakozó terminálok révén az interaktív programokat a fejlesztők futtathatják.

A bevezetőben vázlatosan leírt ITDR részletesebb struktúráját az 1. ábra mutatja. A rendszer az alábbi funkciókat látja el:

- kártya kapcsolási rajz szintaktikusan ellenőrzött gépre-vitele;
- a kapcsolási rajzon szereplő alkatrészek ellenőrzése a katalógussal való összevetés révén;
- a kötések helyességének konstrukciós szempontból való ellenőrzése;
- az alkatrészek elhelyezése és a nyomvonalak megtervezése a kártyán (ezt a funkciót a bevezetőben említett, automatizált rendszerhez tartozó program látja el, melyhez az alapadatokat adat-konverzióval kell eljuttatni), ennek eredményeképpen előáll rajz-


Segédfunkciók: konstrukciós és alkatrészkatalógus feltöltése, megjelenítése
Alkat. rész katalóg.
Konst. katal.
Kártya bázisfile-ok
Panel bázisfile-ok
Berendezés bázosfile-ok
Egyéb összetevők bázisfile-jai
Logikai kapcsolási rajzok ellenőrzött gépreviteleKRKP
Interaktív alkatrészellen-őrzésALKA
Interaktív kötés-ellenőrzésINEM
Konverzió CO-NV
Elhelyezés és nyomvonalterv. KT 15
Konstrukciós címek visszaírásaIKON
Darabjegyzék készítés nyomtatónDARA
Kapcsolási rajz rajzolás rajzgépenKRT
Családfa készítés nyomtatónDARA
Jellista készítés nyomtatónCSOT
Kézzel tervezett nyomvo-nalak gépre vitele digitalizálóróDIGI
Nyomvonal megjelenítése grafikus nyomtatónRAJZ
kiindulási kapcsolási rajz
NC szalag készítés
mesterfilm szitafilm rajzolás
y
x

1. ábra

gépen a nyomtatott lap készítéséhez szükséges dokumentáció: mester-filmek, szita-film, NC lyukszalag;
- a megtervezett nyomtatott lap alapján az alkatrészek konstrukciós címeinek visszaírása az adatbázisba;
- darabjegyzék készítés nyomtatón;
- kapcsolási rajz készítése rajzgépen (ezt a funkciót az automatizált rendszerhez tartozó KRT2 program-modul végzi, amellyel a 6. pontban részletesebben is foglalkozunk);
- kártyánál magasabb szintű egységek családfájának kezelése;
- kártyánál magasabb szintű egységek kapcsolati rendszerének leírása (jel-listák stb.).

Az alapinformáció géprevitele után – amennyiben valamilyen okból speciális, kézi kezelést igénylő összetevőt a tervezett eszköz nem tartalmaz – az érdemi információ kezelése mindvégig gépi eszközökkel valósítható meg.

Azáltal, hogy a kártya nyomtatásához szükséges gyártóeszközök és a dokumentáció azonos adatbázis alapján készül, elértük, hogy a legyártott eszköz és annak dokumentációja pontosan megegyezik. Ez az előny talán jelentősebb, mint az a közvetlen élőmunka-megtakarítás, mely a darabjegyzékek kézi írásából, a kapcsolási rajzok kézi írásának és rajzolásának kiküszöböléséből származik.

6. KRT2 – Elvi elektromos kapcsolási rajzot készítő programrendszer (1).

Mint a bevezetésben említettük, az SZKI-ban egy automatizált tervezőrendszer (KENTAUR) is működik, amely saját, belső adatbázisára épül és ezen keresztül kapcsolódik más gépi tervező rendszerekhez. A kapcsolat kiépítését jelenleg két vonalon végezzük:
- az előbbiekben ismertetett ITDR, valamint
- a CF–22 keretében kidolgozott AUTER (2.)

irányában.

Az automatizált rendszer egyik modulja a KRT2 programrendszer, amely az 1975-ben kidolgozott KRTI-gyel szerzett tapasztalatok felhasználásával 1978-ban készült el és került üzemszerűen használatba. Fontosabb jellemzői:
- az MSZ 9200/33–73 szabvány szerint megtervezi egy nyomtatott áramköri kártya elvi elektromos kapcsolási rajzát;
- a tervezési eredményeket mágnesszalagon állítja elő, amely alkalmas FERRANTI rajzgép, vagy mikrofilm plotter vezérlésére és így tusrajz, illetve mikrofilm elkészítésére;
- közös adatbázist használ a nyomtatott kártyát tervező és más dokumentációt készítő programokkal (ITDR);
- kevés korlátozást tartalmaz;
- fejlett – időosztásos – operációs rendszerben működik;
- terminálról, adatátviteli vonalon keresztül is futtatható.

A szabványos kapcsolási rajzon az elemeket négyszögszimbólumokkal, a közöttük levő kapcsolatokat pedig vízszintes és függőleges vonalakkal jelöljük. Azonosításra a szimbólumba írt adatok szolgálnak.

A programrendszer a következő három programból áll:
- KRT2: dokumentációtervező program,
- TEMPPP: utófeldolgozó program,
- SZALAG: adathordozó konvertáló program.

Az adatokat a következő 4 file tartalmazza.
- a bázisfile-ból vesszük a kártyának a 3, pontban ismertetett leírását;
- a „KATALOG" file-ban az alkatrészek és báziselemek elektromos leírása található;

- a „SZIMBOLUM" file r tartalmazza az egyes báziselem típusokhoz tartozó szimbólumokat;
- a „RAJZ" file a rajzgépfüggetlen, tömörített formátumú eredményt tárolja, amiből a konkrét rajzgép vezérlő utasításait tartalmazó „NC–ADAT" file készül.

A dokumentációtervező program (KRT2) integráltan végzi a teljes kártya elemeinek lapokra ogatását és a lapokon belüli összekötések megtervezését. Ezáltal lehetőség van a laprabontás odosítására, ha az összekötés-tervező programnak nem sikerül minden jelvezetéket megtervez-. Az összekötések megtervezése a két rétegre (táblára) alkalmazott, módosított Lee-algorit-ussal történik (3).

Az utolsó tervezési lépés során elkészülnek a két, vagy több lapra menő jelek lapszéli fel-tai és bekerülnek a tömörített RAJZ file-ba.

Az utófeldolgozó program (TEMPPP) felbontja a tömörített RAJZ-file utasításait az ak-ális rajzgép vezérlő utasításaivá (jelenleg FERRANTI EP–230 rajzgép, vagy mikrofilm-plot-r vezérlése kérhető), majd összegyűjti az egy laphoz tartozó adatokat és optimalizálja a rajzolást a rajzfilmeket vonalvastagságuk szerint csoportosítja és egy csoportra minimálja a rajzolási ót).

A mágnesszalag készítő program (SZALAG) az NC–ADAT file adatait viszi mágnessza-ra.

A programrendszer SIEMENS 7755-ös gépen, BS–2000 időosztásos operációs rendszerben ködik, a programozás nyelve FORTRAN és ASSEMBLER. Egyes részeknél felhasználtuk a EMENS strukturált programozást segítő COLUMBUS előfordítóját és makro rendszerét.

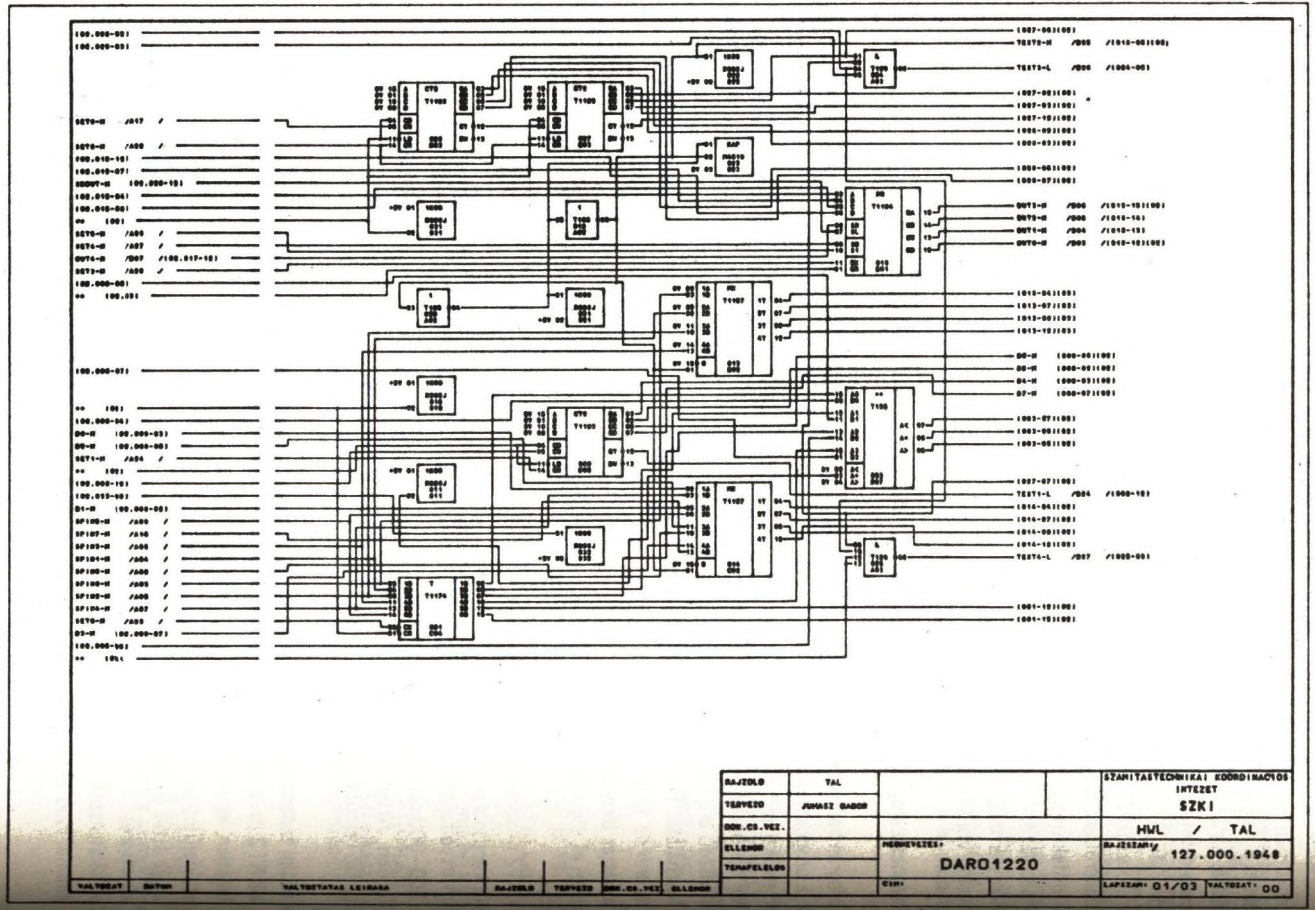
Mivel a programrendszernek különböző méretű és bonyolultságú kártyák rendkívül vál-ó mennyiségű adatát kell feldolgozni, kellemetlen korlátozást jelentett volna fix méretű mbök használata. Ezért kidolgoztunk egy FORTRAN-ból is használható *dinamikus tömb ezelési technikát* (run time support).

A módszer lényege, hogy az adattömböket a program vége utáni címtartományra dinami-san helyezi el. Egy-egy tömb számára fenntartott hely az alapérték (256 byte) egészszámú öbszöröse.

A KRT2 programrendszerrel eddig kb. 130 db dupla TEZ méretű (150x300 mm) kártya ványos kapcsolási rajzát készítettük el.

Egy kártya rajzának teljes megtervezése a rajzgépvezérlő mágnesszalag elkészültével bezá-ag kb. 500–800 sec CPU időt igényel a SIEMENS 7755-ös gépen a kártya bonyolultsági któl függően. Átlagos bonyolultságú kártya kapcsolási rajza 4–5 db. A3-as méretű lapon ké-ül el. Egy lap rajzolási ideje a FERRANTI EP–230-as rajz gépen, tussal rajzolva 40–50 perc 2 ábra).

Az eddigi tapasztalatok alapján megállapíthatjuk, hogy
- a KRT2 programrendszer használatával jelentős műszaki rajzolói kapacitás szabadul fel;
- hibátlan, egyenletes minőségű rajzok készülnek és idejekorán, akár a kártyatervezéssel egyidejűleg is;
- a rajz áttekinthető, könnyen javítható (pl. változások átvezetése) és alkalmas a jelkövetésre;
- az alkatrészek integráltságának fokozódásával szükségszerűvé válik a négyszögszimbólummal való ábrázolás, ami különben megfelel az ESZR követelményeknek is.



RAJZOLO	TAL			SZÁMÍTÁSTECHNIKAI KOORDINÁCIÓS INTÉZET SZKI
TERVEZO	JUHASZ GABOR			HWL / TAL
DOK.CS.VEZ.		MEGNEVEZES: DAR01220		RAJZSZAM: 127.000.1948
ELLENOR				
TEMAFELELOS		CIM:		LAPSZAM: 01/03 VALTOZAT: 00

VALTOZAT	DATUM	VALTOZTATAS LEIRASA	RAJZOLO	TERVEZO	DOK.CS.VEZ.	ELLENOR

Tapasztalatok, következtetések

A rendszer egészéről összefoglalóan megállapítható, hogy az az eredeti célkitűzéseknek és ...soknak megfelel és a kialakításával kapcsolatos ráfordításokat rövid idő alatt megtérí-...

A tapasztalatok igazolták a kialakításkor feltételezett szempontokat, körülményeket, az ...lvek helyességét.

Emellett meg kell említeni egy ilyen rendszer bevezetésének, alkalmazásának emberi prob-...át. A gépre-vitelhez szükséges nagyobb pontosság és az ezzel kapcsolatban elkerülhetetle-...fellépő formai korlátozások, továbbá a gépi szolgáltatások értelemszerűen kisebb rugalmas-...nem elhanyagolható nehézségeket okoz, ellenőrzőseket vált ki. Ezek alulértékelése, vagy túl ...ői felismerése adott esetben a műszakilag egyébként használható rendszer életképességét ve-...yeztetheti, ezért különös figyelmet kell fordítani) — a korábbiakban egyébként hangsúlyo-...t - követelményre, hogy a fejlesztő a gépi adatokkal bármikor, kis fáradsággal olyan kapcso-...t teremthessen, mely biztosítja a felelősségéhez értelemszerűen tartozó informálódási és be-...kozási lehetőséget.

...alomjegyzék:

KRT2. Elvi elektromos kapcsolási rajzot készítő programrendszer.
Programozók kézikönyve.
SZKI, TAL. 1979.
Számítógépek alkalmazása az elektronikus áramkörök tervezésében és ellenőrzésében.
OMFB tanulmány 1977. április.
KT15. Kétrétegű nyomtatott áramköri kártyákat tervező programrendszer.
Programozók kézikönyve.
SZKI, TAL. 1977.
ITDR. Interaktív Tervezési és Dokumentálási Rendszer
Rendszerterv HWLD. 02548–00.00–RT.00
Adatbázis HWLD. 02547–01.00–DR.))
Programok HWLD. 02560–00.00–DR.00
SZKI, HWL. 1978, 1979.

JÁRÓBETEGEK ADATAINAK NYILVÁNTARTÁSA AZ R–10-EN

Az AMSY rendszer alkalmazásai*

dr. Huhn Edit–dr. Annus János–dr. Boros Mihály
SZOTE

A következőkben egy R–10-es off-line járóbetegnyilvántartási rendszert ismertetünk. Elöljáróban néhány szót a rendszer előzményeiről:

Az 1973–75-ös években – amikor még a CII–10010-es számítógép üzemelt a számítóközpontunkban – jelentkeztek az első olyan igények, hogy bizonyos szempontok szerint fel kellene dolgozni ambuláns betegek adatait. E feldolgozások menete a következő volt. Először is a felhasználó orvos rögzítette, hogy melyek azok a kérdések, amelyekre választ szeretne kapni. Ezután szerkesztettünk olyan adatfelvételi lapokat, amelyek a vizsgálandó kérdésekhez szükséges adatokat tartalmazták. Ezeket a lapokat azután visszamenőleg az ambuláns kartonok és kórlapok alapján kitöltötték.

A felvetődött kérdések megválaszolásához szükséges táblázatokat, statisztikákat FORTRAN nyelven írt célprogramokkal állítottuk elő. Ha a menetközben felmerült újabb igényeknek megfelelően az adatlapon, vagy az előállítandó táblázatokban bármilyen módosítást hajtottunk végre, akkor a feldolgozó programokat is minden esetben módosítanunk kellett.

Igy a kérdések megfogalmazásától az eredmények kézhez kapásáig eltelt idő minden esetben túl hosszú volt.

Ehhez képest előrelépést jelentett, amikor a SZOTE Szülészeti és Nőgyógyászati Klinikájának egyik ambulanciáján három évig üzemeltettünk egy kis rendszert, amelyben az adatközlés már folyamatos volt. Olyan ambuláns kartont használtak, amely már a számítógépes feldolgozásra alkalmas volt. A kódolt betegadatok hetenként érkeztek a számítóközpontba, ahol lyukszalagra perforáltuk őket. [1]

Ezekből az adatokból állítottuk elő az ambulancia havi illetve éves betegforgalmi statisztikáit. Három év leteltével a teljes anyagot is feldolgoztuk többféle szempont szerint.

Ez idő alatt szerzett tapasztalatainkat felhasználva az R–10-es számítógép felállításakor elhatároztuk egy eléggé általános járóbetegnyilvántartási rendszer létrehozását. Modellként a SZOTE Szülészeti és Nőgyógyászati Klinikájának ambulanciáját választottuk,de törekedtünk arra, hogy a létrehozandó rendszer a speciális orvosi területtől független legyen.

A rendszer tervezésekor a következő alapvető szempontokat vettük figyelembe:

1. A rendszer off-line adatközlésre épül és olyannak kell lennie, hogy az adatfelvételi lapokon végrehajtott bármilyen változás esetén a meglevő programokat ne kelljen módosítani.
2. Az orvosi fogalmak kódolására egy egységes kódrendszert kell használni.
3. A rendszer fő célja, hogy a folyamatosan érkező betegadatokat olyan formában archiváljuk, hogy azok a későbbi tetszőleges tudományos és statisztikai feldolgozásokra alkalmasak legyenek,
4. A rendszer könnyen bővíthető legyen.

* Az előadás anyaga az Eü. Min. 3–13–0201–03–0GY számú „Számítástechnikai módszerek, rendszerek, berendezések adaptálása az orvostudományban és az egészségügyben" c. tárcaszintű kutatási főirányhoz minisztériumi szinten kiemelten elfogadott „Számítástechnikai és matematikai módszerek alkalmazása az orvostudományban és az egészségügyben" c. témában végzett kutatómunka alapján készült.

Az orvosi fogalmak kódolásának kérdése

A számítógépes orvosi információs rendszerek létrehozásának egyik fő problémája, hogy az ...i szakterület nyelve nem olyan egységes, egyértelmű, mint ahogyan ez a számítógépek ilyen ...yú felhasználásánál szükséges lenne.

A számítógépes információrendszerektől függetlenül a WHO összeállította a diagnózisok ...rchikus, kódszámokkal ellátott listáját. Ez önmagában nyilván nem elegendő, mert például ...n fontos orvosi fogalmakat, mint a műtétek, nem tartalmaz.

Hazánkban a KSH–OSZI megbízásából a SZÁMKI elődjének az INFELOR-nak a munka... 1972. és 1976. között kifejlesztették a MEDREK rendszert, amelynek szerves részét ké... egy hierarchikus struktúrájú orvosi-egészségügyi fogalomrendszer [2]. Minden fogalomhoz ...kód van hozzárendelve: egy ún. külső kód, amelyik négy jegyű sorszám jellegű kód, és egy ...kód, amely az illető fogalom fa-struktúrában való elhelyezkedését írja le.

E fogalomrendszer alapos tanulmányozása után úgy döntöttünk, hogy az orvosi fogalmak ...olására ezt fogjuk használni. 1976. végén az INFELOR-tól megkaptuk az akkor kb. 6300 ...almat tartalmazó fogalomrendszert. A nálunk jelentkezett speciális igényeknek megfelelően ...galomrendszer bizonyos bővítését végeztük el, és így az most 7868 fogalmat tartalmaz.

Az adatok felvétele és kódolása

Egy klinikai ambulancia általában több részlegből áll.

Az egyes részlegek speciális igényeit, az anyagi lehetőségeket valamint az adatfelvételt ill. ...ódolási megkönnyítésének szempontjait figyelembe véve minden részleg számára külön am... ...áns kartont – azaz a mi szempontunkból adatfelvételi lapot – terveztünk. Ez az ambuláns ...rton olyan, hogy rajta történik az adatok kódolása is.

Tekintettel ara, hogy sok olyan adat van, ami biztosan felvételre kerül függetlenül attól, ...y a beteg az ambulancia mely részlegén jelenik meg, terveztük az ún. személyi adatlapot.

A személyi adatlapnak kötelezően tartalmaznia kell a következő adatokat:

- a beteg születési ideje,
- a beteg neme,
- a beteg iker voltára jellemző adat,
- a beteg születéskori neve,
- a beteg anyjának leánykori neve,
- egy 13 karakteres betegazonosító,
- a beteg születési helye.

A betegazonosító első 12 karakterét az előbb felsorolt állandó személyi adatokból állítjuk ..., a 13. karakter egy ellenőrző jegy.

A személyi adatlap további részét tetszés szerint definiálhatjuk. Elképzeléseink szerint ...an adatokat kell tartalmaznia, mint változó személyi adatok, orvosi szempontból kritikus ada... ..., megelőző betegségek, műtétek stb.

A beteg klinikán először mindig a központi kartonozónak nevezett részlegbe megy, ahol ha ...illetőnek ez a klinikán való első jelentkezése, jól képzett egészségügyi középkáder veszi fel a ...mélyi adatlapra kerülő adatait.

Az ambulancia minden részlegén két különböző típusú adatlapot használhatnak: az egyiket ...illetőnek az ambulancián való első megjelenésekor, a másikat az ismételt kontroll-vizsgálatok ...almával.

Az adatlapoknak mindig tartalmazniuk kell a részleg kódját, a betegazonosítót és a napi dátumot.

Ettől eltekintve az adatlapok további struktúráját és tartalmát tetszőlegesen definiálhatjuk.

Az adatlapon szereplő kérdések osztályozhatók a rájuk adott válaszok típusa és a rájuk egyidejűleg adható válaszok típusa és a rájuk egyidejűleg adható válaszok száma szerint.

A kérdésekre adott válaszok a következő típusúak lehetnek:

- szöveges információ,
- számszerű adat: – kódolt adat,
  – mért érték.

Lehetnek az adatlapon olyan kérdések, amelyekre egyidejűleg csak egy válasz adható, és lehetnek olyanok, amelyekre egyidejűleg több válasz adható.

Az adatlapok kitöltése szemléletesen szólva azt jelenti, hogy egy adott részlegben minden betegnél ugyanazon kérdésekre adott esetenként különböző számú választ regisztráljuk. Minden kérdésre válaszolni kell. Tágabb értelemben válasznak tekintjük azt is, hogy „nincs adat". Az adatok kódolását naponta, a rendelési idő után egészségügyi középkáder végzi. A kódoláshoz az ambulancia minden részlege és a központi kartonozó számára külön kódolási utasítást szerkesztettünk.

Az adatrögzítés a kódolt lapok alapján lyukkártyára vagy lyukszalagra történik.

A számítógépes rendszer működése

Az AMSY egy interaktív rendszer, amely bizonyos tevékenységek végrehajtására képes. A rendszer tevékenységei fa-struktúrát alkotnak. A tevékenységek lehetnek összetettek, vagy elemiek. A felhasználónak azt, hogy a rendszer mely tevékenységét akarja végrehajtani egy alfanumerikus display-n folytatott dialógus formájában, általában többszintes választással kell meghatároznia.

Ez a következőt jelenti:

- választunk a tevékenységstruktúra egy adott szintjén lehetséges tevékenységek közül,
- ha összetett tevékenységet választottunk, amikor a tevékenység-struktúra következő szintjén újra választanunk kell,
- ha elemi tevékenységet választottunk, akkor megkezdődik az illető tevékenység végrehajtása.

Valamely elemi tevékenység végrehajtása általában több program futtatását jelenti, de ezek egymásutániságát a rendszer már automatikusan szervezi. A futtatandó programok vagy a diszk EP könyvtárában, vagy a rendszer mágnesszalagon vannak.

Mód van a tevékenység-struktúra menetközbeni módosítására: új tevékenységeket definiálhatunk, már meglevő tevékenységek definícióját megváltoztathatjuk.

Minden tevékenység-struktúrának tartalmaznia kell az adatközlés, az archiválás és a rendszergenerálás tevékenységeket.

A *rendszergenerálás* összetett tevékenység, amely a következő tevékenységeket jelenti:

- a rendszerszalag előállítása,
- tevékenység-struktúra telepítése, illetve módosítása,
- rendszerfile-ok aktualizálása: ami még mindig összetett tevékenység, és az őt megvalósító elemi tevékenységekkel történik a különböző diszkes rendszerfile-ok kezdeti tartalommal való feltöltése.

Az *adatközlés* tevékenység során történik a lyukkártyán vagy lyukszalagon beérkező beteg-adatok ellenőrzése, esetenként szükséges konverziója.

A tevékenység végrehajtása végén outputként egy olyan mágnesszalagot kapunk, amely már ...lható formában tartalmazza a betegadatokat.

Mindezek végrehajtásához a tevékenységnek bizonyos diszkes fileokra van szüksége. Röviden szólva minden lehetséges adatfelvételi laphoz tartozik egy leíró rekord, amelyik ...kező információkat tartalmazza:

- az illető kérdésre adandó válasz milyen típusú adat (szöveges információ, MEDREK-kód, MEDREK-től különböző kód, vagy mért érték — az utóbbi két esetben az illető kérdéshez konverziós rekord is tartozik, amely alapján a szükséges konverzió elvégezhető),
- az illető kérdésre egyidejűleg egy vagy több válasz adható-e,
- az illető adat a betegadatokat tartalmazó, pld. kártyafile egy fizikai rekordjában utolsó adatként szerepel-e vagy sem.

Az *archiválás* tevékenység során az újonnan érkezett adatokkal — ezeket az adatközlés ...anyság output mágnesszalagja tartalmazza — módosítjuk az archivált betegadatokat tar... többkötetes, mágnesszalagos ARCHIV file-t. Az adatok az ARCHIV file-ban fix hosszú... blokkokban vannak tárolva. Egy beteg archivált anyaga, ami tulajdonképpen az ARCHIV ...gy logikai rekordja, egy vagy több blokkban helyezkedhet el. Egy logikai rekord fizikailag ...matosan helyezkedik el a mágnesszalagon. A tevékenység végén előállítunk az ARCHIV ...z két indextáblát.

Az egyik a MUTATÓ file (mágnesszalagos):

...kordja azt mondja meg, hogy egy adott azonosítójú beteg archivált anyaga az ARCHIV ...nyadik kötetének hányadik blokkjában kezdődik. A másik egy diszkes file, ami az előbb ...ett MUTATÓ file durva indextáblája.

A rendszer további lehetséges tevékenységei két csoportba oszthatók:

1. Az első csoportba azokat a tevékenységeket soroljuk, amelyeket megvalósító programok előre definiáltak. E tevékenységek esetén bármilyen változtatás vagy újabb ilyen tevékenység definiálása a programok módosítását, illetve új programok írását teszi szükségessé. Ilyen tevékenység például a kontroll figyelés, az előírt kontroll vizsgálaton meg nem jelent betegek számára felszólító levél készítése.
2. A második csoportba pedig azok a tevékenységek tartoznak, amelyek az archivált adatok későbbi tetszőleges tudományos és statisztikai feldolgozásait biztosítják. E tevékenységek feladata az, hogy az archivált betegadatokból két másik programrendszer számára alkalmas input adatokat állítson elő. Az egyik programrendszer a FREGOLI általános kérdőívfeldolgozó rendszer, a másik a STASYS nevű matematikai statisztikai programrendszer. Mindkét programrendszert szintén az SZOTE Számítástechnikai Központjában hozták létre [3, 4].

Befejezésül az AMSY rendszer jelenlegi alkalmazásairól:

A Szülészeti és Nőgyógyászati Klinika ambulanciájának három részlegén került bevezetésre ...ndszer: az andrológia, a sterilitás és az intrauterin fogamzásgátlás ambulancián (1979. január-...).

A rendszer készíti az egyes ambulanciák havi betegforgalmi statisztikáit.

Minden részleg hetente listát kap, amely tartalmazza — napi bontásban — az illető hétre ...troll vizsgálatra berendelt betegek névsorát. Az előírt kontroll vizsgálaton meg nem jelent ...gek részére felszólító levelet készítünk. A folyamatosan érkező betegadatokat archiváljuk.

A rendszer másik alkalmazása — az I. sz. Sebészeti Klinikán — anaesthesiológiai jegyzőkönyv ...adatainak nyilvántartása.

Az archivált adatok alábbi feldolgozásait tervezzük:

1. Éves statisztikák készítése, ezek összehasonlítása a megelőző időszakokkal.
2. Tudományos célú feldolgozás klinikai és klinikofarmakológiai vizsgálatokhoz.

3. Elektív műtét esetén az esetleges régebbi műtéti anaesthesia legfontosabb adatainak ismerete.

A rendszer egyéves alkalmazásának legfőbb tapasztalata az volt, hogy a hetente ill. havonta előállítandó listák esetén az időkövetelményeket nem tudtuk teljesíteni, mivel az egyes ambulanciákról az adatok késve érkeztek.

Az adatszolgáltatás meggyorsítására a jövőben a GIN–S rendszer [5] felhasználásával on-line adatgyűjtést kívánjuk megvalósítani.

Irodalom

[1] Annus dr., Huhn E. dr., és Oroján I. dr.: Méhenbelüli fogamzásgátlót viselők adatainak tárolása, feldolgozása és értékelése számítógéppel. Magyar Nőorvosok Lapja, 38. o. 1975.

[2] Dr. Széphalmi G.: A MEDREK programrendszer leírása felhasználók számára MEDREK-TK Inf. 1449–1975.

[3] Szerényi L., Kasza F., Stéhlikné Boda K., Győri I., Forczek E.: Adatfeldolgozás céljait szolgáló programrendszer R–10-es számítógépre, különös tekintettel a biometriai feldolgozásra. NJSZT. 7. Kollokvium, Szeged, 1976.

[4] Török R., Kasza F.: Kérdőívek feldolgozására szolgáló általános programcsomag (FREGOLI) az orvosi alkalmazások tükrében 1978. NJSZT 10. Kollokvium, Szeged.

[5] Pasek B., Benedek Sz.: Egy kórházi információrendszer input tevékenységeit megvalósító programrendszer jellemzői. Programozási rendszerek 1978. Szeged.

RENDSZERELEMZÉS, OPERÁCIÓKUTATÁS, IRÁNYÍTÁS

Dr. Jándy Géza
BME

A Neumann Társaság mindkét jogelődjének a neve tartalmazta az *operációkutatás* cím-
t is. Társasággá alakulásunkkor nevünkből ez a szó elmaradt, de a MTESZ-en belül az ope-
ációkutatás legfőbb fóruma továbbra is Operációkutatási Szakosztályunk maradt. És Neumann
os nevével ez nem is összeegyeztethetetlen, hiszen jelentős szerepe volt az operációkutatási
ódszerek kifejlesztésében is. Itt nemcsak a játékelméletre kell gondolnunk, hanem olyan,
onyire beszélgetések során kipattant gondolataira is, mint pl. a lineáris programozás pri-
- és duálmegfogalmazásának a szimmetriája.

Az operációkutatás kialakulásának történetét már sok konferencián elmondták és maga-
t sem akarom ismételni. De arra mégis szeretnék figyelmeztetni, hogy az *operációkutatás*
gimát a valamely irányító hatóság, nagyobb vagy kisebb vezetett szervezet számára nagy
mlakban kitűzött feladat megoldása tudatos *tervezésének*, a feladatmegoldás megvalósítása
datos *irányításának* az igénylése hozta létre. A tudatosságra törekvés elsősorban a matemati-
ai és a statisztikával kapcsolta össze, így fontos eszköze lett a számítógép is; de az operá-
k (vagyis a stratégiai és a taktikai műveletek) tervezése és irányítása a vezetés elméletével
gyakorlatával olyan szoros rokonságba hozta az operációkutatást, hogy egyik amerikai tár-
sága és szaklapja ma is a „vezetéstudomány" nevet viseli; Angliában St. Beer megkísérelte
ztési kibernetikára átkeresztelni (és ezzel csak akkor hagyott fel, amikor elnöke lett az
gol operációkutatási társaságnak), de hosszú ideig a Szovjetunióban is szívesebben használ-
k a gazdasági kibernetika elnevezést.

Ilyen értelemben az operációkutatás elterjedése és sikere egyrészt attól függ, hogy mi-
n hatékonyan képes segíteni a vezetést a döntéshozatalban, vagyis az operációk tervezésé-
n és irányításában, de másrészt függ attól is, hogy az operációk vezetése mennyire kultú-
, mennyire törekszik a tervezésben és az irányításban a tudatosságra, ill. mennyire várják
s el tőle. Ha nem elég erős ez az igény és gyenge a fogadókészség, akkor az operációkuta-
s csak mint egy matematikai ág fejlődhet, méghozzá nem is annyira alkalmazott, hanem in-
ább elméleti. Szerencséjük a matematikus operációkutatóknak, hogy az operációkutatás cím-
ával összefoglalt modellek és módszerek (ami e kifejezésnek egy elterjedten használt másik
neve) a vezetési funkciókból fakadó döntéshozatal mellett gyakran hasznosan alkalmazha-
k más területeken, így néha például a műszaki objektumok tervezésében is. De maradjunk
ost az operációkutatás fő területén, a vezetés tervezési és irányítási funkciójával kapcsolatos
ndszerelemzésnél és optimálásnál. A vezetés szervezési funkcióját itt külön azért nem hang-
ozom, mert az operációkutatásban a tervet megvalósító *szervezet* és azon belül a *munka-*
üzemszervezés lehetőségei részben adottságok, részben éppen a terv által is alátámasztott
relátás alapján a feltárt korlátok között fejleszthetők és átalakíthatók. Akkor pedig az ope-
ációkutató számára ismét tervezésről és irányításról, most nevezetesen *szervezési operációk*
vezéséről és irányításáról van szó. Arra nyilván felesleges is emlékeztetnem, hogy termino-
giánkban az operáció az egymástól kölcsönösen függő és a rendszertől megkívánt eredmény
réséhez szükséges (tervezett vagy várható) ténykedéseknek (aktusoknak) az összessége.

A vezetésről beszélve nem feledkezhetünk meg arról, hogy ahhoz mindig szükséges bi-
nyos – az adott társadalom történelmi és politikai helyzetének, etikai és jogi értékrendjé-
k megfelelő, előretekintően, következetesen és felelősséggel felhasznált, korlátozott *hatalom.*

Ha egy *szervezetnek* nincs *elégséges hatalommal felruházott vezetése,* akkor a szervezet entrópiája (rendezetlensége) folytonosan növekedni fog és végül *irányíthatatlanná* válik.

A modern vezetéstudomány egyik legnagyobb gondja a *komplex szervezet.* Ennek vezetésével, vagyis a vezetési funkciók ellátásával azonban a személyes vezetés önmagában már nem birkózhat meg. Kisegítéseként létre kell hozni a *vezetést közvetlenül kiszolgáló különböző szakapparátusokat* és koordinálni kell azok munkáját. Ennek szükségessége annál nyilvánvalóbb, minél bonyolultabb a szervezet. Ha a nagy és bonyolult szervezet vezetője nem akar indokolatlanul túl nagy kockázatot magára vállalni, akkor hatáskörének, vagyis hatalmának nagy részét a döntéselőkészítés, ill. a vezetésszolgálat szakosított intézményeire át is kell ruháznia. A vezetés gyakorlatában a rendszerelemzés és operációkutatás leginkább az intézményes vezetés szakapparátusaiban valósulhat meg.

A vezetés a *tervező apparátusával* kijelölteti a feladatokat és részcélokat, *szervező apparátusával* gondoskodik a célok megvalósításának folyamatos feltételeiről, és ha a végrehajtó, *megvalósító apparátus* önmagától a tervezett célokat kielégítő pontossággal és valószínűséggel (objektív és szubjektív okok miatt) nem érheti el, de a megvalósulás folyamatába beavatkozhatunk, akkor a vezetés — az ellenőrzési funkción túlmenően — egy irányítási apparátussal (pl. a két szélső esetként, gazdasági szabályozórendszerrel vagy irányító berendezéssel) gondoskodik arról, hogy a megvalósítás folyamata a kitűzött célokra irányuljon és azt kielégítő pontossággal el is érje. Ennek érdekében az irányítás magába foglalja a megvalósítás dinamikus, előrehaladó tervezését, valamint a megfigyelés, összehasonlítás és helyesbítés eszközeit is. Az említett apparátusok nem mindig határolhatók el élesen egymástól, de a funkciók, tevékenységek igen.

Például a termelésirányítás a termelési-szolgáltatási feladatok terv szerinti, ütemezett és előírt minőségű megvalósításának és az erőforrások felhasználásának megfigyelésén, felülvizsgálatán, valamint az idejében beavatkozás lehetőségén alapszik. Ezért olyan kérdésekkel is foglalkoznia kell, mint az irányítás hierarchikus szintjeinek célrendszerei, struktúrája, támpontjai, ciklusideje, az irányítás folyamatában elkerülhetetlen késések figyelembe vétele, a termelésirányítás adatainak számítógépes kezelése, feldolgozása és lekérdezése, és egyáltalán a termelési folyamatok irányíthatósága és időfüggősége.

A tervezés és irányítás mellett meg kell említenem a *vezetési előrelátás* koncepcióját, amely — mint tanulófolyamat — a vezetési probléma felismerésével, a multbeli tapasztalatok és adatok gyűjtésével kezdődik, majd vizsgálja a probléma megoldási lehetőségeit, azután az eredmények figyeléséből, méréséből következtetéseket von le arra vonatkozóan, hogy hogyan kell a döntést módosítani, és az így szerzett ismereteket megőrizni a későbbi döntések alátámasztása érdekében. Természetesen mindebben a piac figyelésének, a költségek és a készletek mérésének és a nyereségjóslásnak alapvető szerepe van.

A vezetett szervezet rendszerként megközelítésében, más szóval *rendszerelemzésében* jól használhatók a Churchman által javasolt aspektusok:

1. a teljes rendszer célja és — ami még lényegesebb — a teljes rendszer teljesítményének mértéke;
2. a rendszer környezete, különösen a kényszerítő feltételek;
3. a rendszer erőforrásai, segédeszközei, pontosabban saját termelési tényezői;
4. a rendszer összetevői (komponensei), azok tevékenységei, céljai és teljesítménymértékük;
5. a rendszer vezetése.

A rendszer célján itt természetesen elvárt, megkövetelt funkciót és főfolyamatát kell értenünk. A rendszer céljai az idő múlásával, a környezeti elvárások és feltételek változásai miatt, előre nem látható vagy csak pontatlanul prognosztizálható folyamatban maguk is változhatnak. Hiányolható a churchmani vázban a történelmi aspektus, az időbeliség, a múlt, a jelen és a jövő kapcsolatának, vagyis a rendszer történetének és prognózisának feltárása, bár ez az öt szempont

egyikébe bele is érthető, ill. másképpen ez a rendszermegközelítési váz eredményesen nem használható.

Fogalmazhatnánk úgy is, hogy az operációk tervezésével és irányításával kapcsolatos rendelemzés feladata a teljes rendszer

- céljainak, teljesítménymértékének,
- környezetének, határvonalának, korlátozó feltételeinek, és a felhasználói elvárásoknak, telményeknek,
- saját termelési tényezőinek, eszközeinek,
- tevékenységeinek, azok kapcsolódó folyamatának és struktúrájának,
- kommunikációs és döntési-irányítási alrendszereinek,
- és a használható probléma-megoldó alternatíváknak feltárása, elemzése, értékelése.

Általánosságban a *rendszerelemzés* valamely helyzetre, jelenségre, feladatra vonatkozóan kisebb működési egységnek, vagy nagyobb ökológiai-társadalmi-gazdasági-műszaki komplumnak a környezetével együttes megismerése, céljainak, működésmódjának, kapcsolatainak, ktúrája és folyamatai befolyásolásának, megváltoztatásának, valamint hatásainak, következményeinek előretekintő tanulmányozása. Ilyen értelemben beszélünk a *rendszerkutatásról* is. lehet a rendszerelemzés egyszerűen feladatelemzés is, vagy általánosabban valamely társadalmi jelentőségű szituációban a bonyolult döntéseket megalapozó vagy az érvelést alátámasztó kutatás útmutatója. Igénybe veszi az operációkutatás különböző módszereit, a statisztikai mzést, a hatékonyságelemzést, a prognosztizálás eszközeit, de nem nélkülözheti az intuíciót, szakértői véleményeket és a próbálgatást sem.

A rendszerelemzés tárgya az adott társadalmi, ökológiai, gazdasági, műszaki stb. helyzet- tervezendő fejlesztendő vagy üzembe helyezendő objektum, de tárgya az a viselkedési dszer is, amelyet valamely döntési probléma megfogalmazása érdekében kell elemeznünk. utóbbi esetben a rendszerelemzés alapján nem fizikai, hanem sokkal inkább szellemi dolt kell kialakítanunk, nevezetesen a problémamegoldás modelljét.

A szakirodalomban a rendszermegközelítés és a rendszerelemzés kifejezéseket néha szinimaként is használják (pl. Churchman könyvének angol címe: „The Systems Approach" a net fordításban „Systemanalyse" lett), de az *operációkutatást* is gyakran értelmezik úgy, t meglevő komponensekből álló konkrét rendszerek elemzését. Az mindenesetre bizonyos, gy csak ismert rendszerek operációit lehet kutatni. A rendszerelemzésnek természetesen ki ll terjednie az adott szituáció, a célok, a környezet, a segédeszközök, s struktúra és a funkik, a kommunikációs, valamint a döntési és az irányítási rendszer feltárására. De magának a ndszerelemzésnek is mindig van célja, amit a *probléma felvetése,* meghatározása jelöl ki, és konkrét esetben segíti megkülönböztetni a lényegest a lényegtelentől. Például a döntési gy irányítási probléma megoldását előkészítő rendszerelemzésnek a fent említett nézőpontból a rendszert úgy kell átvilágítania, hogy láthatóvá váljanak mindazok a dolgok, körülmények, törekvések, amelyek a vizsgált döntési, irányítási problémával összefüggésben vannak zt számottevően befolyásolják. Természetesen a rendszerelemzésnek a korlátozó feltételek, környezet és a visszacsatolások mellett a rendszer elemeinek változásait, a rendszer *dinamiját és érzékenységét* is tanulmányoznia kell. Mindebben az operációkutatásnak oly nagy a erepe, hogy — mint már említettük — a két kifejezést gyakran szinonímaként használják.

A rendszerelemző modell szerkesztése arra kényszeríti a kutatót, elemzőt, hogy a rendrt minden lényeges aspektusból kvantitatíve is meghatározza és a rendszer helyes megtervéhez a szükséges adatokat összegyűjtse. Így a modellezési munka — a maga fegyelmező, ndszerező hatása miatt — már akkor is hasznos, ha számítógépi feldolgozásra, kísérletezésre m kerülhet sor.

Ezek után térjünk át az operációkutatás és az irányítás kapcsolatára. Mint tudjuk, *irányításon* (control) valamely rendszer működésének, viselkedésének befolyásolását értjük, megtározott cél elérése érdekében.

A rendszer folyamatait általában nem elég megtervezni és megszervezni, hanem irányítani is kell azokat. A *folyamat* általánosságban időben lejátszódó mennyiségi, ill. minőségi változás, átalakulás, vagyis történés, esetünkben inkább működés, cselekvés, viselkedés. Természetesen a folyamatok és az őket hordozó, megvalósító rendszerek ebben nagyon eltérhetnek egymástól, hogy befolyásolásukhoz, irányuk, állapotuk, sebességük megváltoztatásához *mekkora energiájú hatások szükségesek és azokra* milyen gyorsan vagyis *mekkora késéssel, és milyen megbízhatósággal, milyen véletlenszerűséggel reagálnak.* A rendszer reagálásának, viselkedésének alapos ismerete nélkül az irányítás nem is lehet eredményes.

Így az irányítás a kommunikáción, a rendszer működése összekapcsolódó folyamatai irányíthatóságának a felismerésén, a beavatkozási alternatívák elemzésén, az elérendő események és a részfolyamatok dinamikus tervezésén, az állapotváltozások nyomonkövetésén, a hibaforrások, szűkkeresztmetszetek, zavarok feltárásán, az irányítói rendszer megfelelő kiépítettségén, eszközellátottságán, az irányítás hatékonyságának, gazdaságosságának elemzésén és használókon alapszik. Jellegzetessége, hogy műveletei általában kis energiájú hatásokkal nagy energiájú folyamatokat befolyásolnak.

Az önműködő, vagy *automatikus* irányítás műveleteit és teljes folyamatát közvetlen emberi beavatkozás nélkül, műszaki berendezések (szervek, készülékek) valósítják meg. Akaratlagos irányításon leggyakrabban — szűk értelemben — a *kézi irányítást* értik. Azonban a nem-önműködő rendszerekben, vagyis amelyek működésében, folyamataiban az ember fizikai és szellemi munkájára is szükség van, tehát pl. a társadalmi-műszaki rendszerekben az irányítás műveletei az egymást követő működési periódusok tervei, a periódusok közbeni tervmódosítások, az írt vagy kimondott intézkedések, a folyamat közvetlen irányítójától származó jelzések, stb. A rövid ciklusidejű, sokszor reflexszerű, a folyamat közben már csak felismerést, azonosítást igénylő kézi irányítással szemben a konstruktív gondolkodást, helyzetmegítélést, a beavatkozási alternatívák felismerését, következményeik végiggondolását, választást, elhatározást igénylő, ezért általában jóval hosszabb ciklusidejű irányítást *vezetési irányításnak* vagy *intézkedéses irányításnak* nevezhetjük. A vezetés fogalomkörén belül erről, az ember által végzett irányításról van szó. Csakhogy a mai bonyolultabb ember-géprendszerek irányítását az ember önmagában már nem tudja ellátni, azok irányítói maguk is ember-gép rendszerek, vagy automaták. Például az irányított berendezés és kezelője együtt már egy nagyobb és bonyolultabb rendszert, ember-gép rendszert képeznek, amelyet általában szintén irányítanak, méghozzá esetleg önműködő irányítással. Gondoljunk csak a nagyforgalmú városi útkereszteződések, csomópontok automatikus forgalomirányítására. Vagyis az automatikus irányítás tárgya ember-gép rendszer is lehet. A műszaki és a vezetési irányítás fogalma közötti határ részleges elmosódására utal az is, hogy egyes szerzők a kézi és az automatikus irányítás mellett értelmezik az *automatizált irányítást* is. Ezzel azt az esetet különböztetik meg, amikor az irányítást a műszaki irányítóberendezések emberi közreműködéssel valósítják meg. Fordítva úgy is fogalmazhatunk, hogy ez a műszerekkel, készülékekkel, automatákkal, esetleg *számítógéppel segített irányítás.* Az ilyen irányító rendszereket szokták *automatizált irányító rendszereknek* is nevezni. Azonban az ilyen rendszerben az automaták és a számítógép nemcsak segítik az ember irányító munkáját, de helyzetmegítélése, elhatározása alapján, tehát akaratlagosan, hosszabb vagy rövidebb időszakokra az irányító ember teljesen rá is támaszkodhat az automatizált irányítási rendszer műszaki alrendszerére, akár úgy is, hogy az az emberi elhatározástól függő időszakban az önműködő irányítást valósítja meg.

A *centralizált egyszintű irányító rendszerben* (vagy más néven az egyszintű és egycélú rendszerben) egyszerűbb esetben egy-egy irányítási feladat (művelet) a teljes irányított folyamatra kiterjedően egyetlen algoritmuson is alapulhat. De lehetséges, hogy a teljes irányítási feladatkörön belül, a kapcsolódó és egymásba skatulyázott irányítási műveletek száma olyan nagy, hogy az ilyen rendszerben is ki kell dolgozni az irányítási műveleteknek az időszükség-

növekvő sorrendje szerinti hierarchiáját. Ebben a mindig sürgősen megoldandó feladatok
ebb rétegekbe, az elvi jelentőségű, de hosszabb ciklusidejű (nagyobb időszükségletű) fela-
a felsőbb rétegekbe kerülnek.

A *decentralizált egyszintű irányító rendszerek* (vagy más néven az egyszintű és többcélú
zerek) több önálló irányítási algoritmus szerint működnek, amelyek az adott részcélokra
elően végrehajtják a folyamat egyes részleteinek azonos fajtájú irányítási műveletét. A tel-
olyamat együttes céljára állandó jelleggel itt nem ügyelnek.

Ha az irányítandó rendszerre az elemek bonyolultsága, sokasága és a kapcsolódások nagy
plexitása a jellemző (ezek az ún. nagyrendszerek), akkor a minőségi és mennyiségi nehéz-
miatt a szokott módszerekkel már nem vagy csak nagyon nehézkesen irányítható. Példá-
központi optimális irányítás információfeldolgozásának, optimálásának időszükséglete na-
megnövekszik. Az elviselhetetlenül nagy ciklusidő és a gyorsan megvalósítandó beavatko-
szükségessége között az ellentmondás a teljes irányítási feladat *felbontásával* küzdhető le.
jutunk el az irányító berendezések és algoritmusok *funkcionálisan hierarchikus struktúrá-*
. A hierarchikus irányítási struktúra egymásnak alá- és fölérendelt irányítási szintekből
Ezt nevezik *többszintű irányító rendszernek.* A magasabb (fölérendelt) szint irányító be-
dezései és algoritmusai meghatározzák és összehangolják a nekik alárendelt irányító beren-
ések és algoritmusok működését. Utóbbiak szükség szerint jelentést küldenek (visszajelente-
) a felsőbb szintnek az adott kereteik között önállóan megoldott részfeladataik eredményei-
(ill. a rájuk bízott részfolyamatok, az irányított szakaszrészek állapotáról), hogy ezáltal a
rendelt hatásos döntéshozatalát lehetővé tegyék.

Rá kell mutatnunk itt arra is, hogy a bonyolult önszervező rendszerekben az input-out-
állapotokat, illetve az ok-okozati összefüggéseket nem mindig tudjuk nyomonkövetni. A
s rendszert befolyásolhatják olyan folyamatok is, amelyeket nem tudunk deduktív úton
malizálni, amelyekben az ok és okozat közötti kapcsolatokat sem funkcionális, sem való-
nűségi alakban nem tudjuk megfogalmazni. Ugyanakkor az emberiség történelme során már
néhány bonyolult rendszerben eligazodott, sőt — más eszközökkel — irányítania és fejlesz-
nie is sikerült azokat.

A bonyolult önszervező rendszerekben a hatásokat csak integráltan, az elemek sokaságán
gyszerre figyelhetjük meg. Ilyen esetekben a legjobb változat kiválasztásában *heurisztikus kri-
riumokon alapuló küszöbökre* és a valószínűségi kombinációk generátoraira támaszkodhatunk.
heurisztikus önszervezés algoritmizálása már az értelmet, problémamegoldó képességet igény-
feladatok számítógépi végrehajtásának problémájához, a *mesterséges intelligencia* fogalomkö-
hez vezet.

A vezetés segítésére hivatott *operációkutatás gyakorlati sikere nagyrészt azon múlik, hogy
ismerjük-e azt a tervezési és irányítási rendszert, amelyben ezt fel akarjuk használni, és a
ndszer ritmusával összehangoltan, a változásokat figyelembe véve dinamikusan tudjuk-e az
erációkutatási modellt működtetni.* Ehhez általában közvetlen hozzáférésű, lehetőleg párbe-
ides üzemmódban használható számítógépre van szükség. Természetesen — mint előadásom
ején is említettem — az operációkutatás gyakorlati sikeréhez az is szükséges, hogy segítségét
vezetés ténylegesen igényelje is. Köztudott, hogy a vezetés színvonala, különösen a *termelés*
zetéséé elmaradt a technikai fejlődéstől és e lemaradás behozása az operációkutatókon csak
sebb mértékben múlik, hiszen az elsősorban szakember- és vezetőképzési, de nemkevésbé ká-
rpolitikai feladat.

A Neumann Társaság Operációkutatási Szakosztálya miközben serkenti a módszertan fej-
sztését, az operációkutatás legfőbb céljának továbbra is a vezetés tervezési és irányítási funk-
ciójának, valamint a vezetési előrelátásnak a segítését tekinti. De egy számítógéptudományi
rsaság keretében predesztinálva van ez a szakosztály arra is, hogy mozdítsa elő a hozzáfér-

hető hazai operációkutatási programcsomagok hatékonyabb felhasználói dokumentálását, annak időszakos publikálását, e programcsomagok és operációkutatási szubrutinok felhasználóinak és kifejlesztőinek, alkotóinak külön-külön, de néha együttes tapasztalatcseréjét. Azt hiszem, ezen a területen ma már többet kellene tennünk.

A SZÁMÍTÁSTECHNIKA SZEREPE AZ INTEGRÁLT INTÉZETI BETEGELLÁTÁSBAN

Jávor András—Körösztes Vince—Sülyi József
MEGYEI KÓRHÁZ – RENDELŐINTÉZET
SZÁMITÓKÖZPONT, SZEKSZÁRD

1. Bevezetés

A Szekszárdi Megyei Kórház felelős Tolna megye integrált gyógyító-megelőző ellátásáért. ros egységben dolgozunk a megye másik 3 járási-városi kórházával, melyek felépítésüknél ya lényegesen kisebb kapacitásúak, s általában csak az alapszakmák (belgyógyászat, sebészet, észet-nőgyógyászat, gyermekgyógyászat) ellátását nyújtják. Kórházunk ma már a Pécsi Or- tudományi Egyetem kijelölt oktatókórháza, ami jelentős feladatokat ró ránk a medikusok korlati képzésében.

A megnövekedett feladatok, az egyre szaporodó adathalmaz újfajta adatfeldolgozás bevezeét tették szükségessé. Ezt az igényt ismerte fel jó tíz évvel ezelőtt a kórház vezetése, személy int dr. Szentgáli Gyula főigazgató-főorvos is. 1967 óta a kórház Hollerith gépparkján dolgoztuk minden ápolási esemény 27 legfontosabb adatát.

A tíz év folyamán közel 200000 ápolási eset adatai gyűltek össze, és állnak állandóan delkezésre. Ez az adatmennyiség komoly morbiditási és mortalitási, betegáramlási informákat nyújt, fontos alapja a fekvőbetegellátás megyei vezetésének és komoly szakmai, tudomás tartalommal rendelkezik.

Ez a munka teremtette meg annak lehetőségét, hogy az Egészségügyi Minisztérium és az zágos Műszaki Fejlesztési Bizottság támogatásával 1975—76-ban VIDEOTON R— 10 szágép üzembehelyezésére került sor kórházunkban. A szakmai főhatóság az alábbi feladatok olgozásával bízott meg bennünket.

1. A kórház teljes betegforgalmára kiterjedő számítógépes archiváló, visszakereső és statisztai feldolgozó rendszert hozzunk létre.
2. Az ápolási-gyógyítási folyamat számítógépes nyomonkövetését valósítsuk meg belgyógyászati osztályon.
3. A központi orvosi kiszolgáló részlegek, így mindenekelőtt a laboratórium, rtg, EKG munkáját számítógéppel támogassuk, adatait rögzítsük, visszakeressük.
4. A járóbetegellátás lényeges adatait számítógépen tároljuk, nyújtsunk módot ezek statisztai elemzéséhez.
5. Kórházgazdasági feladatokat oldjunk meg.
6. Teljes keresztmetszetében alkalmazzuk a számítástechnika eredményeit a kórházhoz rvesen kapcsolt Véradóállomás munkájában.
7. Támogassuk a kórházban folyó klinikai tudományos munkát.

Munkánk modellkísérlet, intézetünk az egyetlen közkórház, mely számítógéppel rendelik, s ebből következően olyan szervezési kialakítás a célunk, mely a későbbiekben országosan ezethető lesz.

A kórházon belül külön számítóközpontot hozunk létre, Dokumentációs és Információs zpont néven. Ennek keretében három osztály és egy orvosi munkacsoport működik.

A Számítóközpont része a Betegfelvételi iroda, a mikrofilm archiválási központ, a statisztai csoport. Igy egy területen megtalálható a hagyományos és a korszerű adatfeldolgozás, ezek rmonikus együttélése így biztosított.

2. A kórházunkban végzett munkák összefoglalása

A kórház, mint rendszer integrált egységeivel a gyógyító-megelőző-ellátó hálózat legfontosabb szervezete. Alacsonyabbfokú rendszerei egymástól jól elhatárolható, de a progresszív betegellátás elveinek megvalósítása érdekében egymással szoros szervezeti és informatív kapcsolatban vannak.

Az elhatárolás a rendszerben végbemenő fő folyamatok vetületéből adódik; (működésterületileg) a következő alacsonyabbfokú rendszereket határozhatjuk meg:

- preklinikai alacsonyabbfokú rendszer tartalmazza az alapellátást és a járóbetegellátást;
- klinikai alacsonyabbfokú rendszer maga a fekvőbeteg intézmény;
- paraklinikai alacsonyabbfokú rendszer az interklinikumban folyó ellátás kiegészítő folyamatait tartalmazza:
- therápiai és diagnosztikai alacsonyabbfokú rendszer a therápia és diagnosztikai területeket foglalja magában;
- postklinikai alacsonyabbfokú rendszer a kórházi ápolás utáni ellátási területeket tartalmazza.

Az integrált kórházi rendszer alrendszereit a főfolyamatok, valamint a főfolyamatok lezajlásának feltételeit biztosító folyamatok oldaláról közelítettük meg. Eszerint gyógyítási-ápolási alrendszert és kórházgazdasági alrendszer különböztetünk meg.

A kórházi információrendszer struktúrájának feltétlenül a kórházi struktúrához kell igazodnia, ezért a kórházi rendszert modulokra bontottuk, amelyek már jól megfoghatók, körülírhatók és a bennük lezajló folyamatok jól megismerhetők.

A modularitást a betegellátás oldaláról fogtuk fel, azaz a beteg útját követtük az integrált hálózaton belül és az ellátáshoz kapcsolódó tevékenységeket rendszereztük.

A kórházi rendszer számítógépes szervezéséhez kidolgozandó modulokat a beteg kórházba lépésétől a beteg elbocsátásáig határoztuk meg.

3. Egységes kórházi betegdokumentációs folyamat leírása

Felvétel: — a kórházban ápolt betegek adatainak dokumentálása a Felvételi irodán a felvétellel kezdődik. A felvétel során rögzítésre kerülnek a beteg személyi adatai és a felvételi adatok (törzsszám, osztály stb.). A felvételi adatok a „Kórlapfej"-re és az INFORM file-ba kerülnek; a „Kórlapfej"-et a Felvételi iroda továbbítja az ápoló osztálynak.

Ápolás: — az osztályokon az ápolás vizsgálatok kezdeményezésével kezdődik. A vizsgálatok elvégzése történhet az osztályokon és a diagnosztikai munkahelyeken (labor, röntgen stb.). A diagnosztikai munkahelyeken végzett vizsgálatok eredményei mágneslemezre és nyomtatott formában az ápoló osztályokra kerülnek. A vizsgálati eredmények alapján az ápoló orvosok meghatározzák a diagnózisokat, amiket a Kórlapfejen és/vagy a szakmai betétlapon dokumentálnak.

Elbocsátás: — az ápolás befejezése után az osztályok lezárják az ápoltak „Kórlapfej"-eit és a lezárt Kórlapfejet a Felvételi irodára továbbítják, ahol elvégzik a számítógépes elbocsátást vagy áthelyezést. Ezzel egyidejűleg az ápolás során a mágneslemezen keletkezett információk az INFORM fileból mágnesszalagra került felvételi adatokkal együtt tárolódnak.

thelyezés: – az osztályok közötti áthelyezés dokumentálását is a Felvételi iroda végzi.

4. Betegfelvételi alrendszer

A betegadminisztrációs adatállomány állandó naprakészsége többirányú adatszolgáltatás le-tőségét biztosítja:

- osztályonkénti bontásban meghatározható a pontos beteglétszám, az újonnan felvett betegek, illetve odahelyezett, valamint áthelyezett, illetve exitált betegek száma és legfontosabb azonosító adatai;
- gyógyító osztályonként pontosan kimutatható a szabad ágyak száma, illetve elhelyezkedése, ami elősegíti a rendelőintézetből konkrét kórházi ágyra történő beutalást;
- a betegélelmezési létszám a mindenkori pontos beteglétszám, illetve a matematikailag meghatározott betegforgalmi összefüggések ismeretében több napra előre becsülhető;
- a 10%-os morbiditási vizsgálatokhoz szükséges betegadminisztrációs adatok mindenkor rendelkezésre állnak.

A programrendszer üzembehelyezésével egyidőben minimális ügyviteli szervezésre volt ükség, hogy a zavartalan működtetés biztosítva legyen.

A betegfelvételi rendszer képes arra, hogy a kórház 1500 ágyán fekvő betegek adminiszt-tív felvételét, ápolási adminisztrációját, valamint az elbocsátáskor tömören összefoglalt orvosi datokat gépre juttatva feldolgozzuk, illetve későbbi ápolások céljára elemi információkat tárol-uk és a későbbiek során szolgáltassunk,

A számítógépes betegfelvétel a következő célokat szolgálja:

- a felvett betegek személyi és felvételi adatai az elbocsátásig bármikor rendelkezésre álljanak és felhasználhatók legyenek;
- egy betegre vonatkozó korábbi ápolások visszakereséséhez szükséges azonosítók rendelkezésre álljanak;
- a statisztikai kimutatásokhoz szükséges adatok rendelkezésre álljanak;
- a betegfelvétellel kapcsolatos ügyviteli feladatok központi irányítás alá – a Dokumentációs osztály hatáskörébe – kerüljenek;
- az Egységes Betegdokumentációs Rendszer alapdokumentumára épülő nyilvántartások és feldolgozások egyszerűsítése, és gyorsabbá tétele, a felesleges párhuzamos nyilvántartások megszüntetése;
- az információáramlás olyan kényszerpályára vitele, amely lehetővé teszi a teljes kórházra kiterjedő betegadminisztrációt;
- az ápolási adatok betegenkénti archiválása.

5. Ápolási-gyógyítási alrendszer

Az ápolás során rögzített és időrendbe sorolt klinikai adatok megteremtették annak lehe-tőségét, hogy a hagyományos zárójelentés helyett félautomatizált gépi zárójelentést készítsünk. Ez azt jelentette, hogy a folyamatos napi adatrögzítés mellett a beteg elbocsátásakor az orvosra ört feladat hárult:

- azoknak az adatoknak a kijelölése, amelyek jelenlétét a zárójelentésben kívánatosnak tartja;
- a diagnózis, epikrízis gépbe juttatása.

A géppel készült zárójelentés tagolt, jól olvasható, négy példányban készül és jelentősen csökkenti az orvosok adminisztratív megterhelését.

Belgyógyászati osztályon on-line ápolási alrendszer kidolgozása a cél, amelyhez jelentős mélységben standardizált struktúra kialakítása szükséges. Ez igen munkaigényes feladat és hosszabb távra szóló.

Az ápolási adatok bizonyos csoportokat alkotnak, mint az anamnézis, fizikális vizsgálat, laboratóriumi eredmények, EKG., stb. A szellemi erőforrások korlátozott volta egyszerre csak egyes modulok kidolgozását tette, illetve teszi lehetővé.

6. Laboratóriumi és röntgen alrendszer

- a laboratóriumi munkahelyeknek megfelelően a központi laboratórium megkapja a vizsgálati kérelmeket munkalisták formájában és az elvégzett vizsgálatok eredményadatait a laboratóriumban elhelyezett intelligens terminál segítségével rögzítik, majd a számítóközpont gondoskodik az adatok megfelelő helyre juttatásáról;
- a röntgenadatok hasonló módon egy kihelyezett display segítségével kerülnek rögzítésre.

7. Vérellátási alrendszer

A Megyei Vértranszfúziós Állomás működésének célja a megye egészségügyi intézményeinek ellátása az igényeknek megfelelő mennyiségű vérrel és vérkészítménnyel.

A következő folyamatok mennek végbe:

- vérbiztosítás: — donornyilvántartás;
  — kiszállásszervezés;
  — donorbehívás ;
- vérkészlet nyilvántartás;
- betegnyilvántartás (haematológiai);
- terhesnyilvántartás.

8. Gazdálkodási alrendszer

- anyag-fogyóeszköz gazdálkodás;
- gyógyszergazdálkodás;
- bér- és létszámgazdálkodás;-
- élelmezés.

a) Az anyag- és fogyóeszköz számítógépes feldolgozásának feladatait a következőkben határoztuk meg:
   - készletek analitikus nyilvántartása a könyvviteli rendszernek megfelelően;
   - főkönyvi szintetikus könyvelés igényeinek kielégítése;
   - készletfigyelés, rendelések előkészítése;
   - statisztikai feldolgozások és egyéb adatszolgáltatások;
   - az anyag-fogyóeszköz nyilvántartás és gazdálkodás komplex gépesítése.

b) A gyógyító osztályokat a kórházi gyógyszertár látja el megfelelő mennyiségű gyógyszerrel. A gyógyítási tevékenység egyik elengedhetetlen feltétele, hogy megfelelő időben mindig rendelkezésre álljon a szükséges gyógyszerkészítmény, akár hazai gyártmányú termékről van szó, akár szocialista vagy éppen tőkés importból szerezhető csak be. A havi gyógyszerforgalmi feldolgozás az anyag-fogyóeszköz nyilvántartó programokkal megoldott.

c) Az intézeti bér- és létszámgazdálkodási modul
   - kielégíti kívánt struktúrában a jelentkező adatigényeket, melyek az intézeti dolgozók létszámára, összetételére, bérezésére és az intézeti álláshelyekre vonatkoznak.

d) Az élelmezési anyagok számítógépes nyilvántartása az anyag-fogyóeszköz nyilvántartási programrendszerrel megoldott.

Ahhoz, hogy a számítástechnikát a mindennapok gyakorlatában egy egészségügyi intézményben, közkórházban használhassuk alapvetően két tényező szükséges, a számítástechnikai apparátus (személyek és gépek), valamint egy intézet, kórház, mely kellő affinitással bír az új, a számítástechnika fogadására, alkalmazására.

Az első tényezőt a számítástechnikai apparátust az OMFB és az Eü. M., valamint a Tolna megyei Tanács együttesen biztosították a kórházban létesítendő számítóközpont beruházási és üzemviteli költségeivel. E mellett a számítóközpont személyi állományának kiválasztása, betanítása és velük az egészségügyi környezet megismertetése volt a leglényegesebb feladat. Itt fel szeretnénk hívnunk arra – az általában köztudott – tényre a figyelmet, hogy egy számítógép telepítése előtt mintegy három évvel meg kell kezdeni a tudatos felkészülést a gép fogadására. A szervezési feltételek előkészítése mellett ide értjük a személyzet kialakítását is. A rendszeres létszámgondok megoldása esetünkben is 3–4 évet vett igénybe, míg eljutottunk a jelenlegi közel stabil létszámhoz. Jelentős előnyt élveznek az olyan számítóközpontok, ahol a környezetben főiskolai, egyetemi bázis biztosítja a szakember ellátást.

Az első tényező a gép és telepítése általános, minden területen érvényes. A második tényező, a kórház már speciális, egészségügyi terület. Feltétlenül biztosítani kell, hogy a számítóközpont feltétlenül az intézet első számú vezetőjéhez tartozzon, mert csak így érhető el valamennyi részleg egyértelmű támogatása, nem kerülhet egyik terület sem a számítástechnika kizárólagos, kiváltságos birtokába. Ennek igen nagy a veszélye, mert az utóbbi időben az egészségügy gazdasági területeire egyre több olyan szakember kerül, akik a népgazdaság más ágazataiban dolgoztak és ott megismerkedtek a számítástechnika előnyeivel – és hátrányaival is – és sürgetően nyúlik az egészségügyi gazdálkodás területén az egységes számítástechnikai módszerek alkalmazása. A gazdasági, statisztikai és irányítási feladatok végrehajtásához azonban nem feltétlenül szükséges a saját számítógéppark, hiszen e feladatok bérmunkában, havi ütemezéssel is jól elvégezhetők.

A leglényegesebb feladatnak az egészségügyi számítógép alkalmazások közül a betegcentrikus, a gyógyítás, ápolás, megelőzés céljait szolgáló rendszerek kialakítását, bevezetését tartjuk. Ehhez együttműködő orvosokra, egészségügyi személyzetre van szükség. Sajnos az oktatás nem követi ezt az igényt, hiszen a ma már alapvető intelligencia szintet biztosító számítástechnikai ismeretekre sem oktatják az egyetemen az orvosokat. Kénytelenek voltunk ennek pótlására 10 éven keresztül – heti egy alkalommal – oktatást biztosítani orvosaink részére, ahol az alapfokon kívül egészségügyi alkalmazásokról is tájékoztattuk a hallgatókat. Ugyanakkor a számítástechnikai szakemberek részére minimális egészségügyi oktatást is szerveztünk, így elértük azt, hogy a két szakterület művelői megértik egymást, kialakult a közös nyelv.

Jelentős feladat volt – és jelenleg is az – a passzív orvosok megnyerése, együttműködésük biztosítása. Ezt az ún. szimpatikus outputok útján igyekeztünk megnyerni. Ez alatt azt értjük, hogy az egyedi – de az általános célkitűzésekhez illeszkedő – igények kielégítésére is törekszünk, azzal támogatjuk a kutató munkát és ugyanakkor a mindennapok rutin tevékenységei során olyan végtermékeket bocsátunk a gyógyító osztályok dolgozói, orvosai, ápolószemélyzet részére, amely napi munkájukat megkönnyíti, adminisztratív terheit csökkenti, így a gyógyító, ápoló tevékenységre több idő jut.

Úgy véljük, hogy számítóközpontunk 5 éves fennállása alatt eljutottunk odáig, hogy intézetünkben jelenleg már nincs „ellenálló, szembe dolgozó" orvos és a kifejezetten együttműködő, számítástechnikai munkában résztvevő orvosok száma egyre növekszik, már a kb. 20%-ot is eléri. Ez első pillanatra alacsony számnak tűnik, de hatásában, súlyában ez a 20% biztosítani tudja az intézet szinte teljes keresztmetszetében, valamennyi gyógyító vonalon a számítástechnikai alkalmazások részére a zöld utat. Úgy véljük, hogy számítástechnikai munkánk mellett ez a legjelentősebb eredményünk és adja azt a szilárd bázist, melyen a számítástechnika közkórházi alkalmazása teljes spektrumában kibontakozhat.

GEOMETRIAI PROBLÉMÁK SZÁMÍTÓGÉPES MEGOLDÁSA

Kellermann Lászlóné – Szabó Józsefné
KLTE DEBRECEN

A számítástechnika mind szélesebb körű alkalmazása a grafikus perifériák gyors fejlődését is magával hozta és ezek is hatásos eszközei a műszaki, tudományos és gazdasági tervező-kutató munkának.

A grafikus software rendszerek a hardware oldallal párhuzamosan fejlődnek; fontos, hogy a grafikus eszközök mind hatékonyabb és sokrétűbb alkalmazását elősegítő software álljon a felhasználók rendelkezésére. A komputer grafika területén egyre több geometriai problémát kell megoldani.

Az előadás két különböző geometriai probléma számítógépes megoldásával foglalkozik: komputergrafikai megoldását és alkalmazását adja egyrészt a síkbeli szerkesztéseknek, másrészt a fotogrammetria alapfeladatainak.

Mindkét témakör eljárásai a FORTRAN–IV nyelv R–30-as reprezentánsában kerültek megírásra, a rajzok a DIGIGRAF–1008 plotteren készültek felhasználva az OKP grafikus alapsoftware-t.

1. A síkbeli szerkesztések számítógépes megvalósítása és alkalmazásai

A számítógépes grafika területén gyakran lehet találkozni olyan problémákkal, amelyek megoldása az elemi geometriai szerkesztési lépések programszerű megvalósítását igényli. Például kifejthető vonalfelületek síkra való izometrikus leképezésekor egymáshoz meghatározott módon csatlakozó, oldalaival adott háromszögeket kell szerkeszteni, vagy műszaki rajzok készítésénél egy sor érintési, metszési, távolsági-szögmérési feladatot kell megoldani.

A hagyományos értelemben vett szerkesztés mindig valamilyen rögzített eszközökkel, megengedett alaplépéseken keresztül történik [6].

Az euklideszi szerkesztés eszköze a körző és a vonalzó; a projektív szerkesztés eszköze egyedül a vonalzó; a Mohr-Mascheroni féle szerkesztések eszköze csak a körző. A geometriai szerkesztések elméletében találkozunk olyan esetekkel is, amelyek a vonalzó használata mellett egy ismert középpontú kör (ív)-t használnak fel, vagy egy másik szerkesztési csoport az egységátrakó vonalzót használó Hilbert féle szerkesztések.

Olyan esetekkel is találkozunk, amikor az eszközt vagy a szerkesztési lépéseket azzal a céllal választják meg, hogy a korábbi szerkesztések határain túlmutató nem euklideszi szerkesztéseket (köbös, transzcendens) is meg lehessen oldani.

Vannak úgynevezett közelítő szerkesztések, amelyeket a gyakorlati élet követelményei hoztak létre. Például a szabályos n oldalú sokszög szerkesztését olyan esetekben is a közelítő szerkesztés módszere szerint határozzák meg, amikor az tisztán euklideszi lépésekkel is elvégezhető lenne, de sok lépésen keresztül, és a kör kerületét is közelítő módszerrel határozzák meg.

A számológép maga is egy eszköz, amely sajátos utasítások végrehajtására alkalmas. Segítségével minden olyan feladat megoldható, melynek megoldásának a menetét a gépnek adható utasítások véges számú sorozatával le lehet írni. Az így készült program, amely alapján a számítógép a feladatát végzi a feladat megoldásának egy speciális módon rögzített algoritmusa.

A szerkesztések számítógépes megoldására elméleti szinten vannak ajánlások, eredmények [1, 2, 4]-ben.

Abból a célból, hogy a geometriai szerkesztéseket számítógéppel is el tudjuk végezni a szerkesztéseket algebrai alakra hozzuk, analitikusan írjuk le. Ily módon a hagyományos szerkesztéssel megoldható feladatok köre egyrészt bővíthető, másrészt az ily módon végzett szerkesztés bizonyos sajátosságokkal rendelkezik.

A hagyományos szerkesztéseket véges méretű és nem túlságosan nagy rajzfelületen hajtjuk végre. A számológép lehetőséget ad a rajzfelület határainak kiterjesztésére. (Ha valós affin koordinátákkal reprezentálunk egy pontot, akkor a legnagyobb abszolut értékű valós szám, amely még a memóriában tárolható szabja meg a gépi rajzfelület határait; projektív koordinátákkal dolgozva még a végrelen távoli elemeket is kezelni lehet.)

A hagyományos szerkesztésnél a használt rajzeszköz pontossága és a szerkesztő szubjektív véleménye dönti el, hogy két pont mikor azonos, két egyenesnek hol van a metszéspontja vagy a geometriai elemek illeszkednek-e stb. Ezeket a konkrét számológép valós aritmetikájának a pontossága szabja meg és itt csak bizonyos hibával tudjuk ezt megállapítani, de tetszőlegessé egységesen és lényegesen kisebb hibával.

Továbbá a hagyományos szerkesztésnél a több megoldást adó szerkesztési lépések eredményei közül szükség esetén a szerkesztő vizuálisan választ ki egyet (ha szükséges többet), vagy az önkényesen felvehető elemeknél is ő állapítja meg, hogy a konkrét feladatnál az megfelelő-e.

A számítógépes szerkesztésnél az utóbbi két dologról automatikusan kell gondoskodni.

A geometriai szerkesztések számítógépes megvalósítására készült egy eljáráscsomag, mely a következő főbb egységeket tartalmazza:

– A belső számításokat végző eljárások. (Egy pontot egy számpár reprezentál,ez a pont affin koordinátája. A belső eljárásoknál célszerű helyenként áttérni a pont homogén (projektív) koordinátás reprezentálására. A projektív síkgeometriában [3] a pont és egyenes egymás duálisai, így a duális feladatokat ugyanazon eljárással lehet megoldani).

– Az euklideszi alapszerkesztések követésére alkalmas eljárások. Ezek egyrészt tartalmazzák azokat a vizsgálatokat, hogy a szerkesztés végrehajtásához szükséges feltételek teljesülnek-e közlik, hogy van-e megoldás. Másrészt több megoldás esetén az eredménypontokat „rendezett" formában szolgáltatják, sőt a felhasználónak módja van a rendezett megoldások közül egynek, vagy többnek a kiválasztására.

– Gyakran előforduló de összetettebb euklideszi szerkesztéseket megvalósító eljárások. Ezek többnyire az előző alapszerkesztésekre épülnek és hasonlóak a leglényegesebb jellemzőik is, azonban matematikai megvalósításuk nem mindig az euklideszi szerkesztési algoritmus szerint történik.

– A nem euklideszi szerkesztéseket megvalósító eljárások.

Ilyen például csak a legegyszerűbbet említve az n oldalú szabályos sokszög szerkesztése.

A szerkesztési alapeljárásokra (melynek köre bővíthető) építve lehet egy-egy problémakör tipikus vagy még komplexebb grafikus alakzatait előállító eljárásait elkészíteni, amelyek szerves részét képezik a speciális alkalmazási programcsomagoknak.

Néhány ajánlott alkalmazási terület:

1. A számítógéppel támogatott oktatás

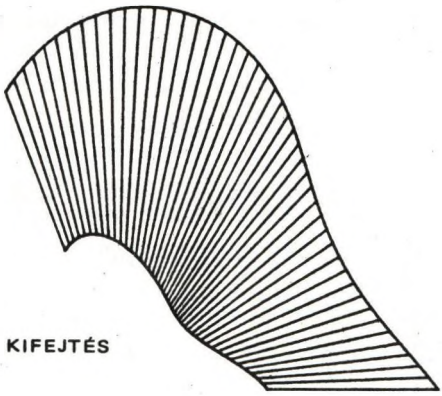
2. A számítógéppel támogatott tervezés (műszaki rajzok készítése, integrált áramkörök tervezése, építészeti problémák megoldása)

3. Egyéb alkalmazások

– Kifejthető vonalfelületek síkra való izometrikus kifejtése

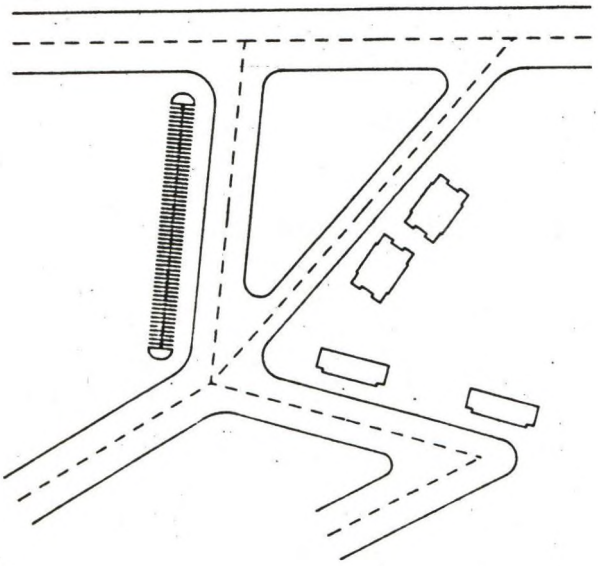
– Térbeli konstrukciós feladatok megoldásánál: Egyrészt a tér egy síkján lezajló szerkesztések megvalósításánál, másrészt egy-egy probléma megoldásánál (mértani feltételeknek eleget

tevő új térelem keresése, transzverzális feladatok (célszerű az ábrázoló geometria hagyományos módszerét [5] beiktatni a gépi megoldásnál, azaz a térbeli alakzatot egy síkra képezzük le, a képsíkon a síkbeli szerkesztő eljárásokkal állítjuk elő a kívánt térelem képét, amit azután visszaállítunk a térbe. (Ezen megoldásnál megfelelő térbeli transzformációs és leképező eljárásokra is szükség van melyekre itt nem térünk ki.)

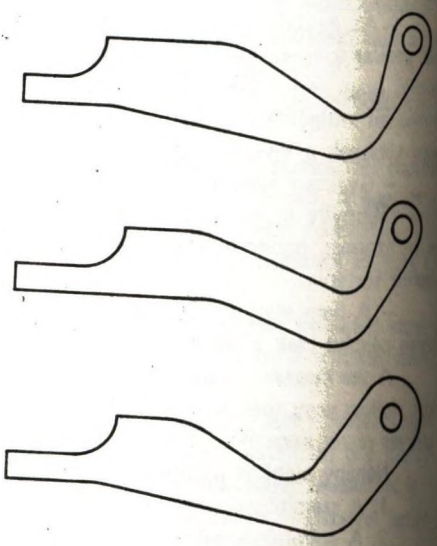


1/a ábra

1/b ábra



2. ábra



3. ábra

Az ajánlott alkalmazási területekhez néhány rajzot mellékelünk. Az 1. ábra egy ferde onka kúp kifejtésére ad példát; az 1/b ábra magának a csonka kúpnak egy axonometrikus mléltető rajza, az 1/a pedig a palást kifejtése. A 2. ábrán lakótelepek tervezésekor előfor- ló gyakoribb motívumokat látunk (különféle egymásba csatlakozó utakat lekerekített perem- nalaikkal, autóparkolót és néhány házgyári típuslakás elhelyezési rajzát). A 3. ábrán a szer- sztési eljárásokkal gépalkatrészt állítunk elő; az alkatrész maga is egy eljárásként lett megír- , annak paramétereit változtatva az alkatrész más-más alakjához jutunk.

2. Síkbeli és térbeli alakzatok valódi méreteinek meghatározása fényképeikből számítástechnikai eszközökkel.

A fotogrammetria különféle objektumok fényképeiből következtet valódi méreteikre. elhasználási területe igen széleskörű. Alkalmazható minden közvetlenül nem vagy nehezen érhető adatok meghatározására. Alapvető felhasználási területei a térképészet, műemlékvé- lem, nagyobb szerkezetek pl. hidak, tartószerkezetek, alak, vagy helyzetváltoztatásainak mé- se, gyorsan vagy folyamatosan lejátszódó jelenségek pl. karosszériadeformálódás, kémiai, fizi- i jelenségek metrikus tulajdonságainak meghatározása stb.

A fenti problémák vizsgálatára a műszaki élet speciális célirányos analóg műszereket ho- tt létre, alkalmazásukhoz azonban a felhasználható fényképekre egy sor olykor teljesíthetet- nül szigorú feltételnek is fenn kell állni.

A fent jelzett rekonstrukciós problémák megoldását csupán számítástechnikai eszközökre ptjük. A feladatok megoldására szükség van egy digitalizáló berendezésre, egy számológépre egy grafikus berendezésre.

A feladatok számítástechnikai megoldása során modellekkel dolgozunk. Matematikai mo- ell volt a kiértékelendő objektum, ennek különféle felhasznált képei, és ennek tekinthető az redményül kapott információ is. Ezáltal kiküszöbölést nyert a bemérendő objektum, a fel- asznált fényképezőgép, képanyag fizikai pontatlansága; illetve megbecsülhető egy hibahatár is. leg lehet adni a digitalizáló berendezés pontosságára vonatkozó paramétereket attól függően, ogy az eredményként jelentkező rajz pontosságára milyen műszaki paramétereknek kell majd eljesülnie.

A modellen szerepeltetni lehet rendkívül kritikus részeket is, melyekkel a program pon- sága grafikusan is jól érzékeltethető.

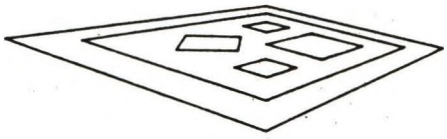
A hagyományos feldolgozási módnál számos különféle esetet szokás számba venni, ma- matikailag azonban alapvetően két problémáról van szó, aszerint, hogy a kiértékelendő ob- ktum síkbeli vagy gyakorlatilag annak tekinthető, illetve, hogy az objektum térbeli. Az utób- i esetet két részre bontjuk és ezek egyike olyan kiértékeléseket is lehetővé tesz, amelyek a otogrammetria hagyományos módszereivel, műszereivel nem lennének elvégezhetők. Síkbeli bjektum rekonstrukciójánál a követelmény olyan rajz előállítása, amely a fényképen ábrázolt lakzatot meghatározott méretarányban kellő pontossággal ábrázolja. Térbeli objektumok ese- n annak fényképeiből az objektum fő nézeteit (pl. homlokrajz, alaprajz) kell előállítani gyancsak meghatározott méretarányban és kellő pontossággal. A kellő pontosság mindig az dott problémától függ, de nem több a grafikus megjelenítésben résztvevő vonalak vastagságá- ól. Az eredményül kapott pontok koordinátáit természetesen numerikus formában is kiírhat- juk.

A programok matematikai hátterét a geometriai vizsgálatoknál nélkülözhetetlen analiti- us geometrián kívül a projektív geometria [3] a sztereoszkopikus projekció [9] és a projek- v ábrázoló geometria (centrálaxonometria) egyes területeiről [7, 8] vesszük.

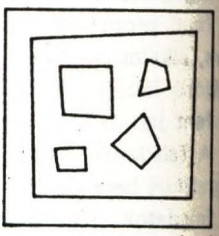
A programok:

SIKREK: Síkbeli vagy gyakorlatilag annak tekinthető alakzat egyetlen lineáris képéből (pl. fényképéből) állítja elő annak valódi vagy méretarány szerint kicsinyített képét. Matematikai hátteréül az $x_i = a_{ik}x_k$ (i, k = 1, 2, 3) projektív transzformáció mátrixának meghatározása és a leképzés megvalósítása áll [1]. A bemenő és kimenő adatoktól eltekintve a számítások az analitikus projektív geometria eszközeivel hajtatnak végre. Mind az objektum felhasznált képén, mind — a valóságban ismerni kell 4 olyan pont koordinátáit, amelyek hármanként nincsenek egy egyenesen. Ezen alapadatokra építve a program meghatározza valamennyi, a képen bemért pont valóságbeli koordinátáit és létrehozza azokat a rajzi információkat, amelyek a megadott méretarány szerint grafikusan tüntetik fel a vizsgálatba bevont pontokat. A vizsgálatba bevont pontokat izolált pontoknak tekintjük, így azok hibái nem öröklődnek át egymásra.

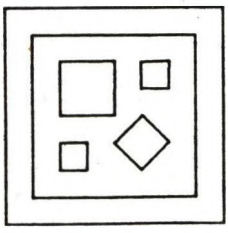
A 4/a ábrán egy síkbeli alakzat szimulált centrálprojekciós képét látjuk. Feltéve, hogy ezen a képen a pontok koordinátáit egy tetszőlegesen választott egyenlőszárú derékszögű koordinátarendszerben 0,1 cm pontossággal be tudjuk mérni, rekonstruált alakzatként a 4/b, 0,01 cm pontosság esetén a 4/c, 0,001 cm pontosság esetén a 4/d ábrát nyerjük. Az utolsó ábra pontjai koordinátáinak pontossága 0,005 cm alatt marad.



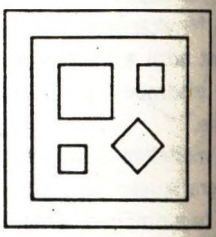
4/a ábra



4/b ábra



4/c ábra

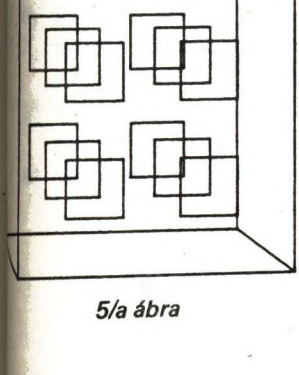


4/d ábra

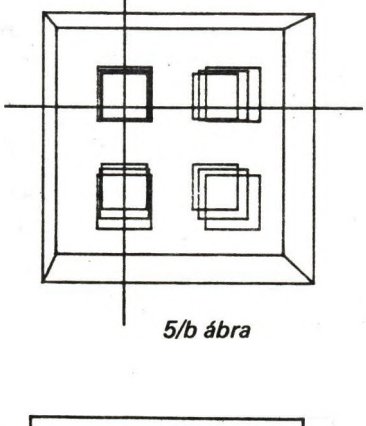
FOTOK: Ez a program akkor alkalmazható, ha a térbeli objektumnak valamilyen fő nézetét (pl. homlokrajzát) kell előállítani és lehetőség van mind a fényképező berendezés, mind a fényképezés helyének elég nagy szabadságú megválasztására. A fényképezésnek a kiválasztott főnézet síkjával párhuzamos síkban kell történnie, a felvételek készítésekor a képek síkja egy közös síkra illeszkedjék. A két felvételt ugyanazon gép (esetleg két azonos típusú) megegyező fókusztávolsággal készítse. A fényképező berendezés egyik állása a másikba párhuzamos eltolással átvihető legyen, ismerjük az eltolás mértékét és a képeken az eltolás irányát! Végül a képeken tudjuk feltüntetni a főpontot (a centrum merőleges vetületét)! A képeket nagyítani csak egyenlő mértékben szabad. Bemérésüket (a vizsgálatba bevont pontok koordinátáinak meghatározását) olyan egyenlő léptékű egyenlőszárú derékszögű koordinátarendszerben kell végezni, amelynek origói a főpontok, az eltolás iránya pedig mindkét képen azonos (pl. x) tengellyel esik egybe. Ezek a feltételek a jelenleg is létező, szabad környezetben lévő

bjektumok esetén általában kielégíthetők. A program matematikai hátterét a sztereoszkopi-us ábrázolás (anaglif ábrázolás) megfordítása adja [9]. A méretarány megválasztása után a rogram létrehozza a vizsgálatba bevont pontok merőleges vetületeinek koordinátáit és azokat rajzi információkat, amelyekkel a vetület grafikusan megjeleníthető.

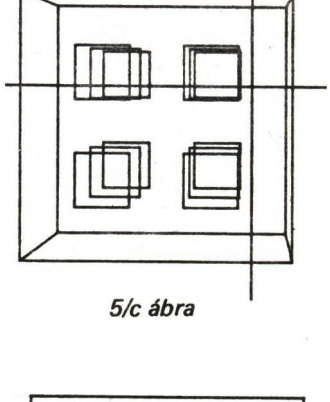
Az 5/a ábra a térbeli modell egy szemléltető rajza, centrálprojekciós képe. Az 5/b ábrán „baloldali", az 5/c ábrán a „jobboldali" speciális centrálprojekciós képet tüntettük fel, a fő-ont origója koordináta rendszerekkel. A képek 0,1 cm pontosságú beméréséből az 5/d, a 0,01 m pontosságú beméréséből az 5/e és végül 0,001 cm pontosságú beméréséből az 5/f ábrát kap-uk. A 0,01 cm-es pontosságú bemérés eredménye már 0,005 hibahatár alatti képet ad.



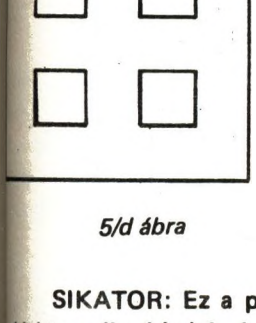
5/a ábra



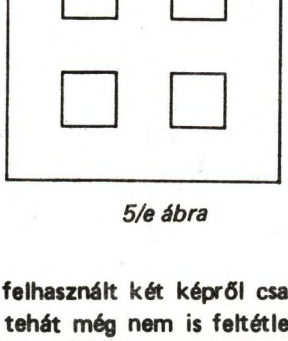
5/b ábra



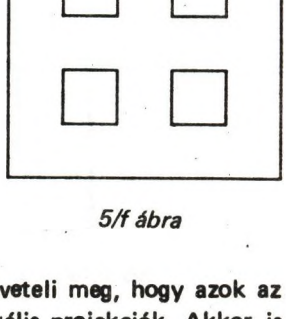
5/c ábra



5/d ábra



5/e ábra

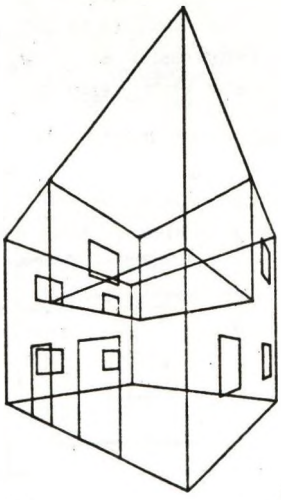


5/f ábra

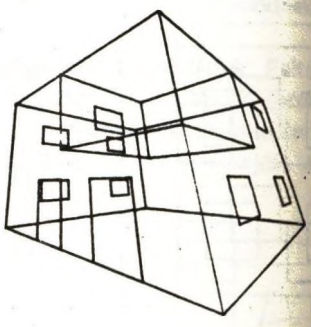
SIKATOR: Ez a program a felhasznált két képről csak azt követeli meg, hogy azok az bjektum lineáris képei legyenek, tehát még nem is feltétlenül centrális projekciók. Akkor is lkalmazható tehát, ha a bemérendő objektumról a környezete miatt nem tudunk a kiérté-elhető analóg műszerek számára megfelelő fényképeket készíteni, és akkor is, ha az objek-um részben már nem is létezik, róla csak ismeretlen eredetű felvételek állnak rendelkezésre. A program használatához — a jelenlegi állapotban — azonosítani kell tudni mind a valóság-ban, mind a fényképeken egy derékszögű egyenlőszárú tengelykeresztnek megfelelő alakzatot és a tengelyekkel a valóságban párhuzamosnak tekinthető egy-egy egyenest a képeken.

A program projektív invariánsokra épül, a számolások az analitikus projektív síkgeomet-ria apparátusával, szintetikus geometriai meggondolások alapján hajtatnak végre. A pontos ma-tematikai háttér a [2] és [3]-ban található. A felhasznált képeken a vizsgálatba bevont pontok koordinátáit tetszőlegesen választott derékszögű, egyenlőszárú koordinátarendszerben kell be-mérni. A kimenet megszervezése a FOTOK programéhoz hasonló A geometriai gondolatmenet hagyományos úton a pontossági követelmények miatt nem lenne kivitelezhető

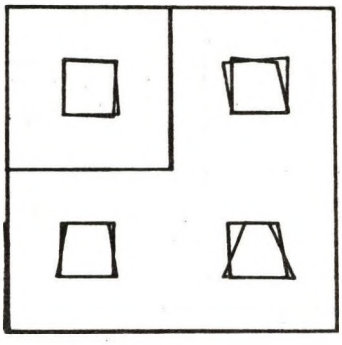
A 6/a és 6/b ábrán egy térbeli alakzat két szimulált lineáris képét látjuk. Egyik kép sem centrális projekció csak ahhoz projektív. Ha a képek bemérését tetszőlegesen választott egyenlőszárú derékszögű koordinátarendszerben végezzük, akkor 0,01 cm pontosság esetén a 6/c, 0,001 esetén a 6/d, 0,0001 esetén a 6/e ábrát kapjuk, A 0,005 cm hiba alatti eredményt csak a mikron pontosságú bemméréssel érhetjük el.



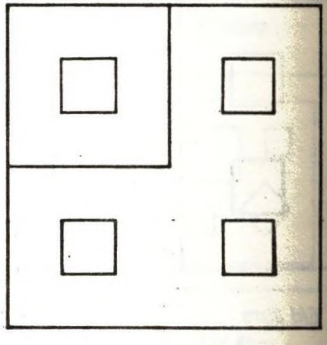
6/a ábra



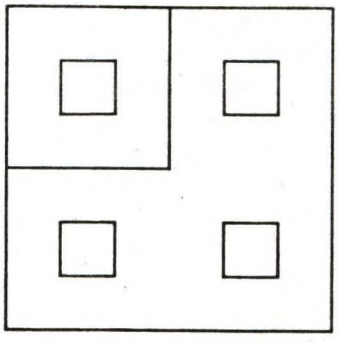
6/b ábra



6/c ábra



6/d ábra



6/e ábra

[1] Geise, G. Ein Kalkür für geometrische Aufgaben, Rostocker Mathematisches Kolloquium 6. füzet. Rostock, 1977. 95–102 o.
[2] Franze, K.: Zur algorithmischen Notation von Konstruktionsbeschreibungen als Vorstufe der Digitalgraphik, Rostocker Mathematisches Kolloquium, 6. füzet, Rostock, 1977. 81–94 o.
[3] Kerékjártó B.: A geometria alapjairól II. Projektív geometria, Bp. 1944. 613 o.
[4] Klix, W.D.: Algorithmische Aspekte des rechnergestützten (geometrischen) Konstruierens, Vortrage zur Digitalgeometrie und Bildschirmtechnik 34/78. füzet. Dresden TU, 1978. 91–99 o.
[5] Petrich G.: Ábrázoló geometria, Bp. 1973. 413 o.
[6] Szőkefalvi Nagy Gy.: A geometriai szerkesztések elmélete, Bp. 1968, 157 o.
[7] Szabó J.: Eine Verallgemeinerung des Eckhartschen Einschneideverfahrens, Publ. Math. 14. 1967, 311–319 old.
[8] Szabó J.: Az Eckhart-féle összemetszési eljárás egy általánosítása és annak néhány komputergrafikai alkalmazása. Kandidátusi disszertáció, Debrecen, 1978. 149 old.
[9] Wunderlich W.: Darstellende Geometrie II. Mannheim, 1967. 234 old.

LEKÉRDEZŐ RENDSZER A PÉNZÜGYI INFORMÁCIÓRENDSZER ADATBANKJÁHOZ

Kisfaludy Sándor—Póti Imréné
PM SZÁMITÓKÖZPONT

A pénzügyi információrendszerben (PIR) egymás után épülnek fel a meghatározott céllal on-line üzemmódban is rendelkezésre álló állományok. Ilyen például a lakossági adó adataiból felépült adatbázis, illetve a pénzügyek áttekintéséhez széles körű lehetőséget biztosító pénzügyi adatbank. További témák például: költségvetési kapcsolatok, illetmény adatok, költségvetési pénzforgalom stb., ilyen elvekre épülő megoldása is soron van. Ezek az adatbázisok SESAM BS2000 adatbankként folyamatosan kerülnek kialakításra. A differenciált tájékoztatási igény szükségessé tette olyan adatkezelő rendszerek elkészítését, mely az AB-ban tárolt adatok visszakeresését és azokkal különböző műveletek elvégzését — gyorsan — programozási munka nélkül lehetővé teszi.

Ezeknek az igényeknek és a fejlesztési elképzeléseknek a kielégítésére készült a SEDI99 elemző programrendszer és egy párbeszédes dialógus program, a PIRAL (PIR adatbank lekérdező program).

SEDI99

A SEDI99 adatkezelő rendszer elsődlegesen olyan tipikus feladatok megoldására alkalmas, amelyek a szekvenciális adatfeldolgozás ismert módszereinek alkalmazását (rendezés, csoportképzés, aggregálás stb.), valamint összetett számítási eljárások alkalmazását kívánják meg, s az eredmény kinyomtatás széles körű választékát igénylik.

A SEDI99 programrendszer alkalmas adatok visszakeresésére és kiértékelésére, valamint a kiértékelés során keletkezett új adatokkal az adatbank kiegészítésére, bővítésére.

A felhasználó igényeit funkciók megoldásával közölheti a programrendszerrel. Egy-egy funkció egy-egy adatfeldolgozási résztevékenységet jelent, amelyet belső sajátosságai szorosan meghatároznak, s amely egy sor logikai összefüggést takar.

Egy feladat különböző funkciók megfelelő együttesével adható meg. Az egyes funkciók egyszerűen megadható paraméterekkel vezérelhetők. A paraméterek megadási sorrendje között.

A SEDI99 programrendszer főbb funkciói:

- Az adatok visszakeresése, lekérdezése keresőkérdéssel.
  A SEDI99 rendszer központi jelentőségű funkciója a keresőkérdés. Paramétereivel tulajdonképpen egy ún. SESAM keresőkérdést fogalmazunk meg SEKOM (SESAM KOMMUNIKÁCIÓS MODUL) nyelven. A tényleges SESAM keresőkérdést a SEKOM MODUL hozza létre.
  A keresőkérdésben kell megadni azoknak az adatmezőknek (aspektusoknak) az azonosítóit, amelyeket a feldolgozás folyamán felhasználni kívánunk. Az egy keresőkérdésben megadható mezők max. száma 50.
- Aspektusok kiválasztása, speciális algoritmusok szerint (szélső érték keresés). Csoportonként speciális algoritmusok segítségével a legnagyobb vagy legkisebb érték kiválasztását biztosítja.
- Az adatbank különböző részeiben (zónáiban) tárolt adatok között kapcsolat létesítése, láncolás, illetve dekódolás útján.

A láncolás, illetve dekódolás funkció lényege ugyanaz. Lekérdezett kódokhoz (kódaspektusokhoz) az adatbank más részében tárolt adatmezőket illeszt hozzá. Ezekkel a funkciókkal lehet a kódértékekhez azok szöveges megnevezéseit hozzárendelni.

– Az adatok rendezése, aggregálása, összesen fokozatok előállítása. Az adatbankból lekérdezett adatmondatokat ezzel a funkcióval rendezhetjük le alfanumerikusan emelkedő sorrendben. A rendezés megadható visszakeresett (tárolt) adatokra, illetve számítási eredményként kapott mutatókra is.

Abban az esetben, ha a feladatban összesenképzési funkciót is megadtunk és az összesenfokozatokra külön kikötést nem tettünk, úgy a rendezés alapján történik a különböző aggregálási szintek előállítása is. Ha az összesenfokozat képzést nem a rendezésnek megfelelően kívánjuk elvégezni, úgy ezt összesenfokozat funkcióival külön kell megadni.

– Számítások végzése.

A négy alapműveleti jellel és zárójelekkel megadott aritmetikai műveletek adatsoronként, illetve aggregálás vagy összesenfokozatok képzésénél összesen soronként végezhetők. Az összesen sorra megadott műveletek valamennyi összesen fokozatra végbemennek, az utolsó szinten képzett értékeket a rendszer tárolja, vagyis ezek további számítási eljárásokhoz tovább felhasználhatók.

Output eljárások

A visszakeresett adatok, illetve az ezekből számított eredmények kiíratása a felhasználó ...ényének megfelelően különböző formátumokkal képernyőn, (mozaik printeren) vagy sor...omtatón, ún. output fázisok segítségével történik. Ezek a fázisok a különböző funkció-együt...ekhez illesztve például automatikus vagy paraméterezett listagenerátorok szerepét tölthetik ... Szükség esetén a felhasználó is előállíthat különleges igényeket kielégítő output fázisokat.

...IRAL

A PIRAL SESAM adatbank lekérdező aritmetikai műveleteket végrehajtó párbeszédes ...ialóg program. Egyszerű kezelőnyelve, öntanító, a dialógus bármely pontján felvilágosítást adó ...hetősége azok számára is lehetővé teszi az adatbank közvetlen használatát, akik számítógépes ...meretekkel nem rendelkeznek. Kialakításánál arra törekedtünk, hogy az adatbankok használatát népszerűsítsük s a közvetlen használat adta széles körű elemzési, áttekintési lehetőséget ...egteremtsük.

A dialóg adta lehetőséget kihasználva a felhasználó igényét (feladatát) lépésről-lépésre fo...almazhatja meg, s az adott helyen az érvényes szabályokról és lehetőségekről ? megadásával ...het felvilágosítást.

A dialógust a PIRAL üzenetei, kérdései és a felhasználó válaszai alkotják, ez utóbbiak ha...rozzák meg a dialógus menetét,

A PIRAL az üzenetre minden esetben a felhasználó üzenetét, válaszát várja.

Az üzenetek három csoportba sorolhatók:

– a program tájékoztató jellegű közleményei, amelyeket kérdés követ,
– a program esetenként kérdést tesz fel a válaszlehetőségek felsorolásával,
– a program (tovább)működéséhez az adott helyen a felhasználótól kér utasítást.

A felhasználó igényeit ún. funkciók megadásával közölheti.

A PIRAL főbb funkciói:

- Adat visszakeresés
  Az adatok visszakeresésére négy funkció áll rendelkezésre.
  KERES funkció elindítja az összehasonlításokból és döntésekből álló keresési folyamatot, melyet a felhasználó a funkció utasításában keresőfeltételként meghatározott.
  A keresőfeltétel ES, VAGY NEM operátorokkal tetszőlegesen összekapcsolt adatmezőkből áll, amelyek értékeihez vagy értékcsoportjaihoz a felhasználó összehasonlító kritériumokat adhat meg.
  Válaszként egy olyan ideiglenes állomány áll elő, amely azokat a válaszokat tartalmazza, amelyek a keresőfeltételeknek megfelelnek. A kiválasztott ideiglenes állományt a felhasználó a továbbiakban úgy használhatja, mint egy új adatbankot.
  Az OLVAS funkció az ideiglenes állományra vonatkozóan indít el egy keresési folyamatot. A keresés feltételeit a KERES funkció keresőfeltételéhez azonos módon lehet megadni. Ez a funkció elsődlegesen elemzési munkáknál hasznos, amikor egy kiválasztott információs kör különböző sorait (például gazdálkodó egység) és/vagy adatait kívánják vizsgálni.
  A SZUKIT funkció az ideiglenes állomány szűkítésére szolgál. A keresőfeltételének megfelelő adatsorokat kiválaszt és ezekből új ideiglenes állományt állít elő.
  Az ideiglenes állomány a dialóg során addig áll a felhasználó rendelkezésére, míg egy KERES vagy SZUKIT funkcióval újat nem állít elő.
  Lehetőség van LEKERDEZ funkció megadásával adatbank sorok feltétel nélküli visszakeresésére. Ebben az esetben ideiglenes állomány nem kerül felépítésre.
- Adatok, számított mutatók ideiglenes tárolása a dialógus tartamára, és a tárolt adatok kiírása képernyőre.
  A TAROL funkcióval lehet egy értéket a dialógus folyamán megőrizni. A felhasználó az értéket névvel látja el, s a felhasználás során ezzel a névvel azonosítja. Egy dialógus alatt egy időben 60 érték tárolása megengedett. A TARIR funkció hatására az összes tárolt érték megjelenik a képernyőn.
- Aritmetikai műveletek végzése
  A MUVELET funkció három különböző T típusú művelet végrehajtására képes:
  - tételsoros művelet, mely a válaszok mindegyikére végrehajtódik,
  - aggregálási művelet, mely a válaszsorok kijelölt adatait és/vagy számított mutatóit felösszesíti,
  - összegsoros művelet, mely az aggregált adatokkal hajtódik végre.

  A tételsoros és összegsoros művelet a négy alapművelettel és zárójelekkel adható meg.
- Felvilágosítás a dialógus folyamán korábban megadott funkciókról.
  Felvilágosításra két funkció szolgál.
  Az UTOLSO az utolsó beadott funkcióhoz tartozó felhasználói utasítást (keresőfeltétel), az AKTUÁLIS az utolsó ideiglenes állományt előállító utasítássorozatot írja ki képernyőre.
- A dialógus megszakítása
  Két kitüntetett funkció áll rendelkezésre a dialógus menetének megszakítására. Mindkettő bármikor megadható a dialógus folyamán.
  A VALTAS funkció megszakítja a dialógust, beadása után új funkciót lehet megadni.
  A ZARAS funkció hatására a dialógus befejeződik, az adatbank lezárásra kerül.

Output eljárások

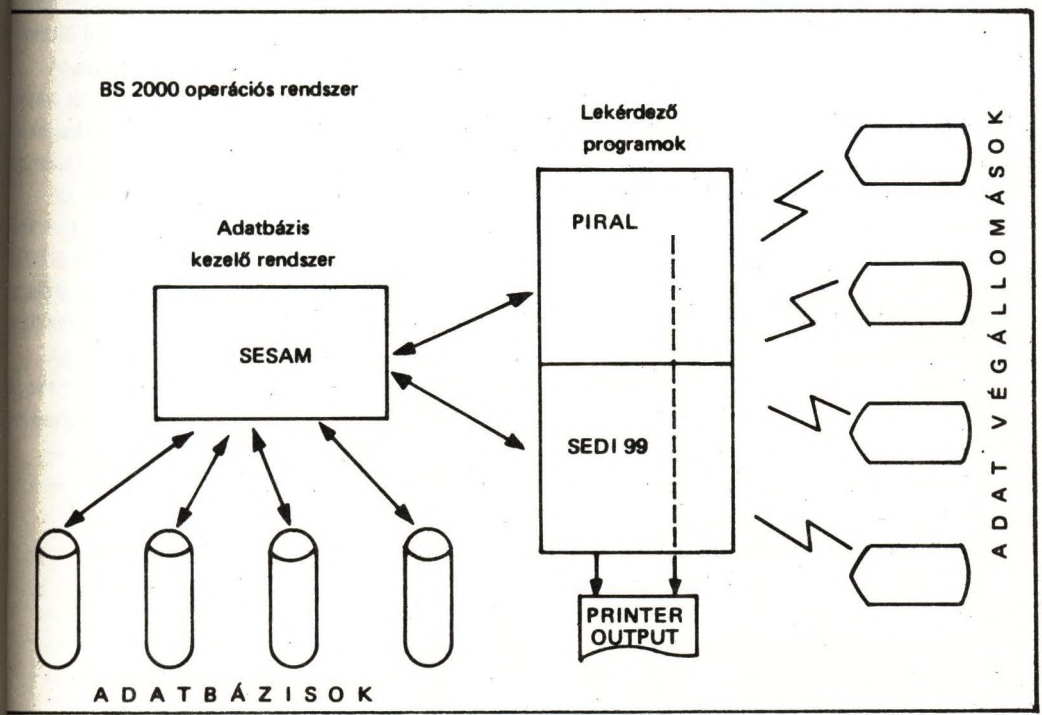
Az egyes funkciókhoz kapcsolódóan lehet a visszakeresett, illetve számolt adatokat kiíratni. Az output típusára, formátumára vonatkozóan a PIRAL kérdéseket tesz fel a válaszlehetőségek felsorolásával. Az output a felhasználó igényének megfelelően megjelenhet képernyőn, sornyomtatón, vagy soros file-ban, válaszonként, illetve összegsoronként.

— — —

A SEDI99 és a PIRAL BS2000 operációs rendszerben működtethető programrendszerek. Mindkét programrendszer használata lehetőséget ad a BS2000 operációs rendszerben működtethető különféle programcsomagok — például a közgazdasági munkában felhasználható matematikai és matematikai-statisztikai — programjainak, illetve szubrutinjainak összekapcsolására.

Ehhez a SEDI99 és a PIRAL lehetőséget biztosít a felhasználónak, hogy a különböző adatbankok lekérdezéseinek eredményei (az adatbankban tárolt, vagy számításokkal előállított adatok) külön file-ként rendezésre állnak. (lásd ábrát.)

Lekérdező rendszer vázlat a pénzügyi információrendszer adatbankjaihoz



Meg kell említeni – bár ez nem a két lekérdező rendszer adottsága – a BS2000 operációs rendszerben működő SESAM program lehetőségeit, mivel a SEDI99 és a PIRAL is a SESAM-on keresztül éri el a különféle adatbankokat.

A SESAM egyidejűleg 256 felhasználóval képes kapcsolatot fenntartani. Ez a lehetőség magában foglalja a multi-file, multi-using üzemeltetést.

Ez a lehetőség adja, hogy például a pénzügyi adatbankot egyidejűleg több adatvégállomásról (PM, területi kihelyezett TAKEH számítástechnikai osztályok. PMSZK stb.) lehet lekérdezni SEDI99 vagy PIRAL segítségével, sőt több adatbank egyidejű lekérdezése is lehetséges.

További lehetőség az is – elsősorban a SEDI99 használatakor –, hogy egyik adatbankból lekérdezett adatokat egy másik adatbankba átvigyük annak érdekében, hogy bizonyos feladatok végrehajtásához szükséges adatok egy adatbankban álljanak rendelkezésre. Ez a lehetőség üzemeltetési problémákat is megold. (Több adatbank egyidejű lekérdezéséhez több adathordozóra van szükség.)

A SZÁMÍTÁSTECHNIKAI SZAKEMBERKÉPZÉS EREDMÉNYESSÉGÉNEK VIZSGÁLATA

Dr. Kocsis András
SZÁMOK

1. Bevezetés

Egy oktatási rendszer működésének eredményességével általában keveset foglalkozunk. Az oktatási rendszereket elemezzük, vizsgáljuk, de általában nem a képzési eredmény szempontjából, hanem legtöbbször önmagukban tekintjük azokat. Pedig nagyon fontos az a vizsgálat is, amelyben arra keresünk választ, hogy az oktatási rendszer működése milyen eredményt, azaz milyen szakembereket produkál. Hiszen a kibocsátott szakemberek oldják meg a jelen és a jövő feladatait és nekik kell az illető szakterületet fejleszteni is. Éppen ezért működő oktatási rendszereket elsődlegesen az eredmény szempontjából célszerű vizsgálni. Ez lehet alapja a rendszer fejlesztésének, módosításának is. Az elemzés azonban meglehetősen nehéz feladat, egzakt eredményekhez csak nagy mennyiségű megbízható adat gyűjtése és ezeknek sokoldalú, egzakt módszerek szerinti elemzése révén lehet jutni. Intézetünk, a KSH Nemzetközi Számítástechnikai Oktató és Tájékoztató Központ, több éve próbálkozik különböző vizsgálati eljárásokkal, amelyek végül is néhány jól értékelhető adatot szolgáltattak. Jelen előadás tárgya ezeknek az elmzési módszereknek, az eredményeknek és néhány, a számítástechnikai szakemberképzést befolyásoló következtetésnek az ismertetése.

Alapvető célkitűzésünk, hogy az intézetünkben 10 éve folyó számítástechnikai szakemberképzési tevékenységünkről megállapíthassuk, hogy az mennyire felel meg a hazai számítástechnika-alkalmazás igényeinek. Nevezetesen folyamatosan szeretnénk ismerni, hogy a képzési rendszer célkitűzései helyesek-e és ezek milyen mértékben valósulnak meg.

Az igények és oktatási rendszerünk működési eredményeinek összevetéséből következtethetünk arra, hogy oktatási rendszerünk struktúrája, (a szakemberképző tanfolyamok és a közöttük fennálló tartalmi kapcsolatok) képzési kapacitása és a rendszert alkotó tanfolyamok célkitűzése, tartalma, valamint az alkalmazott oktatási módszer megfelelő-e, illetve a rendszer milyen paramétereit kell módosítani az igények magasabb szintű kielégítése érdekében.

Az oktatási rendszer igény-kielégítési szintjét a rendszer eredményessége határozza meg. Az oktatási rendszer működésének eredményességét az alábbi főbb rendszerparaméterek befolyásolják:

- *Struktúra:* a képzési struktúra és szakemberstruktúra viszonya.
- *Képzési kapacitás:* adott időszakban (általában tanévenként) kibocsátott szakemberek száma. A képzési kapacitást az alkalmazott oktatási „technológia" ismérvei (hallgatónkénti egységnyi ráfordítások: mint pl. a tanfolyam óraszám, csoportlétszám stb.) és az oktatási rendszer részére rendelkezésre álló kapacitások (tanárlétszám, tantermek száma, számítástechnikai kapacitás stb.) határozzák meg. Mennyiségi szempontból akkor biztosítható leginkább az eredményesség, ha a kibocsátás az igények szintjén van.
- *A képzés minősége* elsősorban azzal jellemezhető, hogy a kiképzett szakemberek milyen mértékben és mennyi idő elteltével képesek munkafeladataikat önállóan, a szakmai környezetbe beilleszkedve ellátni. A minőséget a tanfolyamok célkitűzése, a tanfolyamok tematikája (felépítés és tartalom) befolyásolják.

A kérdés ezekután csupán az, hogy az eredményességet meghatározó paraméterek hogyan vizsgálhatók. Elvileg viszonylag egyszerű a struktúra és a képzési kapacitás helyességének vizsgálata, ha ismerjük az igénystruktúrát (ami nem más, mint a mindenkori szakemberstruktúra) és a

mennyiségi igényeket. Bár az igényekre vonatkozóan vannak durva tervszámok és becslések, ezekre nem lehet megnyugtatóan támaszkodni, mivel a számítástechnika fejlődése eléggé dinamikus ahhoz, hogy pl. egy ötéves időszakra vonatkozó tervszámokat pontosan meg lehessen becsülni és ezeket évről-évre karban lehessen tartani. Igy tehát a strukturális és a mennyiségi igénykielégítés pontos mértékét csak erre vonatkozó vizsgálatok útján lehet meghatározni.

A minőség vizsgálata komplex feladatot jelent még akkor is, ha a minőségi követelmények hivatalosan elő vannak írva pl.: munkafeladat-leírás formájában. A minőséget legcélszerűbben éppen a végzettek munkafeladat ellátási képességével (beválási mutató) értékelhetjük, hiszen a szakemberképzésnek az a feladata, hogy a képzés résztvevőit valamilyen szakma feladatainak elvégzésére felkészítse. Általánosan elfogadott vagy rendeletben előírt munkafeladatleírás a számítástechnikai szakemberekre vonatkozóan ma még nincs, ezért az ilyen irányú vizsgálat szükségessé teszi a munkafeladat-leírások elkészítését. 1977-ben a munkafeladat-leírásokat külső szakemberek bevonásával összeállítottuk és ezt a gyűjteményt /1/ tekintjük alapnak oktatási rendszerünk működése eredményességének vizsgálata során.

A minőség elemzése során még egy módszertani problémát kell megemlíteni. A vizsgálatok során csak olyan szakemberekre vonatkozóan kell adatokat gyűjtenünk, akik intézetünkben végeztek. Ezért a nálunk végzett szakembereket kell végzésük után nyomon kísérnünk és munkaügyi, szakmai munkavégzési adataikat tudjuk felhasználni. Ennek az adatgyűjtésnek két formája lehet: a végzettek által közölt adatok, vélemények beszerzése, másrészt a nálunk végzettek környezetében dolgozó – elsősorban – vezető szakemberek véleményének beszerzése útján. Ez utóbbi körhöz való hozzáférés egy oktatási intézet számára eléggé munkaigényes. Ezért úgy döntöttünk, hogy az egyszerűbb volt-hallgatói adatgyűjtést évente, a munkáltatói adatgyűjtést 3–4 évente végezzük el. Az adatgyűjtés eredményeit statisztikai módszerekkel értékeltük. Ez a kétfajta adatgyűjtés szolgáltatja az alapot, hogy a szükséges elemzéseket elvégezhessük. Ezenkívül eseti adatgyűjtést is végeztünk.

2. Az elemzések és a kapott eredmények

2.1. *A képzési rendszer struktúrájának elemzése*

Ennek a vizsgálatnak az a célja, hogy megállapítsuk, hogy az oktatási rendszer által kibocsátott szakembestruktúra milyen mértékben fedi le az igényeket. Az oktatási rendszer kibocsátási struktúrája ismert, a feladatot a strukturális igények meghatározása jelenti. Igy tehát a vizsgálat az igények feltárására vonatkozik.

A struktúra esetében vizsgálni kell

- a munkamegosztásban bekövetkező változásokat,
- a kihalófélben levő szakmákat,
- a születő szakmákat.

A változások itt viszonylag lassúak és léteznek olyan meglevő információs csatornák, amelyek megbízható eredményeket szolgáltatnak. Ilyenek:

- a munkamegosztás helyzetének elemzése, vizsgálata,
- „A számítástechnikai foglalkozások jegyzéke a Foglalkozások Egységes Osztályozási Rendszere (Feor) szerint" /2/ c. foglalkozási gyűjtemény, amely megfelelő részletességgel tartalmazza az érvényes szakmákat;
- a tanfolyamokat megrendelő vállalatok igényei.
  A megrendelési igények mutatják, hogy milyen szakembertípusokra van növekvő, ill. csökkenő igény;
- egyéb eseti információ.

Oktatási rendszerünk és az igénystruktúra összevetéséből megállapítottuk, hogy a programtervező kivételével valamennyi kibocsátott szakembertípusra valódi igény van, de emellett oktatási rendszerünk a teljes igénystruktúrát nem fedi le. Teljes lefedést nem tudunk megvalósítani kapacitáskorlátok vagy az egyes le nem fedett szakembertípusok iránti alacsony mennyiségi igény miatt (ill. mivel más intézet végzi a képzést). Három szakma esetében tapasztaltunk olyan mérvű szakemberhiányt, ahol a képzést feltétlenül meg kell indítani a közeljövőben: az alkalmazási programtervezőknél, az üzemeltetésvezetőknél és a rendszerprogramozóknál. Ezért tanfolyamstruktúránkat ennek megfelelően módosítjuk, hogy oktatási rendszerünk eredményessége emelkedhessék. Felméréseink szerint a felhasználók számítástechnikai képzése is szükséges lenne, de ezt már kapacitáskorlátok miatt nem tudjuk megoldani.

2.2. *A képzési kapacitás elemzése*

Egy adott oktatási rendszer eredményességét az is befolyásolja, hogy a rendszer által kibocsátott szakembermennyiség mennyire felel meg az igényeknek figyelembevéve azt is, hogy más oktatási intézet is képez ki szakembereket az illető területen. Ha a kibocsátás nem éri el az igények szintjét, akkor — különösen igaz ez a számítástechnikában — folyamatos szakemberhiány lép fel és ez a szakmai feladatok megoldását gátolja, valamint a fejlődési lehetőségeket is csökkenti. A túlképzés sem kedvező. Eredménye megnyilvánulhat abban, hogy a végzetteket nem a szakképzettségüknek megfelelő munkával látják el és/vagy a szakemberek nem találnak képzettségüknek megfelelő munkát és át kell képezni magukat. Ez utóbbi olyan szempontból is káros, hogy az ország jelentős eszközöket pazarol a szakemberképzésre, amelyeket hatékonyabban lehetne felhasználni (pl. a képzés minőségének emelésében, továbbképzésében).

Mint már említettük, a pontos szakemberigények intézetünk számára nem ismertek, ezért az igények helyzetét nekünk kell időről-időre felmérni, hogy oktatásunk eredményességét megismerjük. Célunk az, hogy a mennyiségi igényeket minél jobban kielégítsük az általunk képzett szakmálban.

A vizsgálat területei

A) A számítógépet üzemeltető szervezetek szakemberigénye és a gépbeszerzésekkel kapcsolatos igények

1977-ben két különálló felmérést végeztünk a szakemberigények megállapítása érdekében. A számítógépet üzemeltető szervezetek körében végzett felmérés (munkáltatói adatgyűjtés) egyik célja az volt, hogy adatokat gyűjtsünk e szervezetek munkaerő-fejlesztési szándékairól. A felmérésben résztvevő szervezetek összesen 732 rendszerszervezőt, folyamatszervezőt, programtervezőt, számítógép-programozót és gépkezelőt foglalkoztattak 1977-ben és 1980-ra 1128 főre kívánták emelni a szakemberlétszámot. Ez 54%-os emelést jelent, vagyis 1977-es bázishoz viszonyítva évi 18%-os létszámnövekedést terveztek. Átlagon felüli növekedést terveztek a folyamatszervezők és alkalmazási programtervezők esetében.

Hasonló eredményt adott az ugyancsak 1977-ben végzett másik felmérésünk a számítógépet beszerezni szándékozók között.

A válaszoló szervezetek 70%-a a gépbeszerzéssel gépparkját bővítette, ill, régi gépét cserélte és csupán 30% volt új beszerző. Ezért a szakemberigény-növekedés értékelhető volt. A rendszerszervező, folyamatszervező, programtervező, számítógép-programozó és gépkezelő szakmák esetében az éves igénynövekedés 20% volt a felmérés szerint.

A felmérések jellege miatt a szakemberigényt nem tudtuk pontosan megismerni, mégis az adatok kvalitatíve jól értelmezhetők. Az átlagos szakemberigény évente 18–20%-kal növekszik (bár ezekben az adatokban sok a „vágy" is), ami alapján annyit megállapíthatunk, hogy a szakember kibocsátás volumenét legalábbis ne csökkentsük. Egyébként a felmérésekből származó adatok is hozzájárultak, hogy az alkalmazási programtervező képzést elindítsuk.

A tanfolyamainkra történő jelentkezések adatait feltételezéseinket igazolták.

B) A végzettek által betöltött munkakörök megoszlásának vizsgálata

Az előbbinél – bárcsak relatív értelemben – lényegesen pontosabb, de közvetett igényfelmérésre adott lehetőséget a SZÁMOK szakemberképző tanfolyamain végzettek munkaköri megoszlásának vizsgálata a végzést követő 1 év múltán. Mintavételes felmérésünk alapján az egyes szakemberképző tanfolyamot végzettek az alábbi százalékos megoszlást mutatják a betöltött munkakört illetően:

Szakma	Saját	Egyéb számítás	Más	Összes résztvevő
	szakmában dolgozók %			
Rendszerszervező	35,2	18,5	46,3	100
Folyamatszervező	19,1	26,6	54,3	100
Programtervező	60,0	20,0	20,0	100
Szgép-programozó	60,0	18,6	21,4	100
Szgépkezelő	57,4	32,9	9,7	100
Összesen:	40,7	23,9	35,4	100

1. táblázat

Az ideális helyzet az lenne, hogy a végzettek 100%-a a saját szakmájában dolgozik a végzést követően. Felméréseink tanúsága szerint csupán a végzettek 40%-a dolgozik a saját szakmájában, további 24%-a pedig szakmájához közelálló (esetleg magasabb szintű) munkakörben. A maradék 36% más szakmában dolgozik. Ez az utóbbi érték azt mutatja, hogy „alulképzés"-ről nem lehet beszélni. Meg kell jegyezni azonban, hogy ez az érték nullára sosem csökkenthető, mivel előfordul, hogy a munkaerő és a munkáltatók földrajzi megoszlása eltérő vagy szubjektív tényezők zavaják a szakmában történő elhelyezkedést.

C) A fluktuáció vizsgálata

Egy szakmában dolgozók körében általában akkor nagy a fluktuáció, ha vagy alacsony az igény a szakma iránt (a szakemberek keresik a helyüket) vagy túl nagy (a szakembereket előnyös

ajánlatokkal csábítják a vállalatok). Az alacsony fluktuáció bizonyos egyensúlyról tanúskodik. Felméréseink szerint azok közül a szakemberek közül, akik már a tanfolyam megkezdése előtt is végzettségüknek megfelelő szakmában dolgoztak, csupán 15,4% változtatott munkahelyet. Az „új szakemberek"-nek, akik a tanfolyam elvégzése előtt más munkakörben dolgoztak, viszont 30%-a változtatott munkahelyet, feltételezhetően az új szakképzettségnek megfelelő munkakör betöltése érdekében. Ez azonban nem bontja meg az egyensúlyt. Eszerint a felmérés szerint tehát az igények és a képzés kibocsátási volumene között egyensúly áll fenn.

Összességében oktatási rendszerünk képzési kapacitása jónak tekinthető, mivel a mérések azt mutatják, hogy a mennyiségi igényeket megközelítően kielégítjük. Bár az egyes vizsgálati módok bizonyos mértékben ellentmondanak egymásnak és pontosságuk sem kielégítő még.

2.3. *A szakemberképzés minőségének elemzése*

Mint már a bevezetőben megfogalmaztuk, egy oktatási rendszernek fontos minőségi követelményeknek is eleget kell tenni. Olyan szakembereket kell kibocsátania, akik a munkakörüket minél teljesebb mértékben önállóan és lehetőleg rövid idő eltelte után el tudják látni. Magát a minőségi követelményt is nehéz egzaktul megfogalmazni, még nehezebb problémát jelent ennek vizsgálata. A mi felfogásunk szerint a szakemberképzés minőségét a szakemberek

- munkafeladat ellátási képesség (beválási mutató),
- önálló feladatmegoldó készség,
- termelékenységének felfutási időtartama (önálló munkavégzés egy adott környezetben)

határozzák meg. Ennek elemzése azonban meglehetősen nehéz és nehezíti a problémát az, hogy eléggé eltérő képességű szakemberek „minőségéből" kell következtetéseket levonnunk az oktatásunkra.

A vizsgálatok típusai

A) Az elkészült alkalmazási rendszerek minőségének vizsgálata

Egy ilyen vizsgálat szolgáltatná a legmegbízhatóbb eredményeket, ha egzaktul megoldható lenne. Azonban mint oktatási szervezet erre a feladatra nem vállalkozhatunk, részben mivel az alkalmazási rendszereket nemcsak a SZÁMOK-ban végzett szakemberek készítik. Erről a lehetőségről tehát le kell mondanunk.

B) A munkafeladat ellátási képességek vizsgálata

Ezt a viszonylag egyszerűnek látszó feladatot eddig nem sikerült megoldanunk. Egyelőre csupán volt hallgatóink körében próbálkoztunk a beválási mutató meghatározásával, hogy ma hazánkban még nincsenek általánosan elfogadott, megalapozott munkafeladatleírások és a szakemberek részére eléggé idegen ez a fogalom. Ez évben egy újabb módszerrel próbálkozunk, amitől eredményeket várunk.

C) Az önálló feladatmegoldó készség vizsgálata

A szakmai vezetők körében végzett felmérés során arra igyekeztünk adatokat kapni, hogy a SZÁMOK-ban végzett szakemberek közvetlenül a végzés után képesek-e önálló feladatmegol-

dásra. Az „igen" válaszok aránya 21% volt, ami oktatásunk gyakorlatiasságának hiányaira mutat rá, bár nem tudjuk milyen arány lenne célszerűbb. Véleményünk szerint az önálló feladatmegoldó készséget egy kb. féléves akklimalizálódás után lehetne reálisan meghatározni. Ezt a vizsgálatot azonban nehezen tudnánk megvalósítani.

D) **A termelékenység felfutási időtartamának vizsgálata**

Egy frissen végzett szakember általában lényegesen kevesebb önálló produktumot tud egységnyi idő alatt előállítani, mint egy több éve gyakorló szakember. Az elfogadható termelékenységi szint elérése függ az egyén képességeitől, de szorosan összefügg az oktatási rendszerben elsajátított ismeretektől és azok begyakorlottsági fokától. Így tehát a felfutási idő hosszából vissza lehet következtetni a képzés minőségére, elsősorban gyakorlatias jellegére. Minél több hiányzó ismeretet kell pótolni a végzett szakembernek, annál lassúbb a felfutás. A magasszínvonalú, gyakorlatias oktatás viszont lerövidíti a felfutási időt.

Tulajdonképpen nehéz egyértelműen meghatározni, mit nevezünk elfogadható termelékenységi szintnek. A mi megfogalmazásunk szerint az az állapot, amikor a kezdő szakember önállóan, legfeljebb eseti szakmai segítségnyújtás mellett tudja napi feladatait ellátni és a teljesített produktum sem minőségében, sem mennyiségében nem lényegesen kevesebb a begyakorlott szakemberek teljesítményénél.

A munkáltatók körében végzett kérdőíves felmérés során a kívánt termelékenységi szint eléréséhez szükséges időtartamot kérdeztük meg. Az eredmény szerint átlagosan 13,5 hónap szükséges a kívánt termelékenységi szint eléréséhez. A kapott eredmény ugyan elfogadhatónak látszik, de meglehetősen nagy szórást (0,5—3 év) mutat. Ennek egyik valószínű oka a fogalom nem azonos értelmezése lehet.

Mindenesetre a produktivitás lassú emelkedése arra figyelmeztet, hogy hallgatóinkat nem teljes mértékben készítettük fel a mindennapok feladataira. Különösen elgondolkoztató ez a gépkezelők esetében, akiknél az átlagnál jobb, 8 hónapos felfutás is nehezen fogadható el.

E) **Az oktatott ismeretanyag megfelelésének vizsgálata**

Mivel a munkafeladat ellátási képesség meghatározását nem sikerült megoldani, egy közvezett vizsgálati módszert alakítottunk ki. Eszerint az oktatott ismeretek megfelelési szintjét igyekeztünk meghatározni. Ez az oktatási rendszer működésének minőségi paraméteréről közvetlen, de az igények kielégítéséről márcsak közvetett információt ad. Ez utóbbi szempont miatt ez a módszer kevésbé pontos eredményt ad, mint a munkafeladat ellátási képesség vizsgálata.

Az oktatott ismerethalmaz lényegében kétféle hibát hordozhat:

- felesleges ismereteket is tartalmaz,
- hiányos.

Mindkét hiba jelenléte rontja a minőséget. Az első esetben a felesleges ismeretek helyett vagy hiányzókat lehet oktatni, vagy — ha ilyen nincs —, akkor a szükséges ismeretek oktatását lehet mélyíteni.

Végső esetben a tanfolyam hossza is csökkenthető. A hiányos tartalom eredménye az, hogy az oktatási rendszer nem megfelelően készíti fel a hallgatókat a napi munka ellátására. Ezért ezeket az ismereteket pótolni szükséges.

Az ismeretanyag megfelelésének vizsgálatára két felmérést alkalmaztunk: a szakmai vezetők és a volt hallgatók körében végzett felmérést. A felesleges ismereteket csak a volt hallgatók tudják megnevezni, míg a hiányzó ismereteket rajtuk kívül a közvetlen vezetők is fel tudják sorolni. A hiányzó ismerethalmaz meghatározásához ez a két felmérés jól kiegészíti egymást.

Ezek az eredmények is tartalmaznak szubjektív elemeket, hiszen adott egyedi helyzetben, konkrét személyekről szólnak a vélemények, ill, konkrét személyek saját környezetükről alkotott véleményüket is belevetítik a véleményükbe.

Statisztikai módszerekkel azonban a szignifikáns eltérések jól kimutathatók.

Természetesen, nemcsak végleges információkat kaptunk, hanem le tudtuk mérni azt is, hogy mit oktatunk hiányos, vagy nem megfelelő tartalommal. A tartalomra vonatkozó felmérésenk a legsikeresebbek és ezek hozták a legtöbb eredményt is.

Összefoglalva megállapíthatjuk, hogy az oktatási rendszerünk által kibocsátott szakemberek „minőségének" vizsgálatát több módszerrel kíséreltük megoldani. A munkafeladat ellátási képességet még nem tudjuk vizsgálni. Az önálló feladatmegoldó-készség és a termelékenységi szint felfutási idejének meghatározásában kezdeti eredményeket értünk el. A legeredményesebben a képzés tartalmát tudtuk elemezni.

3. A vizsgálatok eredményeinek összefoglalása, felhasználása

Oktatási rendszerünk működése eredményességének vizsgálatával a rendszer legfontosabb paramétereit elemeztük. A vizsgálatok lehetőséget adtak arra, hogy szakemberképzési rendszerünket átfogóan értékeljük és módosítást megtervezzük. Az eredmények alapján a szakemberképző rendszer szerkezetét megváltoztattuk és három új tanfolyammal (alkalmazási programtervező, rendszerprogramozó, üzemeltetésvezető) bővítettük és a programtervező tanfolyamot – kellő igény hiányában – megszüntettük valamint a kétszintes képzést megerősítettük.

Oktatási rendszerünk képzési kapacitása jól közelíti a környezeti igényeket, így a képzési volumen növelésére nincs szükség.

A képzés minőségi paramétereit csak korlátozott mértékben tudtuk meghatározni, mivel a munkafeladat ellátási képesség vizsgálatát még nem sikerült megoldani.

Az önálló munkavégzésre és a termelékenységi szint elérési idejére vonatkozó felmérések alapján megállapítható, hogy a gyakorlati ismeretet a tanfolyamok tartalmában növelni szükséges. A tanfolyamok tartalmával kapcsolatos felmérések sok tartalmi eltérést mutattak ki, amelyeket az új szakemberképző tanfolyami rendszer tanfolyamaiban korrigálnunk kell.

Meg kell jegyeznünk, hogy az oktatási rendszerben – csakúgy, mint egy termelő rendszerben – a kibocsátott létszám és a szakemberek minősége között szoros összefüggés van. Adott kapacitási és szervezési szint mellett a mennyiség növelése csak a minőség rovására oldható meg, míg a minőség emelése igényli a mennyiség csökkentését. Az együttes emelkedés azonban hatásos tematikákkal, az oktatási kapacitások jobb kihasználásával, segédeszközök alkalmazásával stb. elérhető. Célunk továbbra is az, hogy szakemberképző rendszerünk eredményességét növeljük. Ehhez megbízhatóbb módszereket igyekszünk kidolgozni, hogy a környezet valós igényei ismeretében tudjuk elvégezni a fejlesztést.

Irodalom:

[1] Számítástechnikai szakemberek munkafeladatai SZÁMOK Oktatástervezési Osztály. Belső kiadvány Budapest, 1977.

[2] KSH OSZI: Központi Statisztikai Hivatal Számítástechnikai Adatgyűjtési Rendszere 1978. KSH OSZI, Budapest, 1977.

[3] A munkáltatók véleménye szakemberképzésünkről SZÁMOK. Oktatástervezési Osztály. Belső kiadvány Budapest, 1977.

[4] Dr. Kocsis András: A szakemberképző tanfolyamok volt hallgatói véleményének értékelése SZÁMOK. Budapest, 1978.

[5] A gépbeszerzések (gépcserék és új installációk) szakemberigénye SZÁMOK Oktatástervezési Osztály. Belső kiadvány Budapest, 1977.

A KSH–SZÜV HELYZETE ÉS SZOLGÁLTATÁSAINAK FEJLESZTÉSE

Dr. Kondricz József
KSH SZÜV

A. A KSH–SZÜV bemutatása

A KSH–SZÜV jelenleg Magyarország legnagyobb számítástechnikai vállalata. Reprezentálja a számítástechnikai alkalmazások teljes keresztmetszetét, szolgáltatásai kiterjednek:

- az államigazgatási szervezetre,
- a népgazdaság valamennyi termelő „szolgáltató" ágazatára

A vállalat kiemelt jelentőségű feladata az SZKFP-ben (Számítástechnikai Központi Fejlesztési Program) megfogalmazott regionális hálózat kialakítása.

Budapesten és 13 megyeszékhelyen működik számítóközpont, összesen 28 db közepes teljesítményű számítógéppel, mintegy 500 db lyukkártyás adatrögzítő géppel, valamint 2 mágnesszalagos adatrögzítő rendszerrel.

A vállalathoz tartozik – a népgazdasági igények mintegy 50%-át kielégítő – számítástechnikai nyomtatványok, papíralapú adathordozókat előállító nyomdaüzem is.

A vállalat létszáma kb. 3200 fő. A létszámon belül lényeges kiemelni, a számítástechnikai szolgáltatásban meghatározó szerepet betöltő néhány munkakör létszámarányát. Pl.:

- szervezők-programozók létszáma az összlétszám kb. 17%-a fele-fele arányban,
- termelési területen dolgozók (üzemeltetés vezetők, operátorok, gépkezelők, adatrögzítők) a vállalati összlétszám kb. 50%-át teszik ki,
- számítógép karbantartók létszáma az összlétszám kb. 7%-a.

A KSH–SZÜV tevékenységi köre számítástechnikai és ügyvitelszervezési szolgáltatás, ezen belül:

- a központi állami statisztikai adatfeldolgozási feladatai és a kiemelt, országos jelentőségű adatfeldolgozások ellátása,
- az országos adatfeldolgozási hálózat megszervezése, üzemeltetése, és fejlesztése,
- számítógépre szervezés,
- adatfeldolgozási szolgáltatások nyújtása gépi eszközökkel,
- számítástechnikai szolgáltatás műszaki-tudományos célokra, pl. lineáris programozás, optimalizálási feladatok, hálós programok készítése, matematikai szubrutingyűjtemény, stb.,
- ügyvitelszervezési feladatok megoldása,
- ügyvitelgépesítéshez használt nyomtatványok és papíralapú adathordozók előállítása és forgalmazása.

A vállalat fejlődési üteme az elmúlt 30 év alatt többszörösen meghaladta a népgazdaság ágazatainak fejlődési dinamikáját és e tekintetben megfelelt a szakmában mutatkozó világjelenségnek is.

A KSH–SZÜV jelenleg közel 60 témában, országos szinten 900 megrendelőnek biztosít bérmunkakonstrukcióban *szervezési, programozási, adatrögzítési és gépi feldolgozási,* továbbá nyomdai szolgáltatást, s mindez egyre növekvő hányadában jelenti

– a központi és helyi államigazgatási szervek,
– az országos nagyvállalatok, trösztök ágazati szintű tipizált feldolgozási rendszerének létrehozását.

I. A szolgáltatások felhasználói területek közötti megoszlása

A vállalat számítástechnikai tevékenységének megoszlása az alkalmazások irányának szempontjából a következő:

1. Minisztériumok és országos hatáskörű szervek részére végzett feldolgozások között kiemelt jelentőségűek

– *A Központi Statisztikai Hivatal számára* mintegy 11 témában végzett feldolgozások (pl. Mezőgazdasági statisztika, Egységes Munkaügyi statisztika). A rendszerek tervezése és géprevitele a kijelölt témafelelős számítóközpont által történik, a futtatás a téma jellegétől függően centralizált ill. decentralizált formában valósul meg. Az eredményközlő táblák közvetlenül a megyei igazgatóságokhoz, az összesített adatokat tartalmazó mágnesszalagok a KSH–SZIG-hez kerülnek továbbfeldolgozásra. (Központi Statisztikai Hivatal Számítástechnikai Igazgatóság.)

– *Minisztériumok részére végzett* feldolgozások általában lokális jellegűek, az információk továbbfeldolgozása általában a minisztériumok számítóközpontjaiban történik.

2. Népgazdasági szintű alapnyilvántartáshoz kapcsolódó feladatok:

– *az Állami Népességnyilvántartási* rendszer input alrendszere a tranzakciós tételek rögzítése, logikai ellenőrzése, az adatbázis kreálását követően a hibajavítás elvégzése, stb. A rendszer olymódon került kialakításra, hogy a személyi adatbázis decentralizált működtetése esetén a SZÜV hálózat a helyi államigazgatási szervekkel on-line kapcsolatot tudjon létesíteni (megfelelő hardware ellátás esetén). Adott esetekben a decentralizált adatbázis szegmensek karbantartásával a közvetlen kapcsolaton kívül kötegelt feldolgozásokkal kerülnének kielégítésre a helyi és központi államigzagatási és hatalmi szervek információs igényei;

– *a KISZ KB részére* végzett személyi nyilvántartási rendszer, amely adott eszmei időtartamot tekintve teljeskörű információt szolgáltat a tagokról, a helyi és a központi szervezetekről.

– *az 1980-as évi népszámlálás* adatrögzítése és teljességellenőrző rendszere, amely a területi államigazgatási szervek adatait decentralizáltan dolgozza fel, és az országos szintű centralizált feldolgozáshoz megfelelő adatokat biztosít.

3. Helyi államigazgatási szervek részére végzett feldolgozások esetén a vállalat számítóközpontjai egyrészt – a helyi igények kielégítésére – egyedi rendszereket, másrészt típusrendszereket (pl. tanácsok komplex költségvetési rendszere) dolgoznak ki.

4. Vállalatok, intézmények, trösztök részére végzett feldolgozások a számítástechnikai szolgáltatások jelentős részét képviselik. A feldolgozások egyre növekvő hányadát teszi ki az ágazati szintű szervek információrendszerének kialakítása, és üzemeltetése.

II. Az irányítási rendszer

Az előzőekben vázolt feladatok megoldása *(funkcionális vállalatirányítással* és a technológiai folyamatnak megfelelően szakaszolt *egységes számítóközponti szervezetben* valósul meg.

A vállalat funkcionális irányítása – igazgatóhelyettesi szinten – az alábbiak szerint oszlik meg: szervezés-programozás, termelés, műszaki, gazdasági, valamint a regionális hálózat kiépítését végző hálózatfejlesztés. A vállalat igazgatójához az említett funkcionális igazgatóhelyettesek mellett, még a személyzeti és szociális főosztály, valamint a területi számítóközpontok tartoznak.

A területi számítóközpontok felépítésére jellemző, hogy az SZK igazgatójához osztályszinten tartoznak a szervezési-programozási, termelési, műszaki és gazdasági szakterületek.

III. A szolgáltatások technikai eszközei

A szolgáltatások döntően R-20 – R-22-es típusú számítógépeken történik. Az azonos konfigurációjú gépeken azonos operációs rendszert alkalmazunk: R-20-as gépeknél DOST, R-22-nél DOS és OS-t. Programnyelvként döntően a PI/1-et használjuk, és ahol erre lehetőség van, ott alkalmazzuk a MARK–IV-et és a CFMS láncolásos filekezelő rendszert.

IV. Előkészületek a szolgáltatások közüzemi jellegének kialakítására

A KSH–SZÜV helyzete, adottságai és az utóbbi években egyre fokozottabban jelentkező igények következtében az előzőekben ismertetett ágazati típusrendszerek kialakításán, országos hatáskörű szervezetek információs igényeinek kielégítésén, számítógépes cégek által gyártott programcsomagok alkalmazásán túl, kísérleti jelleggel megvalósult a Mosonmagyaróvári Mezőgazdasági Gépgyár R-10-es és a Győri Számítóközpont R-22-es közötti Távadatfeldolgozás.

Folyamatban van a Budapesti és Székesfehérvári Számítóközpontokban működő csoportos adatrögzítő rendszerek (MDS, VIDEOPLEX–3) fejlesztése olymódon, hogy az

– adatrögzítési, ellenőrzési és hibajavítási;
– előfeldolgozási és input-output feladatvégzési;
– valamint RJE funkciók megvalósuljanak.

A közeljövőben megvalósításra kerül néhány *országos hatáskörű szervezet területi egységéhez terminálok telepítése* (pl. OTP), ilymódon a jelenlegi technológia által megszabott határokat kitágítva a számítógép közelebb kerül a felhasználóhoz, az *időszakos feldolgozások helyett* a döntési folyamatok átfutási idejének csökkentésével *hatékonyabbá válik a számítógép alkalmazása.*

B. A KSH–SZÜV szolgáltatásainak várható fejlődése

A VI. ötéves terv során a vállalat fejlődése néhány kiemelt területen a következőképpen alakul.

I. Vállalatfejlesztés

Az *extenzív növekedés* során további négy megyeszékhelyen alapítunk számítóközpontot és ezzel a már korábban tipizált méretű, funkciójú központok létrehozását befejezzük. E központokba mágneses adatrögzítést és az ESZR II sorozat gépeinek installálását tervezzük.

A *számítóközponti szervezet* alapvetően nem változik. Néhány új körülménnyel számolunk:

– a szakmai létszám alakulására két ellentétes hatás hat: egyrészt nő az igény a szellemi tervező (kivitelező létszám növelésére mennyiségi/minőségi), másrészt az ügyfelek egy része maga képes lesz feladatait számítógépre vinni;

– ki kell alakítani rendszertechnikai szervezetet, amely adatátviteli, operációs rendszer hatékonysági feladatokkal foglalkozik.

A központi szakmai irányítás területén kialakulnak az irányított tevékenységek módszertanával foglalkozó csoportok, (szervezés-módszertan, programozási-, géptermi üzemszervezés, stb.) és kialakul a számítógéphálózat tervezésével foglalkozó szervezet. Előtérbe kerülnek a tipizálás további területei:

– géptermi üzemek,
– előfizetők kiszolgálása, elszámolása;
– típus rendszertervezési, gépreviteli módszerek,
– a teljes technológiai folyamat dokumentációinak szabványosítása;
– alkalmazói típusrendszerek.

Belföldi kooperáció növekedése során egyre több feladatot kap a SZÁMKI. Az ágazati szervező intézetektől újabb típusrendszereket veszünk át terjesztésre és üzemeltetésre. (SZÁMKI: Számítógépalkalmazási Kutató Intézet)

Külföldi kooperációban minden szocialista ország testvérvállalataival bővülnek kapcsolataink (ZETO, KESSZI, VVB MR, stb.).

Ugyfélkörünk (előfizetőink) összetételében a következő változások várhatóak:

– nő a mezőgazdasági (állami gazdaság, mezőgazdasági szövetkezet) ágazat, és a szövetkezeti szektor, valamint az igazgatási-, tanácsi alkalmazások száma;

– a kisebb gazdasági egységek is igénybe veszik a szolgáltatást kötegelt módon, a régi partnerek egy része terminált telepít;

– a nagyobb ügyfelek (1000 fő felett) saját számítástechnikai szervezetet hoznak létre és a bérmunka igénybevétele csökken (az irányítási és információ rendszerének kötegelt működtetéséről áttért az interaktív üzemre).

II. Szolgáltatásfejlesztés

Az alkalmazói rendszerek minőségének színvonala emelkedik. Nő a vállalati vezetést segítő, az információ rendszereket alrendszeri szinten megvalósító alkalmazások száma (műszaki információs rendszer, tervezési rendszer, készletgazdálkodás, stb.). A számítógépes alrendszerek integráltak lesznek (struktúrált terv, modell-szintű kivitelezés). Az alkalmazói rendszerek lehetővé teszik a távolról történő üzemeltetést (kevés adat mozgatása révén).

Az általunk nyújtott szolgáltatásainkban a technológiai fejlődés a következőket eredményezi:

– komplex terminálokat fogunk – mint előfizetői pontokat – kiszolgálni;
– közcélú adatbázisokhoz adunk hozzáférési lehetőséget (batch és interaktív módon);
– a számítógépes erőforrások bérbeadása (batch és távolsági batch és interaktív módon) pl. oktatás, programfejlesztés);
– általános programcsomagok alkalmazása;
– az automatizált mérnöki munka segítése.

A komplex terminálok funkciói:

– adatrögzítés, ellenőrzés, adatjavítás;
– helyi előfeldolgozási és lassú input, output műveletek;
– intelligens terminál (RJE).

Előnyei:

- a megrendelőhöz (előfizetőhöz) visszük a számítástechnikát;
- helyben rögzít (esetleg az adat keletkezési helyén), javít;
- mentesíti a számítóközpontot az adatrögzítés és a nyomtatás feladataitól;
- a terminálról hozzáfér a nagygépen lévő adataihoz, előfizetői JOB-ot indíthat;
- csökken a feldolgozás szezonális jellege.

A komplex terminál típus konfiguráció

- 64 Kb tár
- 4-8 adatrögzítő munkahely
- 1-3 mágnesszalag
- kártyaolvasó
- nyomtató
- szinkron átviteli csatoló
- típus (EC 1010–11–12)

A komplex terminál telepítésére az esetek nagyobb részében 50 km körzeten belül kerül sor, ezért tervezzük alapsávi csatlakozó használatát 9600 bit/sec. sebességig. Más adatátvitelnél a postai vonal max. 2400 bit-xec-ot tesz lehetővé.

Közcélú adatbázisok közül kiemeljük:

- katalógusok, termékjegyzékek, címjegyzékek;
- ágazati tervezési, normák (pl. építőipar, költség norma);
- területi adatbázisok (társadalmi szervek, népesség, ingatlan, statisztika, stb.).

Számítógépes erőforrás bérbeadása batch, távolsági batch és időosztásos módon. Felhasználók: oktatási intézetek, gazdálkodó szervek.

Általános programcsomagok közül kiemeljük az ágazati típusrendszereket, az ismert rendszerek (PMS, PROJACS, PICS, stb.) alkalmazási lehetősége, amely az SZK könyvtárában megtalálható, használatukhoz az SZK dokumentációt, programot, mintaanyagot, tanácsadást ad.

Automatizált mérnöki rendszerek alkalmazása tervező, kivitelező és oktató intézmények részére. Kezdetben batch, majd TAF útján (pl. GPSS, SSP, ICES).

III. A szolgáltatás technikai eszközei:

A legfontosabb hw elemek:

- az ESZR II. sorozat gépei (1 Mbyte tár, 30/100 Mbyte-os lemezegységek, adatátviteli vezérlő, TAF rendszer elemek);
- COM rendszer a legnagyobb ügyfelek kiszolgálására központi telepítésre (OTP részére);

Sw elemek:

- OS és HASP általános használata;
- szakosított, hierarchikus könyvtárak;
- tipizált terminál (átviteli) algoritmusok;
- file-kezelő rendszerek, tömeges alkalmazásuk elterjedése (MARK IV, CFMS);
- az OS TSO változatának bevezetése;
- számítógéppel támogatott rendszertervezés (PROTEE, ISDOS).

IV. A szolgáltatások közüzemi jellegének erősödése

Nő az ágazati típusrendszerek száma, az országos kiterjedésű gazdálkodó szervek (trösztök, nagyvállalatok) információ igényét tipizált rendszerekkel elégítjük ki.

A ma még centralizált az országos alapnyilvántartásokhoz kapcsolódó feldolgozások bizonyos részeit decentralizáljuk (népességnyilvántartás, népszámlálás, ingatlan, KISZ, KSH-statisztika, stb.).

Az előfizetők további igényei:

- módszertani, dokumentációs szolgáltatás, tanácsadás,
- oktatás,
- háttérkapacitás biztosítása.

A számítástechnikai kultúra terjesztésének egyéb formái:

- a társadalmi szervekben képviselet (Neumann János Számítógéptudományi Társaság);
- számítástechnikai ismeretterjesztés;
- megyei SZAB-ok munkájában részvétel;
- kiállítások, bemutatók szervezése.

V. Számítógéphálózat és a KSH–SZÜV

A számítógéphálózat komplex (fizikai) kiépítése jórészt függetleníti a felhasználói feladat keletkezését a megoldás helyétől.Ennek kiépítése több fokozaton át, a kompatibilitás fenntartásával és megőrzésével történik.

Álláspontunk szerint a megyeszékhelyeket hatékony és a Posta kezelésében lévő adatátviteli hálózatnak kell összefognia. Erre kapcsolódnak – másokkal együtt – a SZÜV számítóközpontjai. A megvalósításban fontos szerepe lesz a kisszámítógépeknek.

AZ RA RAJZDIGITALIZÁLÓ BERENDEZÉSEK ÉS ALKALMAZÁSAIK

Krizsanits János
FOK–GYEM

Bevezetés:

Az információ közlés egyik legrégibb és legelterjedtebb módja a rajz, az ábra. Különösen nagy jelentősége van a reáltudományok terén, a műszaki tudományokban. Gondoljunk az építészetre, a gépészetre, amelyeknek legfontosabb leírási eszköze a rajz. De érvényes ez sok más területre is például kartográfia, várostervezés, geodézia, orvostudomány, textilipar stb.

A technika fejlődésével egyre elterjedtebben alkalmaznak olyan automatikus adatrögzítőket, amelyek képi formában közölnek információt, pl. az oszcillográf, az EKG, a röntgen vagy a földmérésben alkalmazott légi felvételek. Ezek a berendezések önállóan dolgozhatnak, így igen sok anyag gyűlhet össze kiértékelésre várva. Ez kétféle problémát vet fel: az egyik a sok ábra feldolgozása, a másik az emberi kiértékelésből adódó szubjektívitás, ami pl. az EKG felvétel esetén kritikus lehet.

A számítástechnika fejlődése, a digitális számítógépek egyre nagyobb mérvű elterjedése már korán felvetette az igényt, hogy jó lenne az ábrás információkat számítógép számára érthető formátummá konvertálni, megoldva ezzel a nagy tömegű információ gyors feldolgozását, és kiküszöbölve az emberi tévedés lehetőségét.

Ezt a feladatot oldja meg a rajzdigitalizáló, amely valamilyen rajzot képes számjegyes formátummá konvertélni, majd ezt valamilyen off-line adatrögzítőre vinni, esetleg on-line módon közvetlenül számítógépbe továbbítani.

A világpiacon a digitalizálóknak igen sokféle típusa van forgalomban. A gyártó cégek közül csak a legnagyobbak neveit soroljuk fel: Aristo, Summagraphic, HP, Ferranti. Az ő általuk gyártott berendezések bemutatása nem célja ennek az előadásnak, csak összehasonlítási alapul említünk egyet, amely hasonló az általunk gyártotthoz. A HP 9864A típusa 431,8 x 431,8 mm táblamérettel rendelkezik, relatív rendszerben dolgozik, folyamatos vagy lépésenkénti üzemmódjai vannak, felbontóképessége 0,254 mm. Rendelkezik Hold üzemmóddal, amely a táblaméretet meghaladó ábrák digitalizálásához szükséges. A digitalizáló a HP cég kalkulátoraihoz kapcsolható.

Hazánkban először a FOK–GYEM által gyártott RA berendezések kerültek kereskedelmi forgalomba.

1. A berendezés fő részei:

– *A preparált rajztábla.* Erre helyezzük a digitalizálni kívánt rajzot. A preparálást a táblát belülről borító drótháló jelenti.

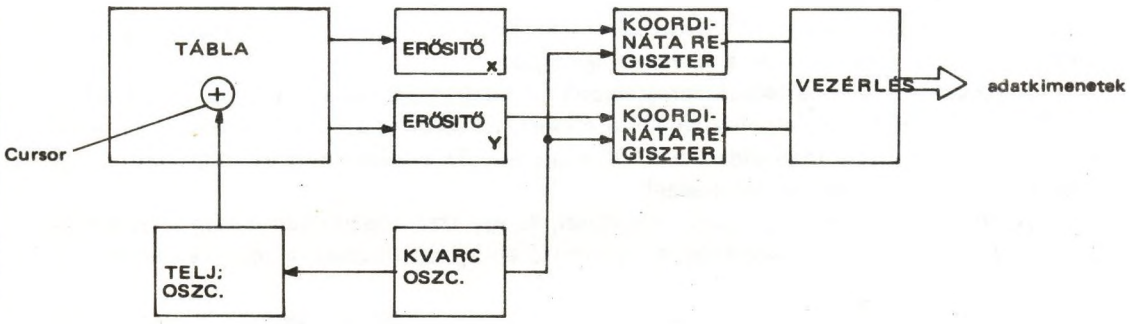
– *A cursor.* Ez tulajdonképpen egy speciális tekercs, amelyet váltakozó árammal gerjesztve feszültséget indukál a hálóban. A cursor középpontjában elhelyezett fonálkereszttel kell lekövetni a digitalizálni kívánt rajzolatot. Ugyancsak a cursoron helyezkedik el néhány kezelőszerv, amely funkcióinak ismertetése később következik.

–*Vezérlőelektronika.* Érzékeli és feldolgozza a táblából érkező jeleket, megfelelő digitális formátummá alakítja, és elvégzi a kommunikációt a külvilág, pl. egy adatrögzítő, vagy egy számítógép felé.

— *Kezelőegység.* Ezen találhatók az üzemállapot beállítására szolgáló nyomógombok, a cursor pillanatnyi helyzetét jelző kijelzők, és néhány egyéb funkciót szolgáló gomb és jelzőlámpa.

2. A digitalizáló működése

A berendezés alapja a rajztáblában elhelyezkedő raszterháló. A cursorban lévő tekercsbe váltakozó áramot vezetve az gerjeszteni fogja a hálót, és feszültséget indukál benne. A feszültséget a közvetlenül a háló mellett lévő érzékelő erősítők felerősítik, és továbbítják a vezérlő és feldolgozó elektronikának, amely előállítja a cursor pozíciójának koordinátáit előjelhelyesen, az előzőleg definiált origóhoz viszonyítva. A berendezés blokkvázlata az 1. ábrán látható.



1. ábra

3. A feldolgozás folyamata és a digitalizáló szolgáltatásai.

A digitalizáló négyféle típusban kerül gyártásra. Ezek táblaméretben és felbontóképességben térnek el egymástól. Kétféle táblaméret és kétféle felbontóképesség van, melyek kombinációja adja a 4 típust. A táblaméretek: 825 x 525-ös az A1 méretű kisebb rajzokhoz, 1125 x 750, az A0 méretű nagyobb rajzokhoz.

A felbontóképesség nyomógomboktól függően az egyik típusnál 1,25 vagy 2,5 mm, a másik típusnál 0,1—2,5 mm-ig 8 lépésben változtatható.

A feldolgozás során a digitalizálni kívánt ábrát a rajzasztalra helyezzük, majd a cursort a rajz origójára téve töröljük a koordináta regisztereket, definiálva ezzel az origót. Működés közben a digitalizáló kiadja a cursor pillanatnyi pozíciójának derékszögű koordinátáit előjelesen, az origóhoz viszonyítva. Eközben tilos felemelni a cursort, akkor a berendezés elveszti a pozíció érzékelését, lévén, hogy relatív rendszerben dolgozik. A cursor felemelkedését hangjelzés és egy lámpa felvillanása jelzi, ekkor újra kell kezdeni a folyamatot.

A digitalizáló folyamatos, vagy lépésenkénti üzemmódban képes dolgozni.

Lépésenkénti üzemmódban egy koordinátapont rögzítését a kezelőnek kell kezdeményeznie a cursorban lévő nyomógomb megnyomásával. Ugyancsak itt jelzi egy lámpa, hogy a kimeneti egység továbbította-e az adatot, azaz kezdődhet-e egy új pont feldolgozása. Ekkor a digitalizálás sebességét a kiviteli periféria szabja meg.

Folyamatos üzemmódban egy pont rögzítését a cursor mozgása során bekövetkező raszterátlépés kezdeményezi. Kezelőgombok segítségével megadható, hogy X, Y vagy mindkét irányú raszterátlépés indítson-e feldolgozó ciklust. Miután egy pont koordinátájának megadásához 12 karakter szükséges (X—Y előjele, és 5 helyiérték pontosság), előfordulhat pl. egy lassú lyukasztó esetében, hogy a cursor továbbmozdul mielőtt megtörténne a kivitel. Az érzékelés azonban ekkor sem áll le. Amíg a befagyott koordinátaregiszterből tart a kivitel, addig a számlálást egy másik regiszter végzi, majd ennek tartalma áttöltődik a koordinátaregiszterbe.

Lehetőség van kapcsolóval beállíthatóan, hogy csak minden negyedik raszterátlépés indítson feldolgozó ciklust, miközben az érzékelés, azaz a felbontóképesség nem változik. Ez a szolgáltatás jól jöhet hosszabb egyeneseket tartalmazó ábrák digitalizálásánál.

Átkötéssel beállítható, hogy a cursor mozgásiránya milyen előjelnek feleljen meg, azaz például a jobbról balra történő mozgatás csökkenő, vagy növekvő számsorozatot eredményezzen. Természetesen ez függőleges irányban is igaz.

Lehetőség van a kiküldött karaktersorozat formátumállítására is. Kétféle lehetőség adott: fix formátum, ebben mindig 12 karakter kerül továbbításra, vagy változó formátum, ekkor az értéktelen nullák nem kerülnek a kimeneti egységbe.

A táblaméreteket meghaladó ábrák, diagrammok digitalizálására is van mód. A berendezés STOP üzemmódjában fel lehet emelni a cursort, az ábrát és a cursort arrébb vinni, majd a cursor visszahelyezése után oldani a STOP állapotot. Ez a tulajdonság jó szolgáltatot tehet hosszú szalagdiagrammok kiértékelésénél.

A berendezéshez tartozó cursor nagyítóval, és egy izzólámpával van ellátva a pontosabb ábrakövetés céljából. Egyenes vonalakat tartalmazó ábrák rajzolásához, a tábla rajzgépére felerősíthető a cursor.

A berendezés BSI* interfacet tartalmaz. ASCII kódtáblázatnak megfelelő kódokat továbbít, átkötéssel változtatható paritásvizsgálattal kiegészítve. Vagy nincs paritásvizsgálat, vagy ha van, akkor lehet páros, vagy páratlan. További opcióként a berendezéshez szállítható interface-k:

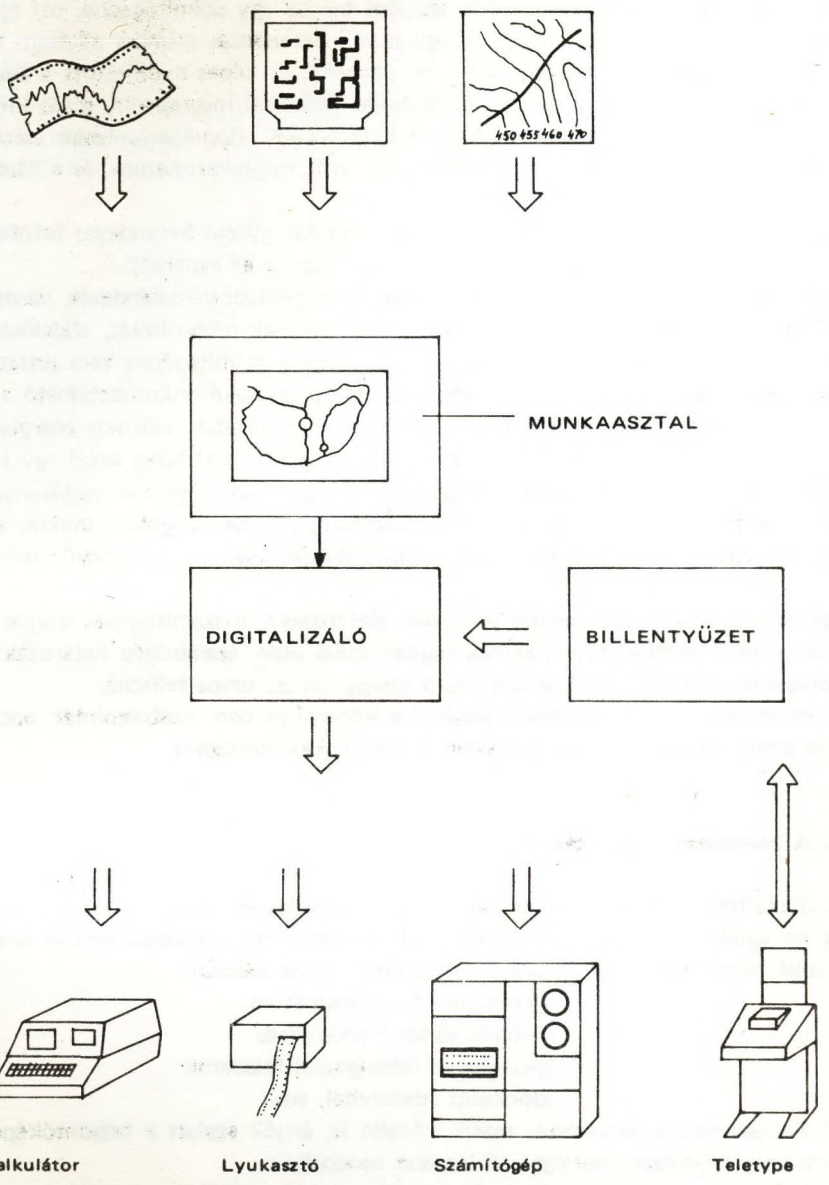
lyukasztó inferface
soros asszinkron interface
TTL szintű BSI interface
EMG 666 interface
billentyűzet interface

Ez utóbbival számítógépes felhasználás esetén, a koordináták közé a számítógép számára vezérlőkódokat, azonosítókat lehet továbbítani.

4. A digitalizáló berendezés felhasználási területei:

Nézzünk meg néhány konkrét példát a digitalizáló felhasználására. Maradjunk először szakmai területen, az elektronikánál. Vizsgáljuk meg egy konfigurációt, amelyben nyomtatott áramkörök gyártásánál alkalmazzák a berendezést. Kövessük végig a folyamatot (2. ábra).

* British Standard Interface


450 455 460 470
MUNKAASZTAL
DIGITALIZÁLÓ
BILLENTYŰZET
Kalkulátor
Lyukasztó
Számítógép
Teletype

2. ábra

Megtervezik a nyomtatott áramkört és kézzel nyers formában megrajzolják pauszra, ahogyan az a tervezés első lépésében történik. A következő állomás, hogy egy digitalizáló, billentyűzet lyukasztó konfiguráció alkalmazásával a szabadkézi vázlatról egy számítógép számára érthető formátumlyukszalagot készítenek. Ezt a szalagot bevive egy számítógépbe, ott egy formátumkonvertáló, illetve optimalizáló program egy másik berendezés számára elkészíti a vezérlőszalagot. Ez a berendezés szintén szövetkezetünk terméke, és képes nyomtatott áramkörök előállítására, lyukszalagvezérléssel. Ez a gép maratásálló festékkel megrajzolja, majd kifúrja a nyomtatott áramkört. Maratás után előttünk áll a kész NYÁK. (Lyukgalvanizálás esetén egy technológiai lépéssel több szükséges.) Ilyen módon sok időt megtakaríthatunk és a klisérajzolás során elkövetett hibák is kiküszöbölhetők.

Ugyanazon vezérlőszalag felhasználásával a NYÁK gyártó berendezés fotófejes típusával közvetlenül, a sorozatgyártáshoz szükséges pozitív film is elkészíthető.

Egy másik lehetséges felhasználási terület lehet például városrendezés, városfejlesztés. Vegyük példának egy város, vagy terület csatornahálózatának módosítását, átalakítását, nyilvántartását. Ezen dokumentációk legtöbbje rajz. Ezeknek a számítógépbe való juttatásához használhatnak digitalizálót. A bármikor adódó módosítás könnyen dokumentálható a digitalizáló segítségével. Természetesen mondhattunk volna telefonhálózatot, villamos energiahálózatot, gáz ivóvíz, út, villamoshálózatot,, stb. vagy akár mindet együtt, kialakítva ezzel egy hatalmas topológiai bázist, melynek nyilvántartását a digitalizáló, számítógép együttes megkönnyíti.

Alkalmazható a digitalizáló térképek rajzolásánál, a magasságokra, utakra, erdőkre, vagy bármi egyéb jellegzetes pontra vonatkozó kódok definiálásával, és billentyűzetről való átvitelével.

Igy fontos alkalmazási terület az orvosi elektronika, a számítógépes diagnosztika. Az elektronikus adatrögzítők diagramjainak digitalizálása után, számítógép határozza meg a lehetséges betegséget, sokszor pontosabban mint ahogy azt az orvos tehetné.

A berendezés jól alkalmazható például a könnyűiparban. Szabásminták optimális elhelyezése adott területen, minél kevesebb hulladék keletkezésével.

5. A berendezés fejlesztési irányai:

A digitalizáló fejlesztése, vagy talán inkább a módszer fejlesztése kétirányú lehet:

a) Az egyik igény, magát a berendezést fejleszti. Intelligensebbé teszi a vezérlést, több szolgáltatást beleépíteni. Gondolunk itt olyanokra, mint például:

változtatható kódformátum
kívülről történő lekérdezés
speciális előfeldolgozási feladatok
időnkénti adatátvitel, stb.

b) Fejleszthető a berendezés másik irányba is, értjük ezalatt a felbontóképesség növelését, a pontosság növelését, esetleges táblaméret módosítást.

A fejlesztés vagy inkább a felhasználhatóság fejlesztésének másik fő iránya, a jól alkalmazható konfigurációk, esetleg speciális célra alkalmazható konfigurációk kialakítása.

Szóbajöhet itt a digitalizáló – számítógép – grafikus display – plotter kapcsolat, vagy bármelyik egy adott területen létjogosult kapcsolat kialakítása.

Az eddigiekből is nyilvánvaló, hogy ez a terület nem egy lezárt, kerek egész, hanem állandó mozgásban, fejlődésben lévő folyamat, amelyet a pillanatnyi igények, az újonnan felmerülő problémák, vagy lehetőségek befolyásolhatnak, módosíthatnak.

I–II. kötet ára: 150.– Ft