Tanulmányok 64/1977

MTA Számítástechnikai és Automatizálási Kutató Intézet Budapest

MAGYAR TUDOMÁNYOS AKADÉMIA
SZÁMITÁSTECHNIKAI ÉS AUTOMATIZÁLÁSI KUTATÓ INTÉZETE

A PSL/PSA RENDSZER HASZNÁLATA AZ INFORMÁCIÓS RENDSZEREK TERVEZÉSÉBEN ÉS DOKUMENTÁLÁSÁBAN

Irta:

Győry György
Halász Ferenc
Szilléry Adnrás
Tóth Beatrix

Tanulmányok 64/1977.

A kiadásért felelős:
DR VÁMOS TIBOR

ISBN 963 311 046 7
ISSN 0324-2951

779143 MTA KESZ Sokszorosító. F. v.: Szabó Gyula

TARTALOMJEGYZÉK

ELŐSZÓ

Napjainkban egyre inkább előtérbe kerül az információ kezelésének, feldolgozásának témája. Nagy anyagi és szellemi erőket forditanak számitógépes információfeldolgozó rendszerek létrehozására.

Ezeknek a rendszereknek a tervezése és létrehozása rendkivül bonyolult, sok hibalehetőséget rejtő folyamat. Észszerünek látszik tehát a számitógép segitségét a tervezésnél is igénybe venni. Ilyen meggondolással hozta létre az Information Systems Design and Optimization System project a michigani egyetemen /Ann Arbor/ a PSL/PSA rendszert, amely a logikai rendszertervezés első általános célú számitógépes segédeszköze. Jelen tanulmány témája ez a rendszer.

Az első fejezet a rendszertervezés általános problémáival és módszereivel, a második a PSL nyelvvel, a harmadik a PSA software rendszerrel, az utolsó pedig a rendszernek az MTA SZTAKI CDC 3300-as gépére történt implementálásával foglalkozik.

A tanulmány - annak ellenére, hogy igyekeztünk a felhasználó szempontjait figyelembe venni - nem felhasználói kézikönyv. Célja, hogy segitségével az olvasó általános áttekintést nyerjen a rendszerről.

1. BEVEZETÉS

Általában a szervezetekről

Ebben a részben a rendszertervezés főbb feladatait, gyakoribb módszereit és problémáit kívánjuk meghatározni. Ehhez legelőször is a szervezet és az ahhoz tartozó információs rendszer fogalmát kell megkülönböztetnünk.

A szervezet, szempontunkból különböző erőforrásokból áll, /anyagi, emberi erő, felszerelés/ amely általában jól meghatározott, de kevésbé jól rögzitett szabályzatnak megfelelően müködve valamilyen /általában termelő/ tevékenységet folytat.

A szervezetek általában kisebb egységekre, blokkokra vannak felosztva, amelyek egy-egy részfeladattal foglalkoznak, amelynek a kiinduló állapota és végeredménye mindenképpen adott számukra.

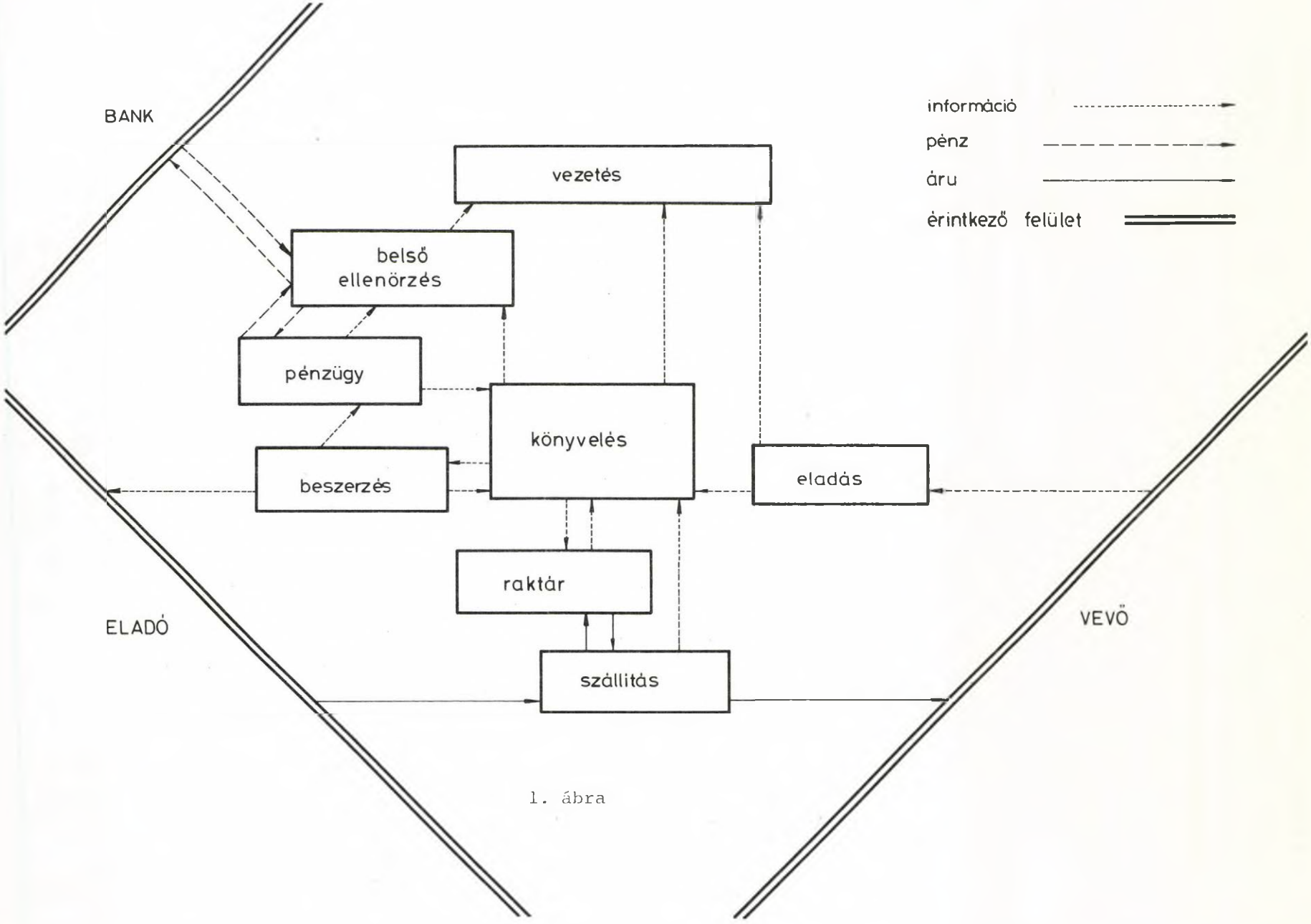
A szervezeteknek ezt a tagoltságát követi az a mód is, ahogy a tevékenységükhöz tartozó információkat kezelik, mivel az információ kisebb egységei leggyakrabban a szervezet egy-egy kisebb egységéhez tartoznak. Ennek megfelelően van előirva az, hogy az információ egyes egységeivel a szervezet mely része, milyen müveleteket végezzen és annak eredményét a szervezet melyik részéhez továbbitsa. Az előirásoknak megfelelő információkezelési módok összességét nevezzük a szervezet információs rendszerének, vagy röviden rendszernek.

Az emlitett előirásokból a rendszer meghatározható, bár az előirások nem mindig irásosak és általában csak a szervezet egységeire lebontva hozzáférhetőek.

A termelésben résztvevő tényezők egy része áramlásban van a szervezeten kivüli környezetből a szervezetbe, a szerve-

zeten belül a szervezeti egységek között, és a szervezetből a külvilágba. Hasonlóan a szervezet által felhasznált és előállitott információ is "folyik" a környezetből, a szervezeti egységek között és a környezetbe. Ezért, hogy a szervezet müködésének leirását adhassuk, ahhoz nemcsak az alegységeket, hanem az erő- és információforrásokat, azok rendeltetési helyeit és áramlatait is ismernünk kell. Erre számos módszer létezik, de általában a blokkdiagramokat tekintik a legáttekinthetőbbnek. A rajz /1. ábra/ egy feltételezett kereskedelmi vállalat leirását adja, háromféle áramlást különböztetve meg:
a "javak", az információ és a pénz "folyását".

Világos, hogy egy rendszer ilyen leirása mindig absztrakción alapul. Például a valóságot jobban közelitő leirást kaphattunk volna a szervezetet kisebb egységekre felosztva, és az áramlásokat is részletesebben ábrázolva. Hogy mi jelenti az ábrázolás finomságának megfelelő szintjét, az mindig a megoldandó feladattól függ, azaz attól a céltól, amire a rendszer leirását használni kivánjuk. A "leirás szintjének" megadása a rendszertervezés egyik fontos feladata.


BANK
információ
pénz
áru
érintkező felület
vezetés
belső ellenörzés
pénzügy
könyvelés
beszerzés
eladás
raktár
szállitás
ELADÓ
VEVÖ

1. ábra

1.1 A szervezet müködésének leirása

A legtöbb szervezet nem rendelkezik saját müködésének ilyen /bármilyen szintü/ leirásával, aminek az az elsődleges oka, hogy /legalábbis eddig/ enélkül is kiválóan müködtek. A müködés jelenlegi módja szerint egyedek illetve csoportjaik egy-egy müveletsort sajátitanak el és ezek elvégzéséből áll elő a szervezet tevékenysége. Számukra elég, ha ezt végzik és nem fontos, hogy az egész rendszer müködését ismerjék. A fentihez hasonló egy-egy blokkséma statikus és elnagyolt, de többnyire kielégitő adatot szolgáltat pl. a kommunikáció csatornáiról. A rendelkezésre álló rendszerleirások elavultak, mivel a szervezet müködését könnyü megváltoztatni, minél kisebb a változtatása, annál inkább. Ezt a rendszer leirása viszont ritkán jelzi. Mivel a leirások kézi erővel készültek, a változtatásokat is igy kell javitani. Ezt nem végzik el, mert feleslegesen munkaigényes tevékenység.

Azóta a szervezetek és környezeteik is komplexebbek és bonyolultabbak lettek, az áramlások lefolyása nehezebben érthető. Hogy az eszközölt változtatások hatásáról még a változtatás előtt meggyőződhessünk, szükségessé vált a szervezetek formálisabb leirása. A leirásnak alkalmasabbnak kell lennie arra is, hogy a szervezet változásait kövesse, hogy abban az egységek, a teljesitmény mértéke és a szervezés alternativ módszerei realizálhatók legyenek. A szervezetek strukturáját és hatékonyságát már észrevehetően érinti a számitógépek megjelenése. A computeres technológia alkalmazásához kivánatos megkülönböztetni a fizikai erőforrások és az információ áramlását a szervezet müködésében és a külvilággal való kapcsolatában.

A rendszer leirásának szempontjából csak a rögzitett /irott, mágnesszalagon rögzitett stb./ információ lényeges. Ezek a szervezet életében is nagy szerepet játszanak,és súlyos anyagi eszközöket forditanak az infor-

máció tárolására és visszakeresésére.

A későbbiek szempontjából hasznos, ha az információ áramlását két részre bontjuk:

a/ adatkezelő müveletek: ezek az adatok tartalmától, illetve jelentésétől függetlenül adhatók meg. Ilyenek pl. a rögzités, nyomtatás, raktározás és visszakeresés.

b/ információkezelő müveletek: ezek az adat tartalmának ismeretében hajthatók végre az adatokon. Az adat jelentése lényeges, pl. egy beszerzési forrás megválasztása, a megrendelés mennyiségének megállapitása stb.

Az adatkezelő müveletekben, ha azok mennyisége elég nagy a számitógép annyira effektiv, hogy azzal a kézi módszerek nem versenyképesek. Ennélfogva a fejlődés kezdeti szakaszában a számitógép alkalmazása az adatkezelő müveletek számitógépre viteléből állt. Később ismerték fel azt a lehetőséget, hogy a számitógép információkezelő müveleteket is képes végezni, feltételezve, hogy a folyamat megfelelő részletességgel le van irva egy alkalmas számitógépprogram megirásához. Ha az információfeldolgozó müveleteket is megfelelő mennyiségben sikerül gépre vinni, akkor ez a folyamat gazdaságos. Ez nemcsak anyagi megtakaritásban nyilvánul meg, hanem az eredmények gyorsabban rendelkezésre állnak, elmarad a továbbitásukhoz szükséges idő, az eredmények pontosabbak stb. Ezt a folyamatot az a két megfigyelés is elősegitette, hogy egyrészt egy rendszer leirásához a részei által végzett müveletek ismerete is elegendő, másrészt, hogy ezek különböző szervezeti egységek esetén is, minél apróbb részeiben vizsgáljuk a tevékenységet, annál inkább hasonlóak. Meg kell azonban jegyeznünk, hogy ezt a jelenséget a számitógépes technológia bevezetéséig nem sikerült gyümölcsözően kihasználni.

1.2 A rendszertervezés lépései

A számitógépes rendszerek tervezésének alapvető sajátossága, hogy mig a kézi adatfeldolgozás esetében a feldolgozás költsége a munka mennyiségével nagyjából arányos, a számitógépes módszernél a költségek tekintélyes hányadát teszi ki a rendszer kidolgozása, tehát a költségek nagyobbik hányada rögzitett és a kisebbik rész változik a feldolgozott információ mennyiségével. Ez a jelenség arra ösztönöz, hogy lehetőleg nagy rendszereket hozzunk létre. Ennek eredménye az a tendencia, hogy az egyes szervezeteken belül - gazdaságossági okokból - az információ- és adatfeldolgozás centralizálására, számitógépre alapozott központi információfeldolgozó rendszer kialakitására törekszenek. Ennek érdekében gyakran meg kell változtatni a rendszer információáramlási strukturáját, új adatok kezelését is meg kell inditani a csatlakozó egységek kompatibilissé tételéhez.

Ezek után a rendszertervezés feladatát úgy fogalmazhatjuk meg, hogy egy szervezet anyagi müködésének ismeretében megtervezze annak célszerü információáramlási és feldolgozási strukturáját, ezt gépre vihető alakra hozza és abból egy olyan gépen futtatható programot állitson elő, amely alkalmas a rendszer információanyagának kezelésére és a rendszer előre feltételezhető változásaihoz alkalmazkodni tud, /pl. úgy, hogy módositják a programot/.

1.2.1 A PSL/PSA szerepe a rendszerszervezésben

Mivel egy számitógépre alapozott rendszer mindaddig müködésképtelen, mig el nem készül, a rendszertervezőkre fokozott feladat és felelősség hárul munkájuk folyamán. A szervezet evolueiós úton végig önkorrekcióval

történt létrejöttének ezt az előnyét az információfeldolgozó rendszer legfeljebb részleteiben végezheti el, azaz legfeljebb egyes részleteit próbálhatják ki a gyakorlatban és illeszthetik a régi rendszerhez, feltéve, hogy azzal kompatibilis. Ezt a próbálkozást egyébként gazdaságossági okok is indokolják. A felelősséget fokozza, hogy a számitógépes rendszer megvalósitása bonyolult és időigényes munka. A gyakorlatban általában a következő fázisokban megy végbe:

1. Az igények felmérése - ez magábafoglalja a megoldandó feladat megfogalmazását, továbbá egy induló becslést a megoldáshoz felhasznált költségekre és a belőle származó nyereségekre; a logikai rendszertervezés bevezető mozzanata.

2. Logikai rendszertervezés - általában a munka legkreativabb része. Hozzátartozik a szervezet információigényének meghatározása és az információs rendszer tervezése. Eredménye a logikai rendszerterv. Ez a fázis a következő lépésekből áll:

   a/ Adatgyüjtés: a jelenlegi rendszer információáramlásának felderitése, az ezen túlmenő felhasználói igények számbavétele, illetve a lehetséges szervezési változtatások leirása.

   b/ Analizis: az összegyüjtött adatok összegzése és rendszerezése. A leirás hibáinak, hiányosságainak helyenkénti kétértelmüségeinek kideritése és javitása, az átfedések feltárása. Az eredmények csoportositása és előkészitése a megfelelő munkacsoportok számára.

   c/ A javasolt rendszer tervezése: az eljárások megválasztása, amelyeket a célrendszernek végre kell hajtania. Elemzendők a régi rendszer módositásával nyerhető és egy teljesen új rendszer kialakitása közötti alternativák. Az új rendszert kidolgozzák, leirják és elemzik.

d/ Értékelés: megfelelő pontossággal meghatározzák az új rendszer létrehozásának költségeit és előnyeit. Vizsgálják és értékelik a rendszer operációs és funkcionális megvalósithatóságát.

e/ Fejlesztés: az értékelés során a rendszer számos hiányossága derül ki. Meghatározandók az alternatívák ezek javitására és értékelendő, hogy a rendszer további jobbá tételének lehetőségei megérik-e a ráforditandó energiát. Nagyobb lépések megtétele esetén a kiértékelési fázis megismételhető; további adatgyüjtés és analizis is szükségessé válhat.

A fenti lépések a logikai rendszertervezésben természetesen párhuzamosan vagy iterativan, egyre növekvő részletességgel is végezhetők. Az eljárás sok papirmunkával jár, megközelitésére számos módszert dolgoztak ki. Ezek közül néhány számitógéppel végezhető és közülük számunkra a PSL/PSA-val segitett rendszertervezés érdekes.

1.2.2 Számitógéppel segitett tervezés

PSL/PSA-t használva a logikai rendszertervezés alapvető mozzanatai ugyanazok, a legfőbb különbség a két módszer között az, hogy az összegyüjtött adatok és a rendszer addig megtervezett része a gépben foglal helyet, a processzus alatt felépitett un. adatbázisban. Ennek segitségével a tervezés bármely szakaszában dokumentáció /táblázatok, blokkdiagramok, listák stb. formájában/ nyerhető. A PSL/PSA-val végzett logikai rendszertervezés eredménye közvetlenül a számitógépben jelenik meg, azt nem kell - újabb kézi munkával - gépre vinni. Ezen túlmenően a PSL/PSA a következőkben a logikai rendszertervezést segiti annak egyes fázisaiban:

a/ Adatgyüjtés: mivel a legtöbb adatot csak személyes kapcsolatok felvételével nyerhetjük, erre ezután is szükség lesz. Ezeket az adatokat viszont csak egyszer kell rögziteni. A PSA időközbeni outputjai arról is tájékoztatnak, hogy milyen adatokra van még szükségünk.

b/ Analizis: az analizishez tartozó eljárás végrehajtásának igényét a PSA tudja megállapitani. Uj eljárás megjelenése esetén az a PSA-ba beépithető.

c/ Tervezés: ez lényegében kreativ munka, igy teljesen nem gépesithető. Mégis a PSA könnyebben elérhetővé és jobban kezelhetővé tesz az eddiginél nagyobb tömegü adatot.

d/ Értékelés: a PSA első közelités céljaira alkalmas képességekkel rendelkezik, hogy a probléma felállitásában szereplő adatokból annak terjedelmét és a végzendő munka nagyságát becsülje. A program újólag kifejlesztett módszerek figyelembevételével bővithető.

e/ Fejlesztés: lényegében ez is kreativ munka, bár a PSA outputok főleg az értékelés fázisából, hasznosak lehetnek az analistának.

A PSL/PSA outputjai a következő formátumban jelennek meg:

1/ Egyrészük angol szövegként olvasható. Ez a rész adatként tárolódik, de a számitógépprogram nem analizálja. Mégis, mivel a végleges leiráshoz közel, vagy annak kapcsán jön létre, segit a hiányosságok és ellentmondások felderitésében.

2/ Listák. Mivel az adatbázisból készülnek, mindig időszerüek és bármilyen kivánatos rendbe könnyen újrarendezhetők.

3/ Táblázatok, vektorok, matrixok. Szintén automatikusan készülnek, időszerüek, pontosak.

4/ Diagramok, folyamatábrák. A rendelkezésre álló PSA rendszer még nem tud grafikus outputot létrehozni, de a rajzolásukhoz szükséges adatokat rendezve, illetve blokksémában szolgáltatja.

Ujabb hirek szerint már létezik olyan PSA-változat is, amelyik plotterrel készit blokkdiagramokat.

Az igények felmérése és a logikai tervezés után következő fázis a rendszer

3. fizikai megtervezése, azaz az optimális hardware és software konfigurációk meghatározása, ami az információfeldolgozó rendszer technikai specifikálását is lehetővé teszi.

4. Konstrukció - az információfeldolgozó rendszer tényleges felépitése, azaz programirás, file-szervezés, stb. Mivel a file-szervezés is a tervezés kreativ részei közé tartozik, ennek is érdemes részletesebb felosztását adni:

a/ Hálózatanalizis:
a rendszer felépitésére és a használt adatok fajtánkénti mennyiségére vonatkozó paraméterek meghatározása, legegyszerübben a rendszer PSL-leirásából. Direkt úton véghezvihető, de terjedelmes munka.

b/ Az eljárások időzitése;
azaz a rendszer által végzett eljárások osztályozása aszerint, hogy azokat milyen időközönként hajtja majd végre a rendszer, figyelembevéve itt egyes /pl. rendezési/ folyamatok elodázhatóságát és mások elvégzésének szükségességét újabbak végrehajtásához. Ezen osztályok számát a tapasztalat szerint nem célszerü korlátozni.

c/ Modulszervezés;
az eljárások modulokba szervezése oly módon, hogy - figyelemben tartva a precedenciákat, a modulok és a puffertárak méretét - az elvégzendő be- és kimeneti müveletek számát minimalizáljuk.

d/ File-okba szervezés;
az egyes adatokat úgy kombináljuk file-okba, hogy minimalizáljuk az egyszerre igénybevett be- és kimeneti egységek maximális számát, figyelemben tartva a precedenciákat, az adathalmazok strukturáját és a pufferméreteket. Erre és az előző lépésre egyaránt vonatkozik, hogy az összes matematikailag lehetséges kombináció helyett a gazdaságossági okok miatt célszerübb az eljárásokat kötegelni, vagy más szuboptimális eljárást alkalmazni.

e/ A file-ok megtervezése;
minden file-hoz megválasztjuk az adatok reprezentációjának módját, a blokkméretet és a hozzáférési módot.

A konstrukciós lépés a logikai rendszerterv használatbavétele.

5. Tesztelés, konvertálás és használatbavétel /a felhasználó gépén/ - magába foglalja az új információfeldolgozó rendszer megbizhatósági tesztjét is.

6. Futtatás, és a müveletek táblázatba foglalása, ellenőrzése.

7. Módositások és fenntartás: az információs rendszer müködésben tartása az eredeti szervezet változásainak megfelelően, a program hatékonyságának monitoros ellenőrzése, a lehetséges változtatások vizsgálata.

1.3 A rendszertervezés problémái

A gyakorlatban ez az eljárás alapvetően a fentebb vázolt lépésekben zajlik le, bár néhol több lépés párhuzamosan, néhol az egész folyamat iterativan hajtódik végre. Bár az eljárás menete nagyjából adott, a gyakorlatban számos probléma adódik vele kapcsolatban.

1.3.1 A szervezet leirásának módja Blokkrendszer

A rendszertervező első problémája, és nem is a legkönnyebb, bármilyen furcsán hangzik is ez, az, hogy mit akar megvalósitani. Pillanatnyilag is a logikai rendszertervezés az egész rendszertervezői munka egyik legkreativabb szakasza és ennek javitásával nagyot lendithenténk az egész tervezői munka minőségén, ha azt egy jobban definiált problémafelvetés által szorosabb kapcsolatba tudnánk hozni a leirandó szervezettel. A baj az, hogy preciz és tömör leirást a szervezetekről nem tudunk adni, amely alkalmas lenne a probléma megfogalmazására és a rendszer megtervezésére. Ha pedig ez nincs meg, akkor gyönyörü rendszereket tudunk ugyan tervezni, de azok mégsem müködnek kielégitően, mert nem csatlakoznak a felhasználói igényekhez, azaz végeredményben nem azt csinálják, amit kellene.

Ezért az automatizálási folyamat továbbfejlesztése helyett aktuálisabbnak tünik az energiát a jobb problémamegfogalmazásra koncentrálni.

Közvetlenül kapcsolódik ehhez a problémához a programcsomagok átvételének kérdése. Annak eldöntéséhez ugyanis, hogy egy programrendszer egy hasonló szervezet leirásához és annak információkezelő rendszeréül alkalmas-e, a probléma jó megfogalmazása, azaz a rendszerek

jó leirása szükséges. Ha nem vagyunk tisztában a rendszer, illetve a programcsomag tevékenységével, akkor azokat nem tudjuk egymáshoz illeszteni, az nem fog kielégitően müködni, sőt példák szerint már az illesztés is teljesen lefulladhat, bár a programcsomag az eredeti felhasználás területén jól müködik és az eredeti felhasználó sem tudja, miért nem válik be az új helyen és mennyi időbe telne az /elég határozatlanul jelentkező/ hiba rendbehozása. Mindez annak dacára, hogy a rendelkezésre álló programcsomagok közül az elérhető legjobbat választották, amely kétségkivül alkalmasnak tünt a probléma megoldására és valószinüleg az is.

A logikai tervezés módja természetesen függ a szervezet méretétől és bonyolultságától. A szervezet méretén első közelitésben a szervezet teljes /a rendszer tervezéséhez felhasznált/ tömör dokumentációjának mennyiségét értem kilogrammban, bonyolultságát pedig az egy egységéhez közvetlenül kapcsolódó egységek átlagos számával mérhetem.

A rendszer felépitési módja befolyással van az eredményre, azaz a kész rendszer strukturájára is. Ismertek hat éve épitgetett rendszerek, melyek kiválóan müködnek, de még mindig távol állnak a befejezettségtől és folyamatosan fejlesztik is őket, mások pedig már a tervezett költség két-háromszorosának felhasználása után kezdtek müködni; gondolom példákkal az olvasó is tud szolgálni. Ennyiben a rendszertervező folyamat befolyása a kész rendszer minőségére egzakt módon mérhető.

1.3.2 A program karbantarthatósága

Felvetődik a kérdés, kivánatosak-e folyamatosan továbbfejlesztett programok? Feltétlenül, mert csak ezek tudnak alkalmazkodni a szervezet változásaihoz, és csak

ezek tudják kielégiteni több felhasználó szimultán igényeit a szervezet különböző részeinek különböző részletességgel történő leirására.

Vegyük itt figyelembe, hogy a program önmaga által szervezett alkalmazkodóképessége azért is fontos, mert bonyolult rendszereknél a szervezet egy egységének megváltoztatása sok kapcsolódó egységre kihat. Igy fontos kérdés egy információs rendszer fejleszthetősége és egyáltalán nem magától értetődő. Ennek magvalósitására eddig a blokkrendszerü programok tünnek a legalkalmasabbnak, amelyek egyébként a szervezetek túlnyomó többségének a felépitését is követik. Az ilyen programon belül az egyes blokkok, illetve azok blokkjai stb. változtathatók, cserélhetők és ha a blokkok közti kapcsolatok tisztázottak, az is könnyen megállapitható, hogy mely további blokkokon kell egyidejüleg változtatni. A fejlesztés folyamán rendszerint a blokkrendszer is megváltozik a programcsomag egymást követő verzióiban, és igy az újabb változtatások egyre nehezebben tervezhetők és követhetők hatásukban. Ezért, mivel a programcsomagokat is időtartamra tervezik, /összhangban a szervezet fejlődéséből eredő memóriaigény - növekedés és az adott gépi konfiguráció memórialehetőségeinek viszonyával/, a blokkrendszer szerkezetét célszerü olyan állapotban tartani, hogy az a program kifutási idejének leteltéig kellően leirható maradjon ahhoz, hogy a programon változtatni lehessen. Ennek megvalósitása adja a lehetőséget arra, hogy a felhasználó a rendszert ténylegesen kezelni tudja.

Ennek másik feltétele az, hogy a usert bevonjuk a rendszer felépitésébe, mivel ettől függ, hogy azt mennyire tudják elfogadni és megitélni a változtatások szükségessége szempontjából. Ennek hiján a user nem is fog foglalkozni a rendszer fejlesztésének gondolatával és

igy nem alakul ki közte és a rendszer közt megfelelő kapcsolat. Kérdés ezért, hogyan és mikortól fogva vonjuk be a usert a tervezésbe és a rendszer megvalósitásába.

Általában minél korábbi fázisban sikerül a usert a tervezés aktiv részesévé tenni, annál jobb. Passzivan ugyan már az adatgyüjtésnél mindenképpen közremüködik, de ha nem sikerül /az igények felmérésétől kezdve/ valamelyest aktivizálni, akkor azon túl, hogy a rendszerrel nem barátkozik meg, homályban marad a rendszerrel kapcsolatos igénye és ez egyébként elkerülhető hibák ismételt elkövetésének forrása lehet, mig a user aktivitása lényegesen effektivizálni tudja a tervezői munkát.

A fentiekre tekintettel a rendszerrel való foglalkozást müvi úton is igyekeznek a szervezeten belül biztositani. Ez általában a vezetésre hárul, pl. úgy, hogy a szervezetnek a vezetés tagjai közti felosztásának megfelelően mindegyik tag a rendszer megfelelő részének fejlesztéséért felelős. Ennél fejlettebbnek tünik az Egyesült Államok legfőbb Állami Számvivőszékének módszere, amely szerint a vezetés a rendszer fejlesztésének csak távlati problémáival foglalkozik, mig a részletek egy rendszerfejlesztő bizottságra hárulnak.

1.3.2.1 Dokumentáltság

A rendszer karbantarthatóságának második feltétele annak kellő dokumentáltsága. Ennek céljai a következők:

1. a/ adatgyüjtés folyamán az adatbank kialakitása

   b/ jelzi a kialakitandó rendszer célját. Ez minél előbb megvalósul a tervezésben, annál jobb, mivel ezáltal válik a rendszer egynél több megfigyelő részére is hozzáférhetővé.

c/ mivel a rendszer célja itt van megfogalmazva, a kész rendszer jóságát annak dokumentációival való összevetésével lehet majd mérni. Erre azért is szükség van, mert a müködő rendszer jóságának mérése amúgy is probléma.

d/ hasonlóan a rendszer körüli vitákban a tervezés alatt tényekkel támogat és kapaszkodóul szolgál.

e/ a tervezésben kontrollt tesz lehetővé; ez alatt hasonló hibák többszöri elkövetésének megelőzése értendő. Szükséges ez azért is, mert a tervezéshez rendelkezésre álló információ másodkézből származik, tehát ellenőrizhetetlen hányada nem felel meg a valóságnak. Hasonlóan, mivel a rendszer felhasználása és értékelése is áttételesen történik, itt is fennáll a kontroll szükségessége.

f/ a rendszer karbantarthatóságához és változtathatóságához is szükséges, mivel kellő dokumentáció hiján egyszerübb új programot irni egy rendszerhez, mint a régiről kideriteni, hogyan müködik. Ez a legfőbb oka annak, hogy "úgyis meg kell csinálni" a dokumentációt. Ez indokolja azt is, hogy a rendszerfejlesztési tevékenységgel párhuzamosan azt célszerü a szervezet igazgatásának átadni.

g/ és végül a rendszer dokumentációjának történeti értéke lehet, de ez a tervezés fázisában többnyire lényegtelen.

A fentiek figyelembevételével a dokumentáció célját és használhatóságának mércéjét abban láthatjuk, hogy a rendszerről olyan információt szolgáltat, amit abba vissza lehet táplálni és ezáltal a rendszer javitható. Ez a fő cél határozza meg alapvetően a rendszer dokumentációjának tartalmát is, melynek részletes ismertetése helyett az annak szabványositására tett javaslat érdemel figyelmet, ami a 123 sz. ISDOS füzetben talál-

ható meg. Ez a szabványos dokumentáció tartalmán kivül annak leirási rendszerét is megadja, mindezt olyan formában, hogy az a PSA adatbázisból közvetlenül kinyerhető.

Végül a dokumentációnak szükséges sajátossága, az olvashatóság, ami azt jelenti, hogy a benne rejlő információ az olvasó számára feleslegesen terjedelmes dekódolási munka nélkül váljon hozzáférhetővé.

1.3.2.2 Csatlakozó felületek megőrzése

A program kézbentarthatóságának harmadik feltételéhez először hozzávetőlegesen meghatározom egy rendszer kapcsolódó felületeinek fogalmát. E célból a rendszert egy irányitott gráffal reprezentálom, amelyben a szögpontok a müveleteket végző egységeket, a nyilak pedig az információ áramlását jelzik. Egy információs rendszer csatlakozó felületein a rendszer információáramlását leiró gráfban egy-egy él által reprezentált információ tartalmát és formáját értem; ez az él kiindulási pontjának kimenete és a végpontjának bemenete s mint ilyen, a szögpontok által reprezentált tevékenységeket funkcionálisan, ha csak statikusan is, de jellemzi. Ezen definició birtokában megfogalmazható a rendszer változtathatóságának harmadik feltétele: ne változtassunk a csatlakozó felületeken. Tehát, ha a rendszerben, egy vagy több egymáshoz csatlakozó egységet megváltoztatunk, a régi és az új résznek ugyanazok legyenek a be- és kimenő információi, tartalmukat és formájukat tekintve egyaránt. Ennek a kikötésnek az oka a következő: a rendszer megadása végeredményben annak csatlakozó felületeivel történik, azaz a legdurvább közelitésben a rendszer be- és kimeneteivel, egyre részletesebb leirásokban pedig a rendszer - egyre kisebb - blokkjainak ki- és be-

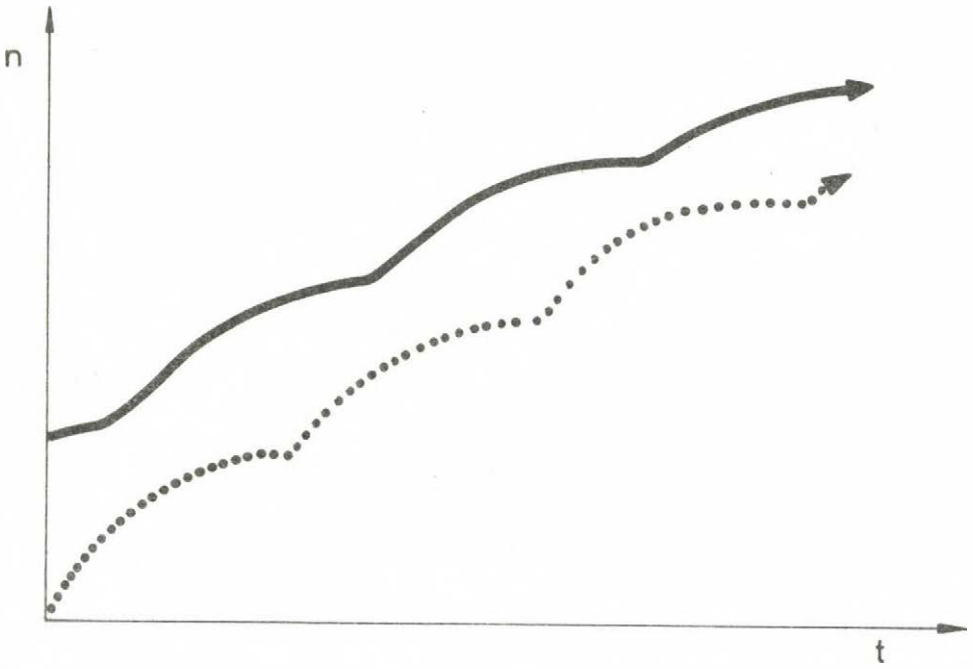
meneteinek megadásával. Amig ezeket tiszteletben tartjuk, addig az egyes blokkok cserélhetők, mihelyt azonban változtatunk rajtuk, a rendszer eredeti leirása nem használható többé és a rendszer kicsúszott az ellenőrzés alól; végeredményben nem tudjuk többé, hogyan lehet rajta változtatni. /A rendszerek blokkokra való felosztottsága a szervezetek hasonló strukturájának követésével jött létre, és terjedelmes rendszerek esetén a leginkább bevált épitési mód./ Ezért, ha kell, akár mesterséges eszközökkel is célszerü a rendszer csatlakozó felületeit változatlanul tartani.

Ennek mindmáig legszellemesebb módszere a programok gépi generálása, amelyre a gyakorlatban a PSA programcsomag megadása a példa. Ez úgy történik, hogy definiálnak /be- és kimeneteivel/ néhány blokkot, amit a felhasználó saját gépi kódjában készit el és megadnak egy segédprogramot, amely a végleges programot a gépi kódú blokkok /mint input/ felhasználásával outputként produkálja. Ez egyrészt a PSA programcsomag adott gépen történő realizációját könnyiti meg az "összeollózás" müveletének elkerülésével, másrészt a csatlakozó felületek tiszteletben tartása esetén a program módositását és újrairását is megkönnyiti, amennyiben ez a megfelelő blokkok újrairására és a PSA csomag újragenerálására redukálódik. Ezáltal a programgeneráló segédprogram léte is abba az irányba hat, hogy a programcsomag kapcsolódó felületeit változatlanul tartsuk. Ezt többnyire keresztül is lehet vinni, kivéve azzal a szervezettel szemben, aminek az információs rendszerét megalkottuk.

Miután megvizsgáltuk az információs rendszerek kézbentarthatóságának feltételeit, a következő probléma a rendszer karbantartásának a munkaigénye és ennek várható alakulása.

1.3.2.3 A karbantartás munkaerő igénye

A rendszer karbantartásának munkaigénye és ennek várható alakulása a 2. ábrán van feltüntetve, t idő eltelтével, mint a program blokkjai számának /folytonos vonal/ és mint az eddig javitott blokkok számának /szaggatott vonal/ alakulása, a rendszer üzemeltetésének kezdetétől.



2. ábra

A rendszer beinditása után az első időszak hibák azonositásával és javitásával telik el, amit a szaggatott görbe erőteljes emelkedése reprezentál. Mikor itt csökken a tennivalók mennyisége, részben a rendelkezésre álló munkaerő miatt is, a rendszert erőteljes fejlesztő

tevékenységnek vetik alá, amitől blokkjai megszaporodnak, /ez a hibajavitás fázisában kevésbé jellemző/, azután a javitás miatt újabb hibák keletkeznek, illetve válnak érezhetővé, és ettől fogva a két periódus váltja egymást, miközben a rendszer blokkjainak száma és a módositott blokkok száma, úgy tünik, a parkinsoni törvények szerint tart a végtelenhez.

1.3.3 Szimulálás

A következő probléma a programok szimulálása, annak használhatósága, illetve létjogosultsága. Az utóbbi alapvetően onnan ered, hogy a rendszertervezésben rendesen nincs lehetőség kontroll kisérletre. Másik rendszert sehol nem valósitanak meg a tervező kedvéért, hogy azzal a sajátjának teljesitményét összevethesse. Ennek hiján pedig a rendszerek teljesitményének mérése bizonyos fokig a levegőben lóg, bár még két müködő rendszer jóságának összehasonlitása sem triviális feladat. A másik rendszer megvalósitása helyett igy annak software éshardware módszerekkel történő szimulációját választhatjuk, aminek alapvetően a rendszer blokkdiagramjával vannak rokon vonásai: vagy nem elég részletes és igy csak durva eredményeket ad /azaz nem kellően pontos becsléseket szolgáltat/, vagy ha kellően finom, többnyire akkor valósul meg, mikor a létrehozását motiváló kérdés már idejét múlta és lérehozása emberi munkában, illetve használata /a szimuláció esetén gépidőben/ nagyon energiaigényes. Azért sem alkalmazzák túl gyakran, mert viszonylag ritka a semmiből indulva közvetlenül létrehozott nagy rendszer és ez a menetközbeni választások lehetőségét eleve korlátozza. Mire jó tehát mégis a szimuláció? Elsődleges haszna nem a tervezés közbeni döntéshozatalnál van, hanem a folyamatok feletti átlátás megszerzésére, konfigurációanalizishez,

mintegy az áttekintőképesség fejlesztésére használható. Igy használhatósága nem korlátozódik a rendszer tervezésének szakaszára, viszont korlátozódik annyiban, hogy méretezőeszközként nem célszerü alkalmazni.

1.4 Kész rendszerek értékelése

A gyakorlatban problematikus pont, a kész rendszerek átvétele, illetve ellenőrzése. Az utóbbi nyilvánvalóan csak használat közben végezhető el, ami az előzőt is problémássá teszi. Ugyanabban az irányban hat az is, hogy a rendszer célját és igy jóságának mércéjét is a felhasználó adja meg /ha megadja explicite/ és ennek a tervezés folyamán használt mércével való egyeztetése is az ellenőrzés és átvétel feladata lenne.

Az átvételre vonatkozó, kialakult és a lényeget érintő szabály alig van. Sőt, már a rendszer kivitelezhetőségének vizsgálata is nagyban eltér, igy pl. a Procter Gamble-nél a költségek évi 20 %-ának várható visszatérülése a kivitelezhetőség kritériuma, az US Army Combat Development Command bonyolultabb, de szintén sematikusan rögzitett módszerekkel végez becslést a várható haszonra és költségre. Egyes cégek a rendszer fejlesztését saját kockázatukra végzik, mások ennek költségét közvetlenül felszámitják a megrendelőnek.

A rendszer átvételének eljárása méginkább változó. Néha "nincs formális átvétel", mert "a vevő úgyis ellenőrzi, hogy mit kap a pénzéért, és ez a legjobb garancia arra, hogy a rendszer effektiv legyen". Máshol az átvétel és ellenőrzés módja esetenként szerződés tárgya. A Procter and Gamble cégnél formális átvétel van, és azt megismétlik 12, ill. 24 hónap múlva, "de valahogy az utolsóra nem szokott sor kerülni".

A rendszer hatékonyságának ellenőrzésére sok helyen folynak elméleti kutatások, de ezek rendszerint kissé a levegőben lógnak.

A rendszer átvétele, illetve ellenőrzése is alkalom arra, hogy a felhasználót bevonjuk a rendszer megismerésébe, de ennek célja már a rendszer használatának megismerése. Az, hogy napjainkban software-t növekvő mennyiségben kell előállitanunk, hogy az erre forditott anyagi források nagyobbak, mint a hardware-re forditottak, egyfajta szakmunkáshiányt okozott, a programozók hiányát. Ez számos problémát a felszinre hozott. A mai módszerekkel termelt software megbizhatatlan, nehezen karbantartható és kiterjeszthető, nem kellően hatékony és kifejlesztésének idő- és pénzigénye nem becsülhető előre kellő biztonsággal. Ezeket a problémákat több irányból is kisérlik megközeliteni. A jelentősebbek a strukturált programozás, az általános szerkezetü adatbázisokat kezelő rendszerek és az automatikus programozás. A következő részben ezeket szeretnénk a rendszerszervezői munka egy részeként értékelni.

1.4.1 Számitógépes segitséggel létrehozott rendszerek

Számos előnyük van. Talán a legfeltünőbb a gazdaságosságuk: a rendszeranalistáktól és programozóktól átveszi a munka egy részét a számitógép, a program kifejlesztési ideje és költsége csökken és szakositott munkaerő szabadul fel. Nagyobb termelékenységet érünk el oly módon, hogy az ember-gép kapcsolat érintkező felületét szükitjük / főleg terjedelmében/. Csökkennek a tesztelés és hibajavitás költségei, mivel a hibák száma és a lehetséges hibafajták száma is csökken. Az elkészülő rendszer és rendszerleirás adekvát volta könnyebben ellenőrizhető az automatikusan elkészülő, a PSL/PSA-ban

szövegként olvasható, tehát érthetőbb rendszerleirás segitségével.

Természetesen a módszernek hátrányai is vannak: több software eszközt használnak, amit a gépben el kell helyezni és amire felügyelni kell, a gépidő-költségek növekednek, a software eszközök egyre komplexebbeknek bizonyulnak, belsőleg is és használat közben is. A legsúlyosabb azonban az, hogy a létehozott információs rendszer kevésbé lesz effektiv, mint a "kisipari" munkával készült.

A helyzet emlékeztet az első complilerek bevezetésének idejére, és ha az ott szerzett tapasztalatokat irányadónak tekintjük, a kézi és a gépi úton készült rendszerek hatásfoka közti különbség a rendszertervező rendszerek finomitásával csökkenni fog.

1.4.2 A rendszertervezés hatékonyságának mérése

A jelen pillanatban tehát azt a kérdést vethetjük fel a felhasználók szempontjából, hogy annak a tevékenységnek, melynek folyamán /számitógéppel vagy anélkül/ egy információs rendszert hoznak létre és azzal egy értelmes kifutási ideig egy szervezet információit feldolgozzák, hogyan mérjük a hatásfokát, vagy legalábbis két lehetséges módszer közül hogy válasszuk ki a hatékonyabbat.

Kézi adatfeldolgozásnál a tapasztalat szerint is optimális hatásfokú információs rendszerekre célszerü törekednünk. Gépi információfeldolgozás esetén azonban, mint azt a korábbiakban láttuk, a rendszer létrehozásának és üzemeltetésének költségei összemérhetők, igy az optimális rendszer létrehozása nem indokolt. Ez a

megállapitás teremti meg egyben a lehetőséget a rendszerek gépi tervezésére is.

Az eddigieknek megfelelően a rendszertervezési tevékenység jóságát legalább háromféleképpen mérhetem:

a/ a létrehozott rendszer hatásfokát vizsgálom /ezt a közelitést alkalmazom a kézi munkával létrehozott rendszereknél/

b/ egy adott rendszer létrehozásának költségét, ill. hatékonyságát vizsgálom

c/ a rendszer létrehozásának költségét és a létrehozott rendszer hatásfokát összevetve hozok döntést arról, hogy milyen áron /ebbe beleértve a rendszer létrehozását is/ tudom a szükséges információkezelő müveleteket elvégezni. A három módszer közül nyilván az utolsó a legjobb.

Mindhárom értékelési módnál a kivitelezésre rányomja a bélyegét, hogy ellenőrzött kontrollkisérlettel, tehát ugyanarra a feladatra a miénktől független megoldással általában nem rendelkezünk; ezt a kontrollt alkalmazni túl költséges volna.

Az a/ módszerrel történt értékelés hátrányaira az jellemző, hogy végeredményben nem csak a létrehozott rendszert mérjük vele. Egyrészt azt a tényezőt nem tudjuk leválasztani az értékelésről, hogy a szolgáltatott /többnyire másod- és harmadkézi információk/ mennyire vezették félre a rendszertervezőt. Másrészt az igy nyert értékelés végeredményben a felhasználó véleménye a rendszerről, aki a rendszer jóságának megitélésére több okból alkalmatlan lehet /lévén, hogy többnyire ez az első számitógépes rendszer, amivel dolga van/. További el nem választható tényező, hogy a rendszertervező mennyire tanitotta meg a felhasználót a rendszert használni,

és a felhasználás ill. fejlesztés további lehetőségeit megtalálni. Ez ugyan a felhasználó részére a rendszer jóságánál nem kevésbé lényeges szempont, de nem is az, amit a felhasználón keresztül mérni akarunk.

A b/ módszer szerinti értékelés hátrányait jól érzékelteti az egyes rendszerek bevezetéséről ill. fejlesztéséről folytatott vitákban használt, fentebb már szidott módszerek bizonytalansága. Ezt az értékelést is nehéz úgy végezni, hogy csak a rendszer létrehozásának hatékonyságát jellemezze. Az egyik lehetséges megközelités a létrehozás költségének vizsgálata pl. forintban, de nyilvánvaló, hogy ez mennyire függ attól, hogy a felhasználónak a tényleges költségekből mit és hogyan számitanak fel. /Pl. a felhasznált gépidőt egyes cégek közvetlenül felszámitják a rendszertervezés költségeként, mig mások közvetve./ Más lehetséges megközelitések még a tervezéshez szükséges idő, illetve az összes megirt program /sorokban mérve/ aránya a végleges program terjedelméhez.

Mint láttuk, az első két módszer az utólagos értékelés céljaira használatos. Döntéshozatalra a c/ módszer alkalmas. Sajnos, ez a legkevésbé egyszerü a három módszer közül és erről esik a legkevesebb szó az irodalomban. Általában azzal intézik el, hogy a tőkeberuházási döntések technikája alkalmazható rá. Ezenkivül több szerzőt emlegetnek, akik igen előrehaladott kutatásokat végeztek e témában, de ezen eredményeket elvontságuk miatt még nem sikerült hasznositani. /ld. 1 5.o./

1.4.3 A rendszer létrehozása egyes fázisainak költsége

Tájékoztató jelleggel megemlitjük, hogy egy Arthur Andersen által végzett felmérés szerint egy információ-

kezelő rendszer megvalósitásának és futtatásainak összes költségét figyelembe véve a rendszer installációja ennek 50, karbantartása 39 %-át tette ki átlagosan, különböző technikákat és jól felkészült rendszertervezőket alkalmazva.

1.5 Áttekintés a számitógépes rendszertervezés állásáról

Végül egy rövid áttekintés a számitógéppel segitett rendszertervezés állásáról /ld. még 4 1975 május/: a szervezet leirásából a logikai rendszertervet a PSA, ill. a SODA programcsomagokkal állithatjuk elő. Az első esetben a PSL nyelvben, a másodikban is megfelelő nyelvben megfogalmazott szervezet-leirás szükséges. A SODA programcsomag ennél tovább megy, segitséget kiván nyújtani egy jó file- illetve modulszervezés megvalósitásához is, de ezt az irodalom a PSL/PSA csomagnál sokkal kevésbé rugalmasnak és kevesebb esetben alkalmazhatónak itéli. Ilyenformán a file- és modulszervezés nem megoldott. Ennek kézi elvégzése után a MODEL, ill. a HSL/1 programcsomaggal nyerhető futtatható program a file- és modultervből. A kapott program általában erősen suboptimális. /A HSL/1-ről ld. 1 8.o./ Ezeken kivül számos különböző célú segédprogram létezik, legtöbbjük fejlesztés alatt, /ld. 4 3-4.o./ amelyekről kellő irodalom hiján nem ejtek bővebben szót.

2. A PSL HASZNÁLATA AZ INFORMÁCIÓS RENDSZEREK TERVEZÉSÉBEN ÉS DOKUMENTÁLÁSÁBAN

A manuális módszerekkel végzett információs rendszertervezésnek van néhány alapvető nehézsége, amelyek egy részéről az első fejezetben már szó volt, de mielőtt a PSL nyelv ismertetésére térnénk, célszerü összefoglalni. Az itt kiemelt problémákat igyekszik a PSL/PSA rendszer megoldani.

- Az igények felmérésekor, a rendszertervezés előkészitő fázisában nehéz az elkészitendő rendszer minden részfeladatát meghatározni. A specifikáció pontossága, nagymértékben függ attól, hogy milyen rutinos a rendszertervező, és mennyire határozottak a megrendelő elképzelései. Nincsenek olyan egységesen megfogalmazott szempontok, amelyek segitenék az előkészitő munkát.

- A logikai tervezés fázisában a rendszerek leirása szöveges megfogalmazásban és blokkdiagramok, táblázatok készitésével történik. A nagyobb információs rendszerek egy bizonyos részletességü leirása már nehezen áttekinthető, igy egyre nagyobb a logikai hibák előfordulásának valószinüsége. Ekkor még nagyobb mértékü változtatásokat is végeznek, amelyek - szintén a nehéz áttekinthetőség miatt - újabb hibaforrásokat rejtenek magukban. A logikai tervet ellenőrző személy nem minden esetben tudja a rendszertervező gondolatmenetét magáévá tenni, igy ő sem vesz észre minden hibát. A fenti nehézségeket még fokozza, hogy a nagy rendszereket több személy tervezi és a közöttük lévő kommunikáció is kivánnivalót hagy maga után. Kivánatos lenne tehát olyan logikai terv készitése, amely tömör megfogalmazású, áttekinthető, egységes szempontok szerint készül, és a résztervek között állandó és teljes a kommunikáció.

- A fizikai megvalósitás problémái a logikai tervezésben

megfogalmazottakból adódnak. A logikai rendszerterv realizálásakor a programozónak teljes egészében azonosulnia kell a rendszertervező gondolatmenetével. Előfordulhat, hogy ez nem sikerül, és a kész program nem pontosan a követelményeknek megfelelően müködik. Ugyancsak a fizikai megvalósitás fázisában okoznak fennakadást a rendszerterv logikai ellentmondásai. Az ekkor végzett javitások újabb hibaforrások, és esetleg a már müködő részterületeken is módositani kell. Látható, hogy ezeken is a tömör, áttekinthető megfogalmazás, egységes kifejezések használata, és logikai hibáktól mentes rendszerleirás segithet.

- A kész információs rendszer dokumentációjáról ugyanaz mondható el, mint a logikai rendszertervről: tömör, áttekinthető, egységes szemléletü leirás szükséges. Súlyosbitja a gondokat, hogy a rendszer használata közben történő módositások nehézkesek. A nagy rendszerekben végzett módositások tovább bonyolitják a leirást, a dokumentációban nehéz követni a változásokat. Végül a rendszer teljesen áttekinthetetlenné válik, és újat kell késziteni. Ezen úgy lehet segiteni, hogy a dokumentáció alapján a legcélszerübb módon végezzük el a módositásokat és ezeket a dokumentáció megfelelő részeinek újrainditásával rögzitjük.

- Nem elhanyagolható, hogy mennyi idő alatt készül el a rendszerterv. Előfordul, hogy egy nagyobb információs rendszer előkészitése és logikai tervezése 15-20 emberévet igényel. Kivánatos lenne ezt az időt csökkenteni, és reális, hogy a jelenleg rendelkezésre álló eszközökkel, módszerekkel 1-2 emberév munkával ekkora rendszereket el lehessen késziteni.

A PSL/PSA információs rendszertervező és dokumentáló nyelv létrehozói elsősorban ezeket a problémákat igyekeztek megoldani. Létrehoztak egy olyan számitógépes dokumentációs

rendszert, amelynek segitségével jól definiálható bármely információs folyamatrendszer, annak környezete, leirhatók a végbemenő események, relációk, adatáramlások, automatikusan megvizsgálja és kijelzi az alapvető logikai ellentmondásokat, az adatbázisban dokumentálja a kész rendszert, és róla többféle megközelitésből könnyen információhoz juthatunk. A módositásokat azonnal beilleszti a korábban leirt információs rendszerbe, és az új dokumentáció azonnal rendelkezésre áll.

A PSL/PSA alkotóelemei a következők:

a/ PSL /Problem Statement Language/: az információs rendszer definiálására, megfogalmazására szolgáló nyelv. A rendszertervező PSL-ben fogalmazza meg elképzeléseit, és ezen a nyelven készülnek a logikailag jó rendszer listái, dokumentációi.

b/ Adatbázis: az információs rendszer adatainak rendezetten, redundanciamentesen tárolt halmaza a számitógép háttérmemóriájában /disk-en/.

c/ PSA /Problem Statement Analyzer/: megteremti a kapcsolatot a rendszertervező, felhasználó és az adatbázis között. PSA nyelven irt utasitások végzik el a PSL-ben megirt információs rendszer felvitelét az adatbázisba, a listázásokat, és az adatbázis kezelését.

2.1 Információs rendszerek logikai tervezése PSL/PSA segitségével

A nagyméretü információs rendszerek logikai tervének elkészitése hosszú folyamat, és minden tekintetben alapos, aprólékos munkát igényel. A manuálisan végzett rendszertervezés módszere többféle lehet, a rendszertervező munkastilusa és gyakorlata szabja meg, hogy

milyent alkalmaz. A PSL/PSA segitségével történő tervezés folyamata is tetszőleges lehet, de a PSL strukturájából egy olyan módszer adódik, amelynek alkalmazása határozottan előnyös: ez a "felülről közelitő" /TOP-DOWN/ módszer.

A TOP-DOWN módszer lényege az, hogy az információs folyamatrendszer követelményeinek megfelelő nagyvonalú rendszertervet állitunk össze, és ennek fokozatos kifejtésével jutunk el a részletes rendszerleiráshoz, amely alapján hozzá lehet kezdeni a fizikai megvalósitáshoz. A PSL/PSA segitségével történő rendszerterveés a következő lépésekben történik:

a/ A követelmények megfogalmazása PSL-ben. Az első lépésben csupán egy nagyvonalú rendszertervet kell készitenі, amely tartalmaz minden főbb tevékenységet. A további lépésekben ezt kell kifejteni. A végső cél, hogy a kivánalmaknak megfelelő részletességü rendszerleirásunk legyen, és az adatok elemi szintig legyenek lebontva. A megfogalmazás manuálisan történik PSL nyelven.

b/ A PSA parancsok segitségével az információs folyamatrendszer leirásának felvitele az adatbázisba. Ekkor kell meghatározni, hogy a PSL megfogalmazás felvitelén kivül még milyen müveleteket kivánunk elvégezni. Például törölni lehet az adatbázisba korábban felvitt információt, különböző listákat kérhetünk, amelyek segitséget nyujtanak a további munkához.

c/ Gépi müveletek. Ekkor a rendszertervezőnek nem kell beavatkozni, a számitógép elvégzi a PSA prancsok által adott utasitásokat, elkésziti a listákat.

d/ A kapott listák kiértékelése. A rendszertervező

megvizsgálja, hogy a leirt rendszer tartalmaz-e ellentmondásokat, történt-e elirás. Ezeket kijavitja, az adatbázist további információkkal bővíti az a/ pontban leirtaknak megfelelően. Ha a rendszerterv készen van, a dokumentáció elkészült, akkor a munkát átadja a programozónak, vagy a megrendelőnek.

2.2 A PSL felépitése

A PSL alkotói arra törekedtek, hogy a leirt információs rendszer megfeleljen a számitógépes felhasználás és a rendszerdokumentálás szempontjainak is. Ezért a PSL strukturája között, a használható alapszavak egyértelmüen meghatározzák rendeltetésüket, ugyanakkor a leirás angol nyelven jól olvasható.

A PSL az információs rendszert az őt alkotó objektumok jellemzőinek, kapcsolatainak definiálásával irja le. Az objektumok rendszerleiró elemek, amelyek az információs rendszerben valamilyen konkrét vagy elméleti "tárgyat" képviselnek. Például objektum lehet egy logikai adathalmaz, vagy egy olyan folyamat, amely definiálja, hogy az adatok honnan származnak. Az objektumot leiró szavak lehetnek:

a/ a rendszertervező által alkotott szavak;

b/ PSL alapszavak.

A rendszertervező által alkotott szavak lehetséges tipusai:

a/ Objektumnév: ennek funkcióját, tulajdonságait, más objektumokkal fennálló kapcsolatait irjuk le PSL nyelven. Minden PSL objektum egy objektumtipushoz tartozik, amelynek leirása a 2.3 részben található.

b/ Rendszer-paraméter: Az információs rendszer méreteiről, használatának gyakoriságáról ad tájékoztatást.

c/ Elbeszélő leirás: a rendszertervező és a későbbi felhasználók részére irt kommentárok, amelyek segitik megérteni egy-egy objektum funkcióját vagy az egész rendszert.

A PSL alapszavak megadják az objektumok szerepét, helyét az információs rendszerben, leirják a köztük lévő kapcsolatokat, vagyis ezek által áll össze maga az információs rendszer.
Tipusai:

a/ Objektumokat definiáló szavak /objektumtipusok és szekciótipusok/.

b/ Állitások alapszavai.

c/ Kiegészitő szavak.

A PSL strukturája olyan, hogy tömören képes leirni az információs folyamatrendszert. A leirás szekciókra /fejezetekre/ tagolódik, ahol minden szekció egy-egy objektumot ir le. A szekció első állitása az objektumot definiáló kifejezés, amelyet az objektum tulajdonságait és kapcsolatait leiró állitások követnek, tartalmazva a vele kapcsolatban álló objektumok neveit. Minden egyes állitás a megfelelő PSL alapszóval kezdődik, ezt követik a PSL-nevek, rendszerparaméterek vagy szöveges leirások. Egyes esetekben az állitás belső részében is előfordulnak alapszavak. Az állitás végét pontosvessző jelzi.

A kiegészitő szavak az állitásokba épülnek, ezek használata nem kötelező. Céljuk, hogy a PSL állitásokat - amelynek alapszavai angol nyelvüek - értelmes angol mondatokká füzzék.

2.3 Objektumokat definiáló alapszavak

Az objektumtipusokat és a köztük lévő kapcsolatokat úgy csoportositjuk, hogy egyúttal bemutatjuk, milyen szempontból jellemezhető velük az információs folyamatrendszer. A rendszer objektumai funkciójuk szerint a következőképpen osztályozhatók:

- az információs folyamatrendszer környezetét leiró objektumtipus;
- az információs folyamatrendszert leiró objektumtipusok;
- az információs folyamatrendszer kezelését meghatározó objektumtipusok;
- az információs folyamatrendszer objektumai tulajdonságát meghatározó objektumtipusok.

Az egyes osztályokhoz tartozó objektumtipusokat részletesen is leirjuk, ezeket összefoglalva a 2.1. táblázatban láthatjuk.

2.1. táblázat

Az objektumtipusok osztályozása

Objektumtipusok osztályai	Objektumtipusok
Az információs folyamatrendszer környezetét leiró objektumtipus	INTERFACE /REAL-WORLD-ENTITY/
Az információs folyamatrendszert leiró objektumtipus	
Az információáramlást leiró objektumtipusok	INPUT OUTPUT ENTITY
Az információ tárolás objektumtipusa	SET
Az információ elemei között fennálló kapcsolatokat leiró objektumtipus	RELATION
Az adatok definiciójá szolgáló objektumtipusok	GROUP ELEMENT
Adatokkal végzett müveletek leirását szolgáló objektumtipus	PROCESS
Az információs rendszer méretét leiró objektumtipusok	SYSTEM-PARAMETER INTERVAL
Dinamikus viselkedést leiró objektumtipusok	EVENT CONDITION

/2.1. táblázat folytatása/

Információs folyamatrendszer kezelését meghatározó objektumtipusok	PROBLEM-DEFINER MAILBOX
Információs folyamatrendszer objektumai tulajdonságát meghatározó objektumtipusok	SYNONYM KEYWORD ATTRIBUTE ATTRIBUTE-VALUE MEMO SOURCE SECURITY

2.3.1 Az információs folyamatrendszer környezetét leiró objektumtipus

A rendszer környezetét képezik azok a rendszeren kivüli objektumok, amelyek vele, vagyis valamely objektumával kapcsolatban állnak. Ez a kapcsolat adatátvitel lehet.

Az információs folyamatrendszer környezetének objektumait INTERFACE /vagy REAL-WORLD-ENTITY/ objektumtipssal definiálhatjuk.

2.3.2 Az információs folyamatrendszert leiró objektumtipusok

A PSL használatának célja az információs rendszert meghatározni, leirni. Ezért az objektumtipusok zöme erre szolgál.

a/ Az információáramlást leiró objektumtipusok:
Az információs folyamatrendszer és környezete közötti információáramlást két objektumtipus irja le:

INPUT: a környezetből az információs rendszerbe;

OUTPUT: a rendszerből a környezetbe áramló adatokat jelenti.

Az információs rendszeren belül mozgó adatokat az ENTITY objektumtipus definiálja.
Az INPUT, OUTPUT és ENTITY által definiált objektum adat, adatstruktúra, vagy névvel ellátott adathalmaz lehet.

b/ Az információtárolás objektumtipusa a SET. Ez tulajdonképpen az információs rendszer egy file-ja. Igy például a belső adatkezelésben a SET egy rekordtipusa egy adatstruktúrát leiró ENTITY lehet. A köztük lévő összefüggések az adatstruktúrák leirásánál láthatók részletesebben.

c/ Az információ elemei között fennálló kapcsolatokat leiró objektumtipust a RELATION alapszó definiálja. Ez tulajdonképpen az ENTITY-k közötti logikai kapcsolatokat irja le.

d/ Adatok tipusát definiáló objektumtipusok:
Az INPUT, OUTPUT és ENTITY értéket tartalmazó egységei az ELEMENT és a GROUP.
ELEMENT az adatok legkisebb egysége, vagyis egyetlen adatot képvisel.
Ha több ELEMENT között a rendszerleirásban logikai kapcsolatot teremtünk, ezek GROUP-ot alkotnak. Egy adott GROUP ELEMENT-eken kivül más GROUP-okat is tartalmazhat, ezáltal épülnek fel az adatstruktúrák.

e/ Az adatokkal végzett müveletek leirását végző objektumtipus:
Az információs folyamatrendszer valamilyen adathalmazból más, új információt szolgáltató adathalmazt állit elő. Ezt a transzformációt a PROCESS tipusú objektum irja le. Mivel a PROCESS az információs rendszer legfőbb része, ennek kapcsolatai a legszerteágazóbbak.

f/ Az információs rendszer méretét leiró objektumtipusok:
Az objektumtipusoknak ez az osztálya tulajdonképpen két funkciót tartalmaz. Az egyik a SYSTEM-PARAMETER objektumtipushoz kapcsolódik, amely egy adott objektum mennyiségét, terjedelmét adja meg. Például megadja, hogy egy SET-ben hány rekord van, vagy egy ELEMENT hány adatnak a neve.
A másik funkciót az INTERVAL objektumtipus fejezi ki. Ez leirja, hogy az objektum használatának gyakoriságát /például: hetente, vagy kétnaponként, stb./

g/ A rendszer dinamikus viselkedését leiró objektumtipusok:
Ezek mutatják meg, az információs rendszer időbeni müködését. A EVENT-ként definiált objektum megadja, hogy egy folyamatot, mely esemény hatására kell kezdeni vagy befejezni. Ez az esemény például egy input adat megérkezése.
CONDITION egy logikai értéket tartalmazó objektum, ennek "true" vagy "false" értékétől függ egy adott folyamat időbeni lefolyása.

2.3.3 Az információs folyamatrendszer kezelését meghatározó objektumtipusok

Egy információs rendszert általában többen készitenek. A munkamegosztásnak célszerü olyannak lenni, hogy egy-egy rendszertervező meghatározott terület leirásáért legyen felelős. Ehhez a területhez tartozó objektumok leirása tartalmazhatja a tervezésért, karbantartásért felelős nevét, amelyet a PROBLEM-DEFINER objektumtipus definiál. A MAILBOX-ban lehet leirni a PROBLEM-DEFINER egyéb adatait, például cimét, telefonszámát, stb.

2.3.4 Az információs folyamatrendszer objektumai tulajdonságát meghatározó objektumtipusok

Az objektumoknak olyan speciális tulajdonságai is vannak, amelyeket az eddigi objektumtipusokkal nem lehet definiálni. Ezek a SYNONYM, KEYWORD, ATTRIBUTE, ATTRIBUTE-VALUE, MEMO, SOURCE és SECURITY objektumtipusok.
SYNONYM: Egy objektumnak több különböző nevet is adhatunk a PSL leirásban, hogy a több néven használt fogalmak minden módon elérhetők legyenek, vagy egy hosszabb névnek röviditését is hasz-

nálhassuk. A már definiált objektumhoz a SYNONYM objektumtipussal rendelhetünk újabb neveket.

KEYWORD: Logikai halmazok létrehozása, rendezések, szelektálások meghatározott adatok alapján történik. Ezeket az adatokat definiálja a KEYWORD objektumtipus.

ATTRIBUTE és ATTRIBUTE-VALUE: Ezek az objektumtipusok az objektumoknak olyan tulajdonságait irják le, amelyeket más objektumtipusban nem definiáltunk. Például ha egy 6 karakteres ELEMENT-nek irjuk le a hosszát, akkor az ATTRIBUTE: HOSSZ; az ATTRIBUTE-VALUE: 6.

MEMO: Több objektumhoz azonos tulajdonságokat is rendelhetünk. Ennek leirására szolgál a MEMO objektumtipus.

SOURCE: Az információs rendszer leirásában gyakran történik hivatkozás valamilyen külső információforrásra. PSL-ben erre ad lehetőséget a SOURCE objektumtipus.

SECURITY: Megjelöli azt a személyt, aki a rendszerdokumentáció adott részét /objektumát/ köteles átnézni, ellenőrizni.

2.4 Szekciók

Mint ahogy a III. részben már emlitettük, a PSL leirás szekciókra tagolódik, ahol a szekció szerkezete a következő:

Objektumot definiáló állitás;

Kapcsolatokat, tulajdonságokat leiró állitások

.
.
.

Az objektumot definiáló állitás kötelezően a szekció első állitása, a többi sorrendje tetszőleges. A szekciók tipusai megegyeznek az objektumot definiáló állitás első szavával, igy a következő szekciótipusok léteznek:

CONDITION
DEFINE
DESIGNATE
ELEMENT
ENTITY
EVENT
GROUP
INPUT
INTERFACE
INTERVAL
MEMO
OUTPUT
PROBLEM-DEFINER
PROCESS
RELATION
SET

A DEFINE szekcióban kell definiálni az

ATTRIBUTE
ATTRIBUTE-VALUE
KEYWORD
MAILBOX
SECURITY
SOURCE
SUBSETTING-CRITERION
SYSTEM-PARAMETER

objektumtipusokat, a DESIGNATE szekcióban pedig a

SYNONYM

objektumtipust.

A szekcióban előforduló összes többi állitás a definicióban szereplő objektumra vonatkozik, és ezt nem kell külön leirni. Lássunk erre egy példát:

Legyen egy ENTITY tipusú objektum, amelynek neve: ANYAG-ADATOK, ezt az információs rendszerben CIKKSZAM szerint kell rendezni. Ezt a következőként irjuk le:

```
ENTITY: ANYAG-ADATOK;
    KEYWORDS ARE: CIKKSZAM;
```

Itt az ANYAG-ADATOK és a CIKKSZAM is objektum, az állitások értelmezése pedig a fenti megfogalmazás segitségével érthetővé válik.

A PSL-ben minden objektumot definiálni kell, de ahol az objektum szerepe egyértelmüen kiderül valamelyik állitásból, ott nem szükséges. Például a fenti leirás mellett használható a következő is:

```
DEFINE: CIKKSZAM AS A KEYWORD;
    APPLIES TO ANYAG-ADATOK;
```

Látható, hogy a két szekció együttes leirása már átfedést jelent az adatbázisba új információ a másodikból nem kerül, de az is látható, hogy ha csak a második szekciót használjuk, akkor az ANYAG-ADATOK objektum tipusa nem meghatározott. Megjegyzem, hogy a PSL által kiadott listák egyike /a FORMATTED PROBLEM STATEMENT/ teljes leirást ad minden objektumról, redundáns módon, függetlenül attól, hogy mi az általunk bevitt PSL leirásban ezt megtettük, vagy sem. Ha egy kapcsolatot két objektum között redundánsan irunk le, akkor természetesen vigyázni kell, hogy ne tartalmazzon ellentmondást.

2.5 Állitások alapszavai

Az objektumok definiciója után szükség van az objektu-

mok közötti kapcsolatok, relációk leirására. Ezek a kapcsolatok nem lehetnek tetszőlegesek, az objektum funkciója meghatározza információs rendszerbeli szerepét, igy azt is, milyen tipusú más objektummal állhat kapcsolatban, és milyen módon. A kapcsolatok funkcióját legkönnyebben úgy érthetjük meg, ha az információs rendszert különböző aspektusból vizsgáljuk. Ezek a szempontok megegyeznek a rendszerleirással szemben támasztott általános követelményeknek. A főbb aspektusok a következők:

- a rendszer folyamatának leirása;
- a rendszer strukturájának leirása;
- adatstrukturák leirása;
- adatszármaztatás leirása;
- a rendszer terjedelmének, méretének leirása;
- a rendszer dinamikus viselkedésének leirása;
- a rendszer tulajdonságainak leirása;
- a rendszerterv kezelésének leirása;

Ezek részletesebb leirása magában foglalja az adott szempontok szerint vizsgált objektumokat, és azok kapcsolatait. Összefoglalva a 2.2 táblázatban láthatjuk ezeket.

Mivel a kapcsolatok két objektumot kötnek össze, szükséges a kapcsolat irányát is megadni. Attól függően, hogy melyik objektum szekciójában szerepel a kapcsolat, általában két, úgynevezett komplementer kapcsolat fejezheti ki a relációt. Például legyen A és B egy-egy PROCESS tipusú objektum, amelyek kapcsolata: B részhalmaza A-nak /strukturális kapcsolat/. Ennek leirása PSL-ben:

```
PROCESS: A;
     SUBPARTS: B;
```

A B objektum szekciójában:

```
PROCESS: B;
     PART OF A;
```

Ezek alapján a SUBPARTS és PART OF komplementer kapcsolatok, jelentésük azonos, csupán a kapcsolat irányát határozzák meg különböző formában. A következőkben megvizsgáljuk, mire használhatók a 2.2. táblázatban felsorolt relációk.

2.5.1 A rendszer folyamatának leirása

Ebből a szempontból az adatáramlást tudjuk leirni. Az adatokat használó objektumok: INTERFACE, PROCESS és SET, az adatmozgatást az INPUT és OUTPUT végzi. A folyamat során köztük fennálló kapcsolatokat szemléletesen mutatja a 2.1. ábra. A kapcsolat irányát mutató nyilak egyúttal a komplementer kapcsolat értelmezését is megadják. Ezek alapján a relációk a következőket jelentik:

GENERATES/GENERATED BY:
INTERFACE és Process közti adatmozgást irja le két közbenső objektum /INPUT és OUTPUT/ segitségével. Alkalmazása:

- INTERFACE szekcióban:
  GENERATES input-név;
- PROCESS szekcióban:
  GENERATES output-név;

Komplementer kapcsolatai:

- INPUT szekcióban:
  GENERATED BY interface-név;
- OUTPUT szekcióban:
  GENERATED BY process-név;

Mint látható, az INTERFACE és PROCESS által előállitott adatok áramlását irja le ez az állitás.

RECEIVES/RECEIVED BY:

2.2. táblázat

PSL objektumtipusok és kapcsolataik különböző aspektusból

Aspektus	Objektum-tipus	Kapcsolattipusok
A rendszer folyamatának leirása	INTERFACE INPUT OUTPUT PROCESS SET	RECEIVES/RECEIVED GENERATES/GENERATED UPDATES/UPDATED RESPONSIBLE FOR/ RESPONSIBLE-INTERFACE
A rendszer strukturájának leirása	INTERFACE INPUT OUTPUT PROCESS SET	SUBPARTS/PART OF SUBSET/SUBSETS UTILIZES/UTILIZED
Adatstrukturák leirása	GROUP ELEMENT ENTITY INPUT OUTPUT SET	CONSISTS/CONTAINED RELATED/BETWEEN IDENTIFIES/IDENTIFIED ASSOCIATED/ASSOCIATED-DATA SUBSETTING-CRITERIA/ SUBSETTING-CRITERION VALUE

/2.2. táblázat folytatása/

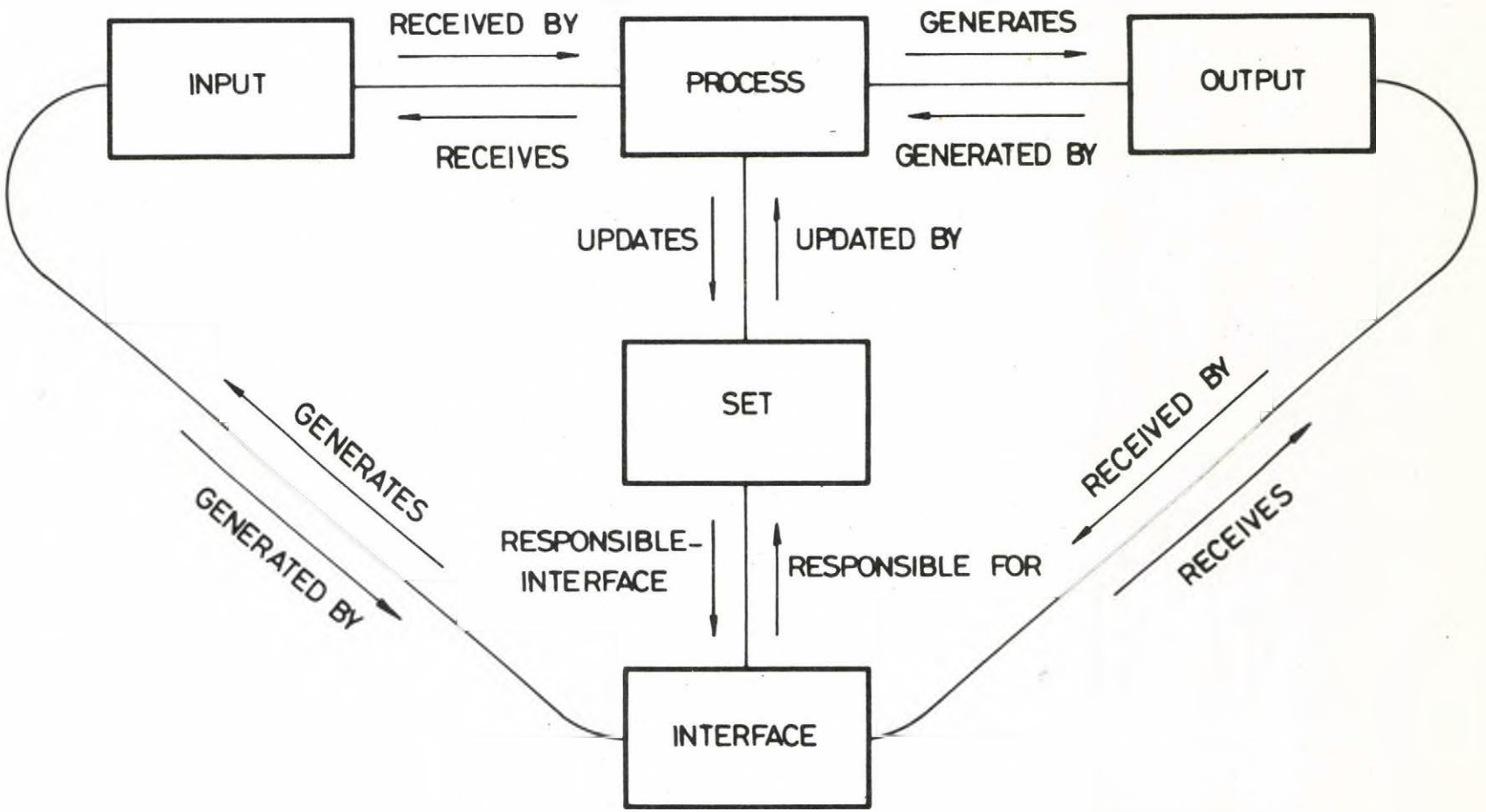
Adatszármaztatás leírása	GROUP ELEMENT ENTITY INPUT OUTPUT PROCESS SET RELATION	USES/USED UPDATES/UPDATED DERIVES/DERIVED MAINTAINS/MAINTAINED PROCEDURE* DERIVATION*
A rendszer terjedelmének leírása	SYSTEM- PARAMETER INTERVAL	VALUE CARDINALITY CONNECTIVITY HAPPENS CONSISTS VOLATILITY* VOLATILITY-SET* VOLATILITY-MEMBER*
A rendszer dinamikus viselkedésének leírása	EVENT CONDITION	BECOMING/WHEN INCEPTION-CAUSES/ ON INCEPTION TERMINATION-CAUSES/ ON TERMINATION TRIGGERS/TRIGGERED WHILE*

*Ezeknek a kapcsolattipusoknak a tartalma elbeszélő leirás.

/2.2. táblázat folytatása/

A rendszer tulajdonságainak leirása	ATTRIBUTE ATTRIBUTE-VALUE KEYWORD MEMO SYNONYM SOURCE SECURITY	SYNONYMS/DESIGNATE ATTRIBUTES/APPLIES KEYWORDS/APPLIES SEE-MEMO/APPLIES SOURCE/APPLIES SECURITY/APPLIES DESCRIPTION*
A rendszerterv kezelésének leirása	PROBLEM-DEFINER MAILBOX	RESPONSIBLE/RESPONSIBLE-PROBLEM-DEFINER MAILBOX/APPLIES

* Ennek a kapcsolattipusnak a tartalma elbeszélő leirás.



2.1 ábra

Az információs rendszer folyamatának leirása

RECEIVES/RECEIVED BY:

Ugyancsak az INTERFACE és PROCESS közti adatmozgást irja le ez a kapcsolat, de e két objektum bemeneti adataival. Alkalmazása:

- INTERFACE szekcióban:
  RECEIVES output-név;
- PROCESS szekcióban:
  RECEIVES input-név;

Komplementer kapcsolatai:

- INPUT szekcióban:
  RECEIVED BY process-név;
- OUTPUT szekcióban:
  RECEIVED BY interface-név;

UPDATES/UPDATED BY:

A SET kezelését a PROCESS objektum végzi, amelyet az UPDATES reláció jelöl:

- PROCESS szekcióban:
  UPDATES set-név;
- SET szekcióban:
  UPDATED BY process-név;

RESPONSIBLE FOR/RESPONSIBLE-INTERFACE:

Ez a reláció mutatja azt az INTERFACE objektumot, amely a SET karbantartásáért felelős.

- INTERFACE szekcióban:
  RESPONSIBLE FOR set-név;
- SET szekcióban:
  RESPONSIBLE-INTERFACE interface-név;

A továbbiakban nem részletezzük a kapcsolatok irásmódját, a fentiekből ez megérthető, a komplementer kapcsolatok használata is világossá vált.

2.5.2 A rendszer strukturájának leirása

A rendszerleirásban szereplő objektumok egy része strukturális kapcsolatban állhat más, objektumokkal. PSL-ben kétféle struktura engedhető meg /lásd 2.2. ábra/:

a/ fastruktura;
b/ hálóstruktura.

A fastruktura értelmezése egyértelmü, ezt részletezni nem szükséges. Hálóstrukturával szemben egy követelményt kell állitani: a struktura ne tartalmazzon ciklust, mert ellenkező esetben a struktura felépitése nem lesz egyértelmü.

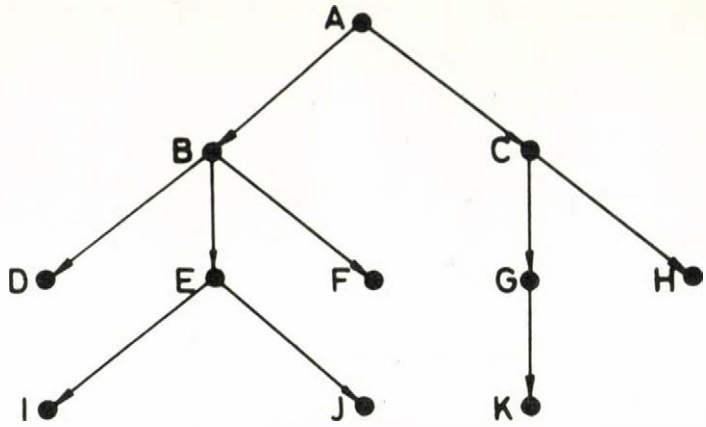
Ez a két strukturatipus mind a rendszerstrukturák, mind az adatstrukturák felépitésében használható. A rendszerstrukturát felépitő objektumok: INTERFACE, INPUT, OUTPUT, PROCESS és SET, de ezek mindig homogén strukturák, vagyis csak azonos tipusú objektumok szerepelhetnek bennük.

SUBPARTS ARE/PART OF:
Ez a reláció csak fastruktura felépitésére használható. Segitségével INTERFACE, INPUT, OUTPUT és PROCESS strukturáit lehet felépiteni. Például egy INPUT szekcióban:

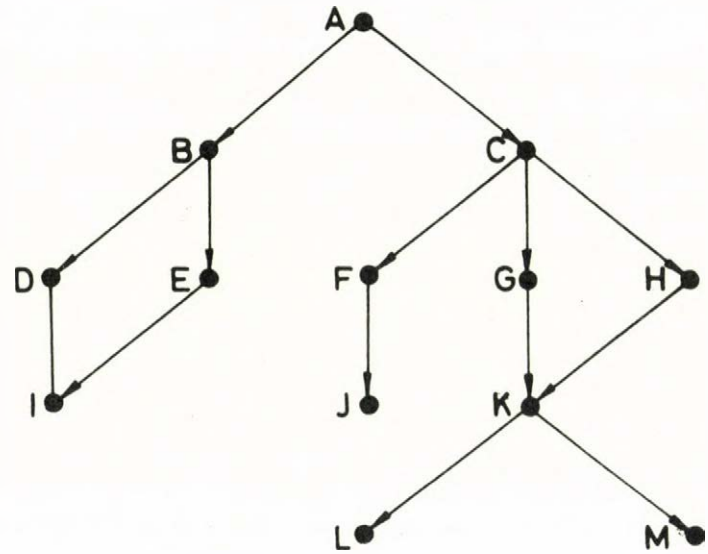
SUBPARTS ARE input-név;

által megjelölt INPUT tipusú objektum a strukturában alacsonyabb szinten áll, mint a szekció első állitásában megnevezett objektum. A PART OF komplementer kapcsolat forditott irányban jelöli meg a strukturális kapcsolatot.

SUBSETS ARE/SUBSET OF:
Kizárólag SET tipusú objektumok közti strukturális kapcsolatot épithetünk fel ezzel a relációval. Különbség a SUBPARTS relációtól, hogy ezzel hálóstruktura felépitése is lehetővé válik.



Fastruktura



Hálóstruktura

2.2 ábra

STRUKTURÁK

UTILIZES/UTILIZED BY:
PROCESS tipusú objektumokkal is szükségessé válik hálóstruktura kialakitása /például számitógépes információs rendszerekben szubrutinok használatakor/, amelyet az UTILIZES relációval irhatunk le.

2.5.3 Adatstrukturák leirása:

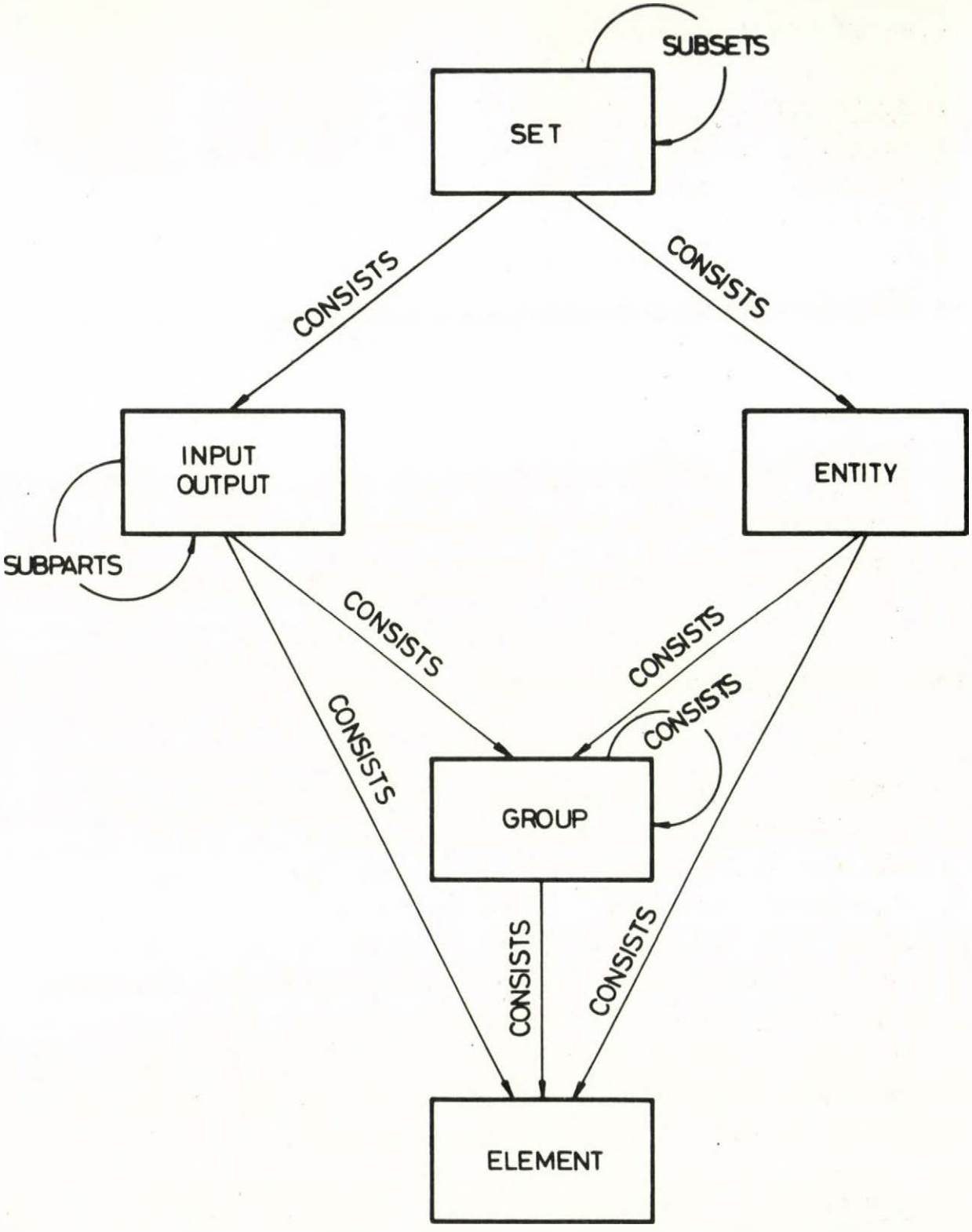
Mint az előző részben is látható volt, a strukturális kapcsolatokat a PSL-ben tulajdonképpen mint objektumok közti kapcsolatokat specifikáljuk.
Az adatstrukturákat alkotó objektumok:

ELEMENT, GROUP, ENTITY, INPUT, OUTPUT és SET.

Strukturájuk két csoportba osztható:

a/ Adathalmazok felosztása az elemi adatok szintjéig:
ezt irhatjuk le a
CONSISTS OF/CONTINED IN
kapcsolattal. Az igy felépithető adatstrukturát mutatja a 2.3. ábra. Ezen látható, hogy bizonyos mértékig ide is sorolható SUBSETS és SUBPARTS kapcsolat is. Az ábrából leolvasható, hogy pl. egy GROUP tipusú objektum alacsonyabb szintü adatai GROUP és ELEMENT tipusú objektumok lehetnek.

b/ Adatok és adathalmazok közti relációk:
RELATED TO... VIA/BETWEEN:
Két ENTITY között teremti meg valamely RELATION objektum által meghatározott kapcsolatot. Ennek irásmódja némileg eltér a szokásostól:
ENTITY szekcióban:
RELATED TO másik entity-név VIA relation-név;
RELATION szekcióban:
BETWEEN entity-név-1 AND entity-név-2;


SUBSETS
SET
CONSISTS
CONSISTS
INPUT
OUTPUT
ENTITY
SUBPARTS
CONSISTS
CONSISTS
CONSISTS
CONSISTS
GROUP
CONSISTS
CONSISTS
ELEMENT

IDENTIFIED BY/IDENTIFIES:
ENTITY tipusú objektumhoz tartozó adatok közül megjelöli azt a GROUP és/vagy ELEMENT tipusú objektumokat, amelyek az ENTITY-t azonositják.

ASSOCIATED-DATA/ASSOCIATED WITH:
Megjelöli azokat a GROUP és/vagy ELEMENT tipusú objektumokat, amelyekhez valamely adott RELATION tipusú objektum tartozik.

SUBSETTING-CRITERIA/SUBSETTING-CRITERION:
A SET-hez rendeli azt a GROUP-ot vagy ELEMENT-et, amely meghatározza a létrehozandó subset-eket.

VALUE IS:
ELEMENT tipusú objektumoknak lehet értéket adni ezzel a relációval. Ez az érték akár numerikus, akár alfanumerikus lehet.

2.5.4 Adatszármaztatás leirása

Az információs folyamatrendszer alapvető feladata az adatkezelés, vagyis meghatározott módon az input adatokból előállitja az output adatokat. Ezt a transzformációt a PSL megfogalmazás szerint a PROCESS tipusú objektumok végzik el. Az adatszármaztatás relációi megadják, hogy a PROCESS tipusú objektum honnan kapja az adatokat, és a transzformáció után hova kerülnek az előállitott új adatok. Olyan kapcsolatpárokat is lehet irni, amelyek egy állitáson belül megmutatják, az adatáramlás útját.
Nézzük először az egyszerü kapcsolatokat:

USES/USED:
Azokat az objektumokat köti össze a PROCESS-szel, amelyektől a PROCESS adatokat kap. Ezek tipusai: SET,

INPUT, ENTITY, GROUP, ELEMENT.

UPDATES/UPDATED:
SET, ENTITY, GROUP és ELEMENT tipusú objektumokkal történő adatcserét irja le ez a kapcsolat.

DERIVES/DERIVED:
A PROCESS által elkészitett adatok átadását SET, OUTPUT, ENTITY, GROUP és ELEMENT tipusú objektumok részére a DERIVES reláció jelöli ki.

MAINTAINS/MAINTAINED:
A PROCESS tipusú objektum által kezelt RELATION objektumot a MAINTAINS reláció kapcsolja a PROCESS-hez.

Az összetett kapcsolatok nem csupán a PROCESS tipusú objektumhoz jelölik az adatáramlás útját, hanem a PROCESS be- és kimenő adatai között is megteremtik a kapcsolatot. A PROCESS szekcióban ez a következő formákban irható le:

```
USES b TO DERIVE c;
USES b TO UPDATE a;
DERIVES c USING b;
UPDATES a USING b;
```

ahol a kisbetük jelentése:

a: SET, ENTITY, GROUP, ELEMENT név,
b: SET, ELEMENT, ENTITY, GROUP, INPUT név,
c: SET, ENTITY, GROUP, ELEMENT, OUTPUT név.

2.5.5 A rendszer terjedelmének leirása

Amint az objektumtipusok ismertetésénél már leirtuk, az információs rendszerben szükség van olyan paraméteres állitásokra, amelyek meghatározzák a rendszer méretét,

PSL-ben ezt a célt szolgálja a SYSTEM-PARAMETER és INTERVAL objektumtipus. Ezek kapcsolatát más objektumokkal a VALUE, CARDINALITY, CONNECTIVITY, HAPPENS és CONSISTS állitások adják meg.

VALUE:

A DEFINE szekcióban definiált SYSTEM-PARAMETER-nek lehet értéket adni a VALUES állitással. Ez lehet egyetlen numerikus érték:
VALUE numerikus-érték;
egy értékintervallum:
VALUES minimális érték THRU maximális érték;
vagy értékintervallumként kijelölhetjük az egész számtartományt
VALUES NEGINF THRU POSINF;

CARDINALITY:

Egy SYSTEM-PARAMETER-t rendel valamelyik ENTITY, RELATION vagy SET tipusú objektumhoz. Jelentése:
ENTITY szekcióban: megadja az adott ENTITY egyidőben létező maximális számát.
RELATION szekcióban: az adott RELATION-hoz tartozó objektumok maximális számát adja meg.
SET szekcióban: meghatározza a SET rekordjainak maximális mennyiségét.

CONNECTIVITY:

RELATION szekcióban szerepel a CONNECTIVITY állitás, amellyel a RELATION-nal összekapcsolt két ENTITY együttes számát határozhatjuk meg. Szintaktikusan:

CONNECTIVITY IS system-parameter-1 TO system-parameter-2;

HAPPENS... TIMES-PER:

Ezzel az állitással adhatjuk meg, hogy egy adott EVENT, INPUT, OUTPUT vagy PROCESS tipusú objektumot egy adott

időintervallumban /INTERVAL/ hányszor használunk. Az állitás az EVENT, INPUT, OUTPUT vagy PROCESS szekcióban a következő formájú:

HAPPENS system-parameter TIMES-PER interval-név;

CONSISTS OF:
SET, INPUT, OUTPUT, GROUP szekciókban már ismerjük a CONSISTS OF állitás szerepét. Ezt az állitást kibővithetjük egy SYSTEM-PARAMETER-rel, amely pl. megadja egy SET-hez tartozó, adott nevü ENTITY-k számát. A teljes CONSISTS OF állitás pl. egy SET szekcióban:

CONSISTS OF system-parameter entity-név;
A system-parameter használata nem kötelező.
CONSISTS OF állitás INTERVAL szekcióban is használható a következő formában:

CONSISTS OF system-parameter interval-név;
Itt a szerepe: a szekció első állitásában /definicióban/ megadott időintervallum kisebb egységeit lehet megadni. Ha például a szekcióban "év"-et definiáltunk, akkor a CONSISTS OF állitásba, mint kisebb egységet, a "hónap"-ot irhatjuk. A system-parameter megadja, hogy a nagyobb egységü időintervallumban hány kisebb INTERVAL van.

2.5.6 Az információs rendszer dinamikus viselkedésének leirása

Az eddigi kapcsolatok, állitások az információs rendszer időben állandó, statikus állapotát irták le, de szükség van időtől függő események szemléltetésére is. A rendszer dinamikus viselkedését bizonyos feltételek vagy feltétel sorozatok szabályozzák, ezek teljesülésétől függ egyes folyamatok végrehajtása. A dinamikus viselkedéshez sorolhatjuk részben a HAPPENS állitás hatását is, mert időtől függő eseményt határoz meg, de az idő-

függést ebben az esetben semmilyen feltétel nem szabályozza. A dinamikus viselkedést meghatározó két objektumtipusnak /CONDITION, EVENT/ meghatározott kapcsolata van. A CONDITION meghatározza azt a feltételt, amelynek TRUE vagy FALSE értékétől függő EVENT /esemény/ végrehajt valamilyen folyamatot.

BECOMING... CALLED/WHEN... BECOMES

Ennek az állitásnak hatására CONDITION értékétől függ, hogy melyik EVENT megvalósitása következik.
CONDITION szekcióban leirt állitások:

BECOMING TRUE CALLED event-név;
BECOMING FALSE CALLED event-név;

Ennek komplementer állitása az EVENT szekcióban:

WHEN condition-név BECOMES TRUE;
WHEN condition-név BECOMES FALSE;

EVENT és PROCESS közti kapcsolatok:

ON INCEPTION/INCEPTION-CAUSES:

Egy HAPPENS meghatározott időszakonként végrehajtandó folyamat kezdő PROCESS tipusú objektumát határozza meg, ez a kapcsolat. EVENT szekcióban az ON INCEPTION, komplementer párja pedig a PROCESS szekcióban szerepel.

ON TERMINATION/TERMINATION-CAUSES:

Egy HAPPENS állitásban meghatározott időszakonként végrehajtandó folyamat befejező PROCESS tipusú objektumát határozza meg.

TRIGGERS/TRIGGERED:

Az EVENT hatására induló folyamatokat lehet a TRIGGERS állitással meghatározni.

2.5.7 Az információs rendszer tulajdonságainak leirása

Ide sorolhatjuk azokat a kapcsolatokat, amelyek minden objektumban előfordulhatnak. Ezek a következők:

SYNONYMS:
Egy objektumnak több nevet, szinonimát adhatunk. Abban a szekcióban, amelyben az objektumot definiáltuk,

SYNONYMS szinonimák listája;

formában irhatjuk le az összetartozó neveket. A SYNONYM-ot, mint objektumtipust a DESIGNATE szekcióban definiáljuk, amelyet a SYNONYMS állitás komplementer párjának tekinthetünk.

ATTRIBUTES:
Az ATTRIBUTES állitás az objektumoknak olyan jellemzőit, tulajdonságait irja le, amelyet más állitással nem tudunk megfogalmazni. Ebben az állitásban az attributumnak értéket is lehet adni a következőképpen:

ATTRIBUTES attribute-név attribute-value-név;

Ennek az állitásnak komplementer párja a DEFINE szekcióban leirt ATTRIBUTE és ATTRIBUTE-VALUE objektum definició.

KEYWORDS/APPLIES TO:
Az objektumoknak azt a részét irja le ez az állitás, amely alapján rendezni, osztályozni lehet az objektum elemeit. Az objektumok szekcióiban a KEYWORDS állitást használjuk, komplementer állitásként a DEFINE szekcióban definiált KEYWORDS objektumban az APPLIES TO-t irhatjuk.

SEE-MEMO/APPLIES TO:
Ha egy tulajdonságot több objektumhoz is rendelhetünk, ezt a MEMO szekcióban irjuk le, és a kapcsolatot az ob-

jektumokkal az APPLIES TO állitás teremti meg. Az adott objektumban a MEMO tipusú objektum nevét a SEE-MEMO állitással adjuk meg.

SOURCE/APPLIES TO:
Az információs rendszeren kivüli hivatkozást teszi meg a SOURCE állitás, komplementere a DEFINE szekcióban irható le.

SECURITY/APPLIES TO:
Azt a személyt irhatjuk le ezzel az állitással, aki a dokumentáció adott részét, objektumát köteles átnézni, ellenőrizni. A DEFINE szekcióban használjuk az APPLIES állitást.

2.5.8 A rendszerterv kezelésének leirása:

Kétféle kapcsolat tartozik ide:

RESPONSIBLE-PROBLEM-DEFINER/RESPONSIBLE FOR:
Bármely objektum /kivéve: MAILBOX és PROBLEM-DEFINER/ és a PROBLEM-DEFINER között teremti meg a kapcsolatot. Az objektumnak megfelelő szekcióban használjuk a RESPONSIBLE-PROBLEM-DEFINER állitást, komplementer párját pedig a PROBLEM-DEFINER szekcióban.

MAILBOX IS/APPLIES TO:
PROBLEM-DEFINER-hez rendeli a hozzátartozó MAILBOX objektumtipust. A DEFINE szekcióban irjuk az APPLIES TO állitást.

A fenti nyolc pontban az objektumok között kapcsolatot teremtő állitások szerepeltek, de ide tartoznak azok az alapszavak is, amelyek nem objektumneveket vagy értéket rendelnek a szekcióban definiált objektumhoz, hanem el-

beszélő leirást. Mivel ezek a leirások a felhasználó részére adnak bővebb tájékoztatást, különválasztva tárgyaljuk.

DESCRIPTION:
Minden objektumhoz irható, kivéve a SYNONYM objektumtipust a DESIGNATE szekcióban. Arra szolgál, hogy körülirják a definiált objektumot, közérthető formában leirjuk annak tulajdonságait.

PROCEDURE:
A PROCESS szekcióban használható, ezen belül irjuk le részletesen azt a folyamatot, amelyet az adott PROCESS tipusú objektum végez.

DERIVATION:
Két szekcióban is szerepelhet:
RELATION szekcióban szöveges formában megfogalmazhatjuk a két ENTITY közti RELATION tipusú kapcsolat lényegét, célját, stb. SET szekcióban a DERIVATION elbeszélő leirás célja az adott SET adatainak eredetét, az adatáramlás útját megfogalmazni, vagy azt, hogy egy-egy ELEMENT honnan kap értékeket, stb.

VOLATILITY:
ENTITY szekcióban használható; az ENTITY értékeinek időbeni változásait irhatjuk le szöveges formában. Megadhatjuk a változás módját, eredetét, stb.

VOLATILITY-SET és VOLATILITY-MEMBER:
Mindkét állitás a SET szekcióban irja le a változásokat, hasonlóképpen, mint a VOLATILITY állitás az ENTITY szekcióban. A cimben szereplő két állitás közti különbség: a VOLATILITY-SET a teljes SET változását, a VOLATILITY-MEMBER pedig a SET egy részének, egy vagy több elemének változását fogalmazza meg.

WHILE:
CONDITION szekcióban irhatjuk le a WHILE állitás segitségével, hogy milyen feltételek mellett, milyen szituációban lesz a CONDITION tipusú objektum értéke TRUE vagy FALSE. Ennek megfelelően két tipusa van ennek az állitásnak:

TRUE WHILE és FALSE WHILE.

2.6 Kiegészitő szavak

A kiegészitő szavak használata nem kötelező, ezek teszik a PSL szövegeket angol nyelven folyamatosan olvashatóvá. Ezek kötőszavak, elöljárók, mutatószavak, névelők, stb. lehetnek.

Külön magyarázat nélkül nézzük a használható kiegészitő szavak listáját:
A, AN, AND, ARE, AS, BEING, BY, FOR, FROM, IN, IS, IT, OF, ON, THE, THIS, TO, WHETHER, WITH.

Ezek egy részének használatával már megismerkedhettünk az állitások ismertetésénél.

2.7 Összefoglaló

A PSL információs rendszerleiró nyelv, amelynek segitségével számitógépes úton végezhetjük el az információs rendszerek logikai tervezését és karbantartását.

A nyelv strukturájában a legnagyobb egységek a szekciók, amelyek állitásokból állnak. Egy szekció első állitása egy objektumot definiál, a többi állitás pedig leirja a definiált objektum más objektumokkal fennálló kapcsolatait. Az igy készült leirást könnyebben érthetővé a

szöveges megfogalmazások, és a használható kiegészitő szavak teszik. Legtöbb PSL alapszónak röviditése is van, amelyet eddig nem emlitettünk, de a most következő táblázatban, amelyben szekciótipusonként foglaljuk össze a lehetséges állitástipusokat.

3. A PROBLEM STATMENT ANALYSER /PSA/

3.1 A PSA rendszer, mint a logikai rendszertervezés segédeszköze

A PSA software rendszert az Information Systems Design and Optimization System /ISDOS/ project fejlesztette ki a michigani egyetemen. Az elemző célja kettős:

1. Eszközt szolgáltasson azon igények leirásának karbantartására, melyek egy információs rendszerre vonatkoznak
2. ezekről az igényekről elemzéseket és riportokat szolgáltasson.

A PSA a logikai rendszertervezés fázisában használható az igények feldolgozására, amikor a kérdés az, hogy a rendszernek "mit" kell csinálni, nem pedig, hogy "hogyan". A PSA tehát nem végrehajtható object /tárgy/ kódot hoz létre, hanem egy dokumentációt, amely azután a fizikai rendszertervezés folyamán eredményesen használható a létrehozandó object kód minőségének javitására.

A PSA fő inputja a követelmények PSL nyelvü leirása. A PSA elemzi az inputot /azaz a PSL nyelven leirt követelményeket/ abból a szempontból, hogy konzisztensek-e a már előzőleg megadottakkal és ezután az addigi összes követelmények adatbázisához csatolja. Ezt az adatbázist PSA adatbázisnak nevezik. A PSA adatbázis bármely időpontban a célul kitüzött rendszerre vonatkozó összes addig ismert követelmény leirását tartalmazza. Lehetőség van természetesen egy már a PSA adatbázisban tárolt probléma leirás módositására is. Az ilyen megközelités nagy előnye az, hogy a követelmények konzisztens és jól definiált módon vannak leirva a PSL útján, továbbá, hogy a már ismert követelmények egy

adatbázisban tárolódnak, és a probléma kezelőjének csupán az újonnan felfedett vagy módositó információt kell csatolnia ehhez. A probléma kifejlődésének bármely szakában a probléma kezelője az összegző és elemző riportok különféle formáit igényelheti, melyek az adatbázisbeli probléma leirás akkori állapotán alapulnak. A PSA-nak vannak olyan riportjai is, amelyek különféle embercsoportok igényeinek megfelelően orientáltak, akiknek szintén tudniuk kell az adatbázisbeli probléma leirás állapotáról, mint pl. a célul kitüzött rendszer felhasználói, a fizikai rendszer tervezői, a probléma meghatározás vezetői.

3.2 A PSA rendszer használata

A későbbiekben elmondásra kerülő részletek könnyebb érthetősége és egymáshoz kapcsolódásuk könnyebb áttekinthetősége miatt szóljunk néhány szót a PSA használatáról a felhasználó szempontjából. Ez nagyvonalakban a következő lépéseket jelenti. A felhasználó létrehoz egy adatbázis file-t és "inicializálja" azt. Ez a lépés elmarad, ha már létezik az adatbázis file, amivel dolgozni akar. Ezután a felhasználó hozzáfér a PSA rendszerhez. Az eddigi lépéseket az cperációs rendszer hajtja végre megfelelő parancsok hatására. Amint a felhasználó hozzáfért a PSA software rendszerhez, úgy mondjuk "PSA módba" került. Ekkor rendelkezésre áll egy parancs nyelv - melyről a későbbiekben kissé bővebben szó lesz - mellyel irányitja a PSA rendszer tevékenységét saját igényeinek megfelelően. Parancsokat ad ki a PSL nyelven leirt problémája bevitelére az adatbázisba, riportokat generál az adatbázis tartalmáról stb. Végül a STOP parancs hatására kikerül a PSA módból és a vezérlést az operációs rendszer veszi át.

A PSA software rendszer mind "batch", mind "on-line" feldolgozási módban használható. Igazán hatékonyan interaktiv környezetben müködik, ahol a probléma kezelője a rendszert on-line terminálon keresztül közvetlenül vezérli. Ha hibás parancsot adott rögtön javithat, riportokat igényelhet, a riportok alapján azonnal módositásokat tehet.

3.3 A PSA rendszer felépitése

A PSA software rendszer egymáshoz kapcsolódó modulok összessége, amelyek a következők: egy parancs nyelv közvetitő /CLI, Command Language Interface/ modul, kb. 20 parancs feldolgozó modul, egy modul a név szelektálásra, egy adat-báziskezelő rendszer, és többféle különálló hasznositás. A software-t, amennyire lehetséges volt gép-függetlenné szándékozták tenni. Ennek eléréséhez a software-t, mely az adatbáziskezelő rendszert is tartalmazza majdnem egészében FORTRAN IV nyelven irták. A majdnem azt jelenti, hogy az implementáláskor kb. 30 alapvető rutint a gép assembly nyelvén /esetünkben COMPASS nyelven/ kellett megirni.

A CLI modul a probléma kezelő parancsait interpretálja és meginditja a megfelelő parancsfeldolgozó modul végrehajtását. A parancsfeldolgozó modulok kétféle kategóriájuak: az adatbázist update-oló modulok és a riport generáló modulok. A parancsfeldolgozó modulok az adatbáziskezelő rendszeren keresztül érintkeznek a PSA adatbázissal. Ez a probléma leirásban tárolt információ kényelmes reprezentálását teszi lehetővé. A név szelektáló modul az adatbázis egy részhalmazának kiemelésére szolgál, melyet azután különféle riport generáló modulok hasznositanak. A különálló hasznositásoknak nevezett modulok bizonyos periférikus tevé-

kenységeket végeznek, úgy mint pl. az adatbázis inicializálása, dumpolása, vagy újra tárolása.

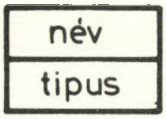
3.4 A PSA adatbázis

A PSA adatbázis mindig a probléma leirás teljes és naprakész verzióját tartalmazza. A célul kitüzött rendszerre vonatkozó PSL leirás darabonként és bármilyen sorrendben bevihető az adatbázisba. A PSA adatbázisban tárolt információ módositható. Háromféle módositás végezhető: módosithatók maguk az objektumok, az objektumok között lévő kapcsolatok, és az adatbázisban tárolt szövegszerü információ. Valahányszor riportot generálunk a PSA adatbázisból, az összes információ naprakész, bármely módositás figyelembe van véve. A PSA adatbázisban az információ három alapvető tipusa van tárolva:

1. az objektumok nevei és tipusai
2. a comment bejegyzések /az objektumok szabad formájú, narrativ leirása/
3. az objektumok közötti ill. egy objektum és comment bejegyzés közötti kapcsolatok.

Az összes információ PSL utasitásokból, az INPUT-PSL parancs hatására kerül az adatbázisba. A legtöbb esetben az objektumok nevét és tipusát a "fejezet nyitó utasitások" /section header statment/ definiálják, a comment bejegyzéseket a "comment bejegyzés utasitások", a kapcsolatokat pedig más PSL utasitások.

A három információ tipusnak alapvetően három rekord tipus felel meg az adatbázisban. Hogy grafikusan is illusztrálhassuk az adatbázis struktúráját, használjuk a következő szimbólumokat e három rekordtipusra:



név rekord; egy adott, elnevezett objektum leirására szolgáló rekord

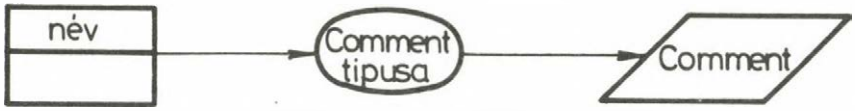
comment bejegyzés rekord; az adatbázisban lévő comment bejegyzésekre szolgáló rekord

NUB rekord; az objektumok közötti ill. egy objektum és egy comment bejegyzés közötti kapcsolat létesitésére szolgál.

E jelöléseket használva, egy egyszerü kapcsolat két objektum /név rekord/ között a következőképpen nézhet ki:

A NUB rekordban lévő adatok a kapcsolat tipusát definiálják a nyilak pedig azt a módot, ahogy a kapcsolatot interpretálni kell.

Egy objektum és egy comment bejegyzés közötti kapcsolat ilyen lehet:



A NUB-beli adatok a comment bejegyzés tipusát definiálják.

3.5 A PSA elemző képessége

A PSA kétféle elemzést végez. Az egyik az információ bevitele folyamán történő elemzés a másik a riportok útján történő elemzés. Előbbi a PSL nyelvü leirás szintaktikus és szemantikus elemzését, valamint konzisztencia elemzést jelent. A szintaktikus elemzés a korrekt PSL szintaktika betartását ellenőrzi. A szintaktikai

szabályok áthágása hibaüzenetet okoz. Pl. ha 30 karakternél hosszabb nevet használunk a következő hibaüzenetet kapjuk:

PSA002 NLEX: NAME TOO LONG

A szemantikus elemzés az utasitások korrekt használatát és a kétértelmüségek kiküszöbölését szolgálja. Ha pl. egy adott név tipusaként előzőleg PROCESS-t jelöltünk meg, akkor e nevet nem használhatjuk GROUP-ként. Vagy pl. két PROCESS név nem kapcsolható össze a USES kapcsolattal. Ha mégis megkisérelyjük a PSA a

MUST BE ELEMENT, ...

hibaüzenetet generálja.

A konzisztencia elemzés azt ellenőrzi, hogy a beviendő információ összhangban van-e az adatbázisban levővel. Megvizsgálja, hogy az inputon definiált kapcsolatok nincsenek-e ellentomondásban a már meglevőekkel. Ha pl. a következő információ már az adatbázisban van

ENTITY XYZ;
  CARDINALITY IS 100;

és megkisérelyjük az XYZ-re vonatkozóan a CARDINALITY IS 50 információt bevinni, akkor a

  CARDINALITY ALREDY GIVEN AS DIFFERENT VALUE

hibaüzenet generálódik.

Az input feldolgozása folyamán történő most elmondott ellenőrzések állandóan szintaktikusan helyes, korrekt és konzisztens állapotban tartják az adatbázis tartalmát. Az input-információ feldolgozásakor nem történik teljességi ellenőrzés, mivel a feltevés az, hogy a probléma leirás darabonként kerül az adatbázisba, tehát a hiányzó információ a későbbiekben csatolva lesz.

A másik fajta elemzés riportok útján történik. Két fajtája a teljességi és összegzési elemzés. Mivel az input feldolgozás - mint éppen az imént emlitettük - bármely stádiumban megengedi a hiányos probléma leirást a riportok azok, melyek a probléma leirás bármely potenciális vagy aktuális hiányosságát alkalmas helyen megjegyzik.

Például ha a probléma leirásban szerepel egy olyan PROCESS, amelynek nincs inputja, akkor a PSA a DATA PROCESS REPORT-ban felhivja figyelmüket erre a hiányosságra: DERIVES SOMETHING, BUT DOES'NT USE ANYTHING üzenettel. Vagy pl. ha hiányos adatstrukturát definiáltunk, akkor a CONSIST COMPARISON REPORT-ban a NAME DOES'NT CONSIST OF ANYTHING figyelmeztetést kaphatjuk.

A másik fajta elemzést, melyet összegzési elemzésnek neveztünk, tulajdonképpen nem a PSA, hanem a probléma kezelője végzi. Itt ugyanis olyan hibákról, inkonzisztenciákról, redundanciákról van szó, amelyet a PSA nem képes észlelni. Pl. ugyanazt a dolgot két különböző névvel is definiáltuk, akkor ezek a PSA szemszögéből különböző objektumok lesznek. Mégis a PSA-ban rendelkezésre álló összegzési és összehasonlitó riportok igen hasznos segitőeszközt nyújtanak az ilyen hibák felfedéséhez is. Pl. ha egy bérelszámolási rendszerben ugyanazt az egységet /ENTITY/ két különböző néven is definiáltuk, pl. ALKALMAZOTT-REKORD és ALKALMAZOTT-ADATAI akkor e redundancia felfedezésére hasznos segitséget nyújt a CONTENTS REPORT és a DATA PROCESS REPORT.

Az előbbiből, amely az adatstrukturát jeleniti meg észrevehetjük, hogy két azonos strukturájú ENTITY-nk van. Az utóbbiból pedig, hogy - legalábbis remélhetően - mindkét ENTITY-t ugyanaz a PROCESS használja ill.

származtatja, ugyanolyan output ill. input származtatásával ill. használatával. Ezen észrevételek birtokában pedig mindenesetre gyanunk támadhat arra, hogy redundáns objektumokat definiáltunk.

A PSA 2.0-ás verziója 1974. júliusa óta müködik a michigani egyetem IBM 360 gépén az OS operációs rendszer vezérlése alatt. Egy korábbi verzió már 1971 óta implementálva van. A rendszert más gépekre is implementálták: UNIVAC 1100-asra az EXEC 8 operációs rendszer és CDC 6000-re a MACE operációs rendszer vezérlése alatt. 1976. július óta a rendszer az MTA SZTAKI CDC 3300-as gépén is müködik a MASTER operációs rendszer vezérlése alatt. A michigani egyetemen további munkák is folynak a PSA rendszer fejlesztésére.

4. A PSA PARANCS NYELV. A PSA OUTPUTJAI ÉS AZ ADATBÁZIS KEZELŐ RENDSZER

A továbbiakban három részt ismertetünk bővebben, melyet a felhasználó szempontjainak megfelelően választottunk ki. Az első rész a PSA parancs nyelvvel foglalkozik, a második a PSA outputokkal, a harmadik pedig az adatbázis kezelő rendszerrel. Az utolsó rész a PSL/PSA rendszer felhasználóját nem kell érdekelje, mivel a PSL/PSA rendszer az adatbázis kezelő rendszert automatikusan használja, mégis azért emeltük ki, mert a PSL/PSA rendszertől függetlenül, önálló felhasználásra is számot tarthat.

4.1 A PSA parancs nyelv

4.1.1 A parancsok tipusai; a HELP parancs

Mint már emlitettük, a PSL/PSA rendszer felhasználója a rendszert úgy használhatja, hogy operációs rendszer parancsokon keresztül "PSA mód"-ba kerül, s ekkor PSA parancsokkal vezérli a software rendszer tevékenységét. A PSA parancsok három nagy csoportba tartoznak. Ezek: a vezérlő parancsok, módositó parancsok, riport parancsok. A vezérlő parancsok bizonyos vezérlő információt visznek át a PSA-ba. Ezek a parancsok függnek az operációs rendszertől. A PSA adatbázis tartalmát nem változtatják. A módositó parancsok a PSA adatbázis tartalmát módositják, a probléma kezelője által meghatározott módon. Inputjuk PSL utasitás vagy PSL név. Generálnak hibadiagnózist és riportokat a változás kimeneteléről. A riport parancsok az információt keresik vissza a PSA adatbázisból, és standard riportok formájában megjelenitik azt. Az adatbázis tartalmát nem változtatják. Van egy parancs a HELP parancs, amely az előző csoportok egyikébe sem sorolható, mivel ez sem a

PSA müködési módját nem befolyásolja, sem a PSA adatbázisban levő információhoz nem fér hozzá. Ez a parancs a PSA parancsok szintaxisáról és paramétereiről ad információt a usernek. A parancs hatására a PSA listát ad a rendelkezésre álló parancsokról és röviditésükről. Ha a HELP parancshoz paraméterként egy PSA parancs nevet csatolunk, akkor a jelzett parancs rendelkezésére álló összes paraméter listája jelenik meg. /pl. HELP FPS/ Ha pedig egy adott PSA paranccsal együtt még a LONG paramétert is használjuk, akkor a jelzett parancs összes paramétere megjelenik, az egyes paramétereknél a paraméter parancsra gyakorolt hatásának leirásával együtt. /pl. HELP FPS LONG/

4.1.2 A parancsok paraméterei

Mint az emlitettekből kitünik, a parancsok müködését paraméterekkel lehet befolyásolni. A paramétereknek öt alapvető tipusa van.

1. Input adat paraméterek
   Ezek a paraméterek specifikálják a parancshoz inputként használandó adatokat. Input adat paraméterekre példa az INPUT, a FILE és a NAME paraméter.

2. Input kontrol paraméterek
   Ezek a paraméterek specifikálják, hogy a parancsnak hogyan kell használnia, változtatni stb. az input adatokat. Erre a tipusra példa a TYPE paraméter a CHANGE-TYPE parancsnál.

3. Output adat paraméterek
   Ezek a paraméterek szabják meg, hogy a parancsokból generáltassék-e output is, ha igen milyen formában. Erre a tipusra példa a PUNCH és PRINT paraméter.

4. Output opció paraméterek
   Ezek a paraméterek az outputba befoglalható vagy elhagyható opciókat specifikálják. Példa ilyenre, a LEVELS paraméter a CONTENTS parancsnál.

5. Output formátum paraméterek
   Ezek a paraméterek a parancsból származó output információ megjelenitési formátumát határozzák meg. Példa ilyen paraméterre a NEW-PAGE és a HMARG paraméter az FPS parancsnál.

Két fontos szempont a parancsok használatával kapcsolatban, hogy hogyan szolgáltatunk inputot a parancsokhoz, illetve hogyan kapunk outputokat belőlük. Mint az imént láttuk ezek megoldása paraméterek segitségével történik. Az input szolgáltatás esetében az input adat paraméterek nevezetesen a NAME, az INPUT és a FILE paraméter használható. Ha csupán egy input adatot /PSL nevet/ akarunk szolgáltatni a parancshoz, akkor ezt a NAME paraméter útján /NAME=név/ tehetjük meg. Ekkor a parancs által megszabott tevékenység csupán erre az egy input adatra fog vonatkozni /pl. DELETE NAME=vmi/.

Ha több input adatunk van, melyek mindegyikével ugyanazt a tevékenységet kivánjuk elvégezni, akkor elkerülhetjük azt a kényelmetlenséget, hogy mindegyikre egyesével kiadjuk ugyanazt a parancsot a NAME paramétert használva. Az elkerülés módja, hogy input adatainkat meghatározott formátumban tároljuk egy file-ban és a parancs számára jelezzük, hogy e file tartalmát kell inputként használnia. Ez a jelzés a FILE ill. az INPUT paraméter útján történik. A FILE és az INPUT paraméterek abban különböznek egymástól, hogy az általuk jelzett file-ban az input adatok milyen formátumban vannak megadva. A FILE paraméterhez tartozó file általában rekordonként /80 karakteres rekord/ egy nevet tartal-

maz. Az INPUT paraméterhez tartozó file rekordjai általában PSL utasitásokat tartalmaznak kártyaképszerüen.

Egy kényelmes mód arra, hogy inputot specifikáljuk parancsokhoz, hogy a NAME-GEN parancsra bizzuk az input file elkészitését. A NAME-GEN bizonyos neveket szelektál az adatbázisból /ezt paraméter szabályozza/ és ezeket egy file-ba irja a FILE paraméternek megfelelő formátumban. Ezt a file-t azután más parancs inputként használhatja /pl. NAME-GEN GROUP, CONTENTS F/. E példa teljes magyarázatához a paraméterek rendszeréről el kell mondanunk a következőket. Sok paraméternek van igen-nem változata /pl. PRINT-NOPRINT/ és az igen változatnak sok esetben értékadási lehetőségei. Meg kell azonban itt jegyezni, hogy néhány parancs nem engedi meg az alternativát. Ha azonban egy parancsnál az igen-nem változat bármelyike használható, akkor az egyik rendszerint "default" érték, azaz ha egyiket sem adjuk meg, akkor a rendszer valamelyiket feltételezi. Továbbá sok esetben van "default"-ja az igen változat értékének is. Igy tehát az előző példa magyarázata a következő. Miután a NG parancs rendelkezésére álló PUNCH és NOPUNCH paraméterek egyikét sem adtuk meg a rendszer a default értéket veszi, ami PUNCH. Nem adtuk meg továbbá PUNCH=dsi formában azt sem, hogy melyik legyen a punch-file, igy a rendszer a NG parancs számára kijelölt default punch-file-t használta. A CONTENTS parancsnál használtuk a FILE paramétert, de nem jeleztük FILE=dsi formában, hogy melyik legyen az input file. A rendszer tehát a CONTENTS parancs default input file-ját használja, amely jelen esetben - s egyébként a legtöbb parancsnál, mely a FILE paramétert megengedi - megegyezik a NG parancs default punch file-jával. A példa rávilágit egy másik dologra is. Igaz ugyan, hogy a megfelelő időben /vagyis

PSA módban/ a felhasználó bármilyen sorrendben, egymástól függetlenül kiadhatja parancsait, azonban - mint az előző példa rámutatott - sok esetben célszerü sorrendet alkalmazni.

Az emlitett másik szemponttal, vagyis a parancsokból származó outputok nyerésével kapcsolatosan a következőket érdemes megjegyezni. A PSA parancsokból származó output alapvetően kétféle lehet. Az egyik a PRINT paraméter hatására adódó, riport formájú, azaz fejléccel, hibaüzenetekkel, összegzésekkel stb. ellátott output. A másikat csak néhány parancs engedi meg, és ez a PUNCH paraméter hatására jön létre. Ez az output lényegileg ugyanazt az információt tartalmazza mint az előbbi, csak más formátumban. A formátum a fejlécek, hibaüzenetek, összegzések, stb. elhagyásával olyan, hogy az elfogadható más parancsok FILE ill. INPUT paramétere által. A PUNCH paraméternek érték adható, vagyis meghatározhatjuk, hogy az output milyen file-ra kerüljön. Ha nem adunk értéket akkor a default file a használt parancshoz tartozó default punch file.

Bizonyos parancsok esetében lehetőség van a PRINT paraméter párjának vagyis a NOPRINT-nek a használatára. Ez a paraméter a parancsból származó print formájú output elnyomását eredményezi. Sok minden indokolhatja e paraméter használatát, példaként álljon itt a már emlitett eset, /NAME-GEN GROUP, CONTENTS F/, ahol az NG-t csupán a CONTENTS parancs input file-jának elkészitésére használjuk, és nem vagyunk kiváncsiak a NG printjére.

4.1.3 A módositó parancsok

A PSA parancsait három csoportba osztottuk, a vezérlő,

a riport és a módositó parancsok csoportjába. A vezérlő parancsokról - operációs rendszertől való függésük miatt - itt nem lesz bővebben szó, csupán rendeltetésükkel együtt felsoroljuk őket. A négy vezérlő parancs: STOP, SET, SAVE, MTS. A STOP a PSA mód befejezését jelző parancs. A SET parancs bizonyos globális paraméterek beállitására szolgál. A SAVE az adatbázis tartalmának kimentését végzi. És végül az MTS parancs célja, hogy segitségével a PSA módban lévő felhasználó időlegesen visszaadhassa a vezérlést az operációs rendszernek.

A másik csoportról, a riport parancsokról a PSA outputjaival kapcsolatban lesz szó, mivel ezen parancsok célja, éppen az outputok létrehozása. A harmadik csoport a PSA módositó parancsai. Mint már tudjuk ezen parancsok tevékenysége a PSA adatbázis változtatása, továbbá output generálása a változtatás kimenetéről. Az adatbázis vázlatos strukturájának ismeretében nézzük meg, milyen feladatokat kell a módositó parancsoknak megoldani.

1. Uj információ felvitele az adatbázisba

2. Egy név rekord megváltoztatása
   i/ az objektum nevének megváltoztatása
   ii/ az objektum tipusának megváltoztatása

3. Objektumok közötti kapcsolat törlése

4. Egy név rekordnak és minden más objektumokkal való kapcsolatának törlése

5. Egy comment bejegyzés megváltoztatása

6. Egy comment bejegyzés törlése

A PSA-ban az összes emlitett tevékenység végrehajtására lehetőség van. A megfelelő parancsok a következők:

1. INPUT-PSL - a parancs a PSL nyelvü leirás információit tárolja az adatbázisban

2. i/ RENAME - a parancs csupán a név rekordon belüli információt változtatja
   ii/ CHANGE-TYPE - a parancs csupán a név rekordon belüli információt változtatja

3. DELETE-PSL - a parancs név rekordok közötti kapcsolatokat töröl /vagyis a megfelelő NUB rekordokat/; nem töröl név rekordokat, sem comment bejegyzést

4. DELETE - a parancs név rekordot töröl; ha a törölt név rekordnak kapcsolata van más név rekorddal vagy comment bejegyzéssel, akkor a megfelelő NUB rekordokat is törli

5. REPLACE-COMMENT-ENTRY - a parancs csupán egy comment bejegyzés rekordon belüli információt változtatja

6. DELETE-COMMENT-ENTRY - a parancs comment bejegyzés rekordokat és a megfelelő NUB rekordokat törli

A módositó parancsokat, paramétereiket és ezek leirását az 1. táblázat tartalmazza a teljességre való törekvés nélkül.

1. táblázat
Módositó parancsok

Parancs /zárójelben a röv./	Paraméter /zárójelben a röv./	"Default" érték	A paraméter funkciója
INPUT-PSL /IP/	INPUT /I/=dsi	INPUT=	A parancs input file-ját, azaz a PSL nyelvü leirást tartalmazó file-t specifikálja
	DBREF /D/ NODBREF /ND/	DBREF	A parancs végrehajtása során a PSA használja-e vagy sem a már adatbázisban levő információt
	UPDATE /U/ NOUPDATE /NU/	NOUPDATE	Az új információ bekerüljön-e az adatbázisba vagy sem
	SOURCE /S/ NOSOURCE /NS/	SOURCE	Generáltassék-e az AS-IS SOURCE LISTING riport vagy sem
	XREF /X/ NOXREF /NX/	NOXREF	Generáltassék-e a fenti riportra vonatkozóan a PSA CROSS REFERENCE vagy sem

1. táblázat
folytatása

RENAME /REN/	INPUT /I/=dsi OLD /O/ = objektum NEW /N/ = név	nincs nincs nincs	Az OLD paraméterrel jelzett objektum neve a NEW paraméterrel jelzett névre változik. Egyszerre végzendő több változtatás esetén az INPUT paraméterrel egy file-t adunk meg amely az OLD-NEW párokat tartalmazza egyes soraiban.
CHANGE-TYPE /CT/	FILE /F/[=dsi]	nincs; ha azonban a FILE-t használjuk és dsi-t nem adunk, akkor a default a NG parancs default PUNCH file-ja	A NAME paraméterrel jelzett objektum tipusa a TYPE paraméterrel adott tipusra változik. Több változtatás esetén a FILE paraméterrel adjuk meg azt a file-t, melynek egyes sorai az objektum nevet és ha TYPE paramétert nem használunk, akkor az új tipust is tartalmazzák

1. táblázat
folytatása

	TYPE /T/=tipus	nincs	Az új tipus
DELETE-PSL /DPSL/	INPUT /I/=dsi	INPUT=10	Az egyes paraméterek hatása megegyezik az IP parancs megfelelő paramétereinek hatásával
	SOURCE /S/ NOSOURCE /NS/	SOURCE	
	XREF /X/ NOXREF /NX/	NOXREF	
DELETE /DEL/	FILE [=dsi] NAME=objektum név	mint a CT parancs esetén	A NAME paraméterrel megadott objektum törlődik az adatbázisból. Több név törlése esetén a FILE paraméterrel adjuk meg azt a file-t, melynek egyes sorai a törlendő neveket tartalmazzák.

1. táblázat
folytatása

REPLACE- COMMENT- ENTRY /RCOM/	INPUT=dsi	INPUT=a PCOM parancs default PUNCH file-ja	Az INPUT file formátuma: név comment-entry tipus; ⋮ a comment szöv. ⋮ stb. ismétlődve Az adott "név" adott tipusú "comment-entry"-je a megadott szövegre változik
	PRINT NOPRINT /NP/	PRINT	Generáltassék-e a REPLACED COMMENT ENTRIES riport vagy sem.
DELETE- COMMENT- ENTRY /DCOM/	FILE [=dsi] NAME=objek-tum név	mint a CT parancs esetén	A megadott név /NAME/ vagy nevek /FILE/ jelzett tipusú "comment-entry"-jei törlődnek.

1. táblázat
folytatása

	DERIVATION /DER/ DESCRIPTION /DESC/ FALSE-WHILE /FW/ PROCEDURE /PRCD/ TRUE-WHILE /TW/ VOLATILITY /VOL/ ⋮	nincs	a törlendő tipus
	PRINT NOPRINT	PRINT	generáltassék-e a DELETED COMMENT ENTRIES riport vagy sem

4.2 PSA outputok

A PSA outputjai azok, melyek a rendszer elemzőjét közvetlenül segitik, melyek a logikai rendszertervezésben érdekelteket tájékoztatják. Az outputokban kell hogy megjelenjen a PSA minden hatékonysága, segitő képessége. A végcél is output, vagyis a PSL/PSA rendszer esetében a logikai rendszertervezés fázisában létrejött cél-rendszer teljes, konzisztens, kétértelmüségektől mentes leirása.

4.2.1 Az outputok, a parancsok tipusa szerint

A PSA outputjait osztályozhatjuk aszerint, hogy milyen tipusú parancs hatására jönnek létre. Mint tudjuk a parancsok három tipusa: a riport, a módositó és a vezérlő parancsok. A vezérlő parancsoknak - eltekintve az esetleges hibaüzenetektől - nincs outputjuk. A riport parancsok outputjai a parancsból származó visszakeresés eredményét jelenitik meg. Az adatbázis tartalmára vonatkozó kérdések a PSA-ban riportként vannak implementálva. Egy kérdés a PSA-ban, egyszerüen egy riport egyetlen névre. Ez a képesség lecsökkenti a különféle visszakeresési módokra vonatkozó különféle tipusú parancsok megtanulásának nehézségét. A visszakeresés négy féle kritérium szerint illetve ezek bármilyen kombinációja szerint történhet.
Ezek a következők:

- név tipus szerint /pl. PROCESS/
- PROBLEM-DEFINER szerint /a visszakeresés az összes olyan objektumra vonatkozik, melyet a jelzett probléma kezelő definiált/
- KEYWORD szerint /a visszakeresés az összes olyan objektumra vonatkozik, melyhez a jelzett kulcsszót hozzárendeltük/

- egy bizonyos objektumra vonatkozó SUBPART-ok szerint /a visszakeresés az összes olyan objektumra vonatkozik, melyet a jelzett objektum részeiként definiáltunk/

A visszakeresési kritériumok lehetővé teszik, hogy az elemző tökéletesen szelektálhassa az output riportok tartalmát, azaz a felhasználó sem többet, sem kevesebbet nem kap a kivánt információnál.

A módositó parancsok outputjai, amellett, hogy a PSA automatikus konzisztencia, egyértelmüségi és teljességi elemzéseket végez, bizonyos speciális elemzések lehetőségét kinálják a végső dokumentáció javitása érdekében.

4.2.2 Az outputok, rendeltetésük szerint

Az outputok osztályozásának egy másik szempontja, hogy hogyan használják fel őket illetve kinek vannak szánva. Eszerint az outputoknak a következő célokat kell szolgálniuk.

1. Segiteniük kell a probléma kezelője és a felhasználók kapcsolatát. A felhasználók itt azokat jelentik, akik a célrendszert annak elkészültekor használni fogják. Mint emlitve volt, a célul kitüzött rendszer leirása éppen ezen felhasználók által gyüjtött adatokon alapszik, igy tehát ők azok, akik a célrendszer alkalmasságát eldönthetik. Ezért az outputoknak olvashatóknak kell lenniük a felhasználó számára, tartalmazniok kell szótárakat, szómagyarázatokat, ezzel segitve a terminológia összhangját.

2. Segiteniük kell a probléma kezelő munkáját. A szempontok:
   - Az inkonzisztenciák, kétértelmüségek stb. felfedezése és javitása a PSA adatbázisban már tárolt információban.
   - Annak meghatározása, hogy milyen további adatok szükségesek a probléma leirás teljessé tételéhez.
   - Gyors, könnyü és egyszerü módszerek az újonnan csatolandó és a korrigált adatok inputra előkészitéséhez.
   - Olvasható beszámolók az elvégzett munkáról.

3. Segiteniük kell a project koordinálását. Az idevágó riportok több probléma kezelő munkájának koordinálását segitik pl. globális szótárak és vezetők által. A munkák átfedésének elkerülését az a lehetőség segiti, hogy a project állását ellenőrzendő a meglévő információ bármikor visszakereshető az adatbázisból.

4. A végső dokumentáció kialakitása. Az outputoknak szolgáltatniuk kell a teljes probléma dokumentációját, miután a probléma kezelője a logikai tervezést befejezte és az összes információ az adatbázisban van. Végső dokumentációkat a következő célokra készitenek:
   - folyamatos beszámolás
   - végső értékelés a felhasználói igények kielégitéséről
   - leirás a fizikai rendszertervezéshez
   - bárkinek, aki a rendszer megértésében érdekelt.

   Következésképpen a dokumentációnak
   - tartalmaznia kell az ilyen dokumentációra vonatkozó szervezeti szabványok által követelt információkat

- könnyen olvashatónak kell lennie, hogy bárki akinek a rendszerhez köze van, a neki szükséges mélységig megérthesse
- tartalmaznia kell összegzéseket és keresztmetszet referenciákat, hogy az objektumokat különböző fokú részletességgel le lehessen irni és az egy objektumra vonatkozó összes információt könnyen meg lehessen találni.

5. Segiteniük kell a project vezetését. Szükség van olyan riportokra, amelyek segitik a project vezetését abban, hogy a PSL/PSA-t használó tervezési folyamat állását áttekinthessék és haladását értékelhessék.

Hogy az emlitett célokat a riportok hatékonyan szolgálhassák, a riportok formátumának olyannak kell lennie, hogy a riportok önmagukban érthetőek, kifejezőek legyenek és megfeleljenek - akár még lapméretben is - a szervezet dokumentációs szabványainak. A riportok, a megjelenési formátum szerint három fő részből állnak: a cimrész, a részletezési rész, összegzési rész /ha van értelme/.

A cimrész tartalmazza a riport nevét, a dátumot, a probléma nevét /ha a user megadta/, és a riportot generáló parancs paraméter értékeit.

A részletezési rész a riport fő része. Ez rendszerint az adatbázis tartalmának egy kiválasztott részhalmazát jeleniti meg. A riport parancs feldolgozása során bekövetkezett hibaüzenetek legtöbbje is itt jelenik meg.

Az összegzési rész riport statisztikákat és értesitő üzeneteket tartalmaz. Miután ez nem mindig értelmes, ez a rész csak azokban a riportokban jelenik meg, amelyekben hasznos az ilyen információ pl. DATA PROCESS

riport.

4.2.3 Az outputok tartalmuk szerint

Egy újabb és végül a leglényegesebb osztályozása a PSA outputoknak az, amelyik a tartalmuk illetve az általuk elemzett kapcsolatok szerint történik. Tiz csoport lesz, itt már megemlitjük az egyes riportokat is a teljes részletességre való törekvés nélkül. Az egyes riportoknál jelezzük, hogy mely parancs hatására jönnek létre. A riport parancsokat a 2. táblázat foglalja össze. Ha egy riportnál nincs jelezve, hogy milyen parancs hozza létre, akkor ez azt jelenti, hogy a riport még nincs beépitve a rendszerbe, egyelőre nem hozzáférhető.

1. Input adat és módositás outputjai

A rendszerépités logikai rendszertervezési fázisában a célul kitüzött rendszer leirása állandóan változás alatt van. Uj információkat csatolunk a leiráshoz régieket változtatunk. A leirás összes változtatásait, módositásait dokumentálni kell. A PSA a különféle módositás-tipusok mindegyikéről külön riportot generál, melyek a következők:

PSA AS-IS SOURCE LISTING		INPUT-PSL	
PSA CROSS REFERENCE	1		
CHANGE-TYPE REPORT		CT	parancs
DELETION REPORT		DELETE	hatására
DELETED COMMENT ENTRIES		DCOM	
RENAME REPORT		RENAME	
REPLACED COMMENT ENTRIES		RCOM	

2. A teljes probléma leirásról való beszámolók

A rendszerleirás folyamán gyakran szükséges, hogy egy bizonyos objektumra vonatkozó összes információt felleljünk. A megfelelő outputoknak, hogy a rendszerleirás e szempontját szolgálják, meg kell jeleniteniük az összes információt, amely egy megadott objektummal kapcsolatban rendelkezésre áll. A PSA-ban egyetlen riport, a FORMATTED PROBLEM STATMENT oldja meg ezt a feladatot. A riport az ugyanolyan nevü parancs /FPS/ hatására jön létre.

3. Vezetők, szótárak, szómagyarázatok, KWIC index

A vezetők, szótárak, szómagyarázatok információira szükség van egyrészt a különböző emberek munkájának koordinálásához, másrészt összegző információk megjelenitéséhez, amikor is nincs szükség teljes probléma leirásra. Ezek az outputok megfelelően rostált információt adnak az objektumokról. A PSA-ban a rendszerleirás e szempontjai név listák, név indexek és adat definiáló szótárak formájában jelennek meg. A NAME GEN és a PSA NAME LIST outputok /NG ill. NAME-LIST parancs hatására/ szelektált név listákat adnak. A PSA KWIC INDEX riport az adatbázisban levő nevekről, a nevekben előforduló kulcsszavak alapján készit indexet. /KWIC parancs hatására/ A DICTONARY REPORT és a PUNCHED COMMENT ENTRY riportok /DICT ill. PCOM parancs hatására/ összegző információkat adnak a probléma leirásáról, szótárként, szómagyarázatként szolgálnak.

4. Struktura outputok

A rendszer leirás egy másik megjelenitendő szempontja a különböző objektumok strukturája a rendszeren belül. Pl. segitségként a fizikai rendszertervezőnek ismertetni kell, hogy a rekordok miként kapcsolódnak a file-okhoz és milyen adatok tárolandók a rekordokban. Ez a speci-

fikus információ szükséges a célrendszer adatstrukturájának megkonstruálásához. A probléma kezelője által definiált logikai és fizikai strukturák megjelenitésére a PSA-ban a következő riportok állnak rendelkezésre:

CONTENTS REPORT	CONT	
STRUCTURE	STR	parancs
INTERVAL STRUCTURE	-	hatására
PICTURE	PIC	

A PICTURE riport a CONTENTS és a STRUCTURE riport információit grafikus formában jeleniti meg. Azok az outputok, amelyek a file és adat kapcsolatokat jelenitik meg és segitenek strukturáik optimalizálásában a következők:

CONSISTS MATRIX	CM	
IDENTIFIER INFORMATION REPORT	EI	parancs
RELATION INFORMATION REPORT	--	hatására

5. "Áramlási" outputok

Egy újabb követelmény, melyre szükség van a rendszer teljes leirásához az, hogy az információk milyen dokumentumokhoz tartoznak és az adatok hogyan kerülnek felhasználásra a rendszerben. Mit kell inputként használni a müködés bármely adott fázisában és mik az outputok; két megválaszolandó kérdés. Az idevágó outputok tartalmának változniuk kell aszerint, hogy a kapcsolatokat az objektumok milyen szintjén jelenitik meg. Az egyik adat elemek szintjén jeleniti meg az információ áramlást, a másik file szinten. A PSA idetartozó outputjai:

DATA PROCESS REPORT	DP	
PROCESS INPUT/OUTPUT	PRIO	parancs
PICTURE	PIC	hatására

6. Dinamika elemzés outputjai

A rendszerleirás dinamika-analizis szempontja, a rendszer időbeni változásának leirásához szükséges. A megválaszolandó kérdések itt: milyen gyakran kell a rendszernek egy bizonyos tevékenységet végrehajtania és mikor kell azt végrehajtania. A rendszert nyilván egészen másként fogják megtervezni, ha a célja évi egy, vagy ha napi hat output létrehozása. A dinamika-elemzés két alapvető szempontja tehát, hogy /1/ mikor történik valami és /2/ hogy milyen gyakran történik az a valami. A PSA-ban a mikor kérdéshez tartozó információt megjelenitő outputok,

EVENT/CONDITION REPORT	--
DYNAMIC ANALYSIS REPORT	--

A milyen gyakran kérdéshez pedig a

FREQVENCY	FREQ	parancs
DYNAMIC	--	hatására

riportok tartoznak.

7. Méret és volumen outputok

A rendszer leirásának tartalmaznia kell a rendszer méretére és a végzendő feldolgozás volumenére vonatkozó információkat is. Ezek ugyanis igen nagy befolyással vannak arra, hogy a rendszert hogyan kell felépiteni. A volumen és a méret pl. kikötéseket tehet a CPU-ra és a háttérmemória méretre. A három nagyobb terület, amit a méretre vonatkozóan le kell irni, a rendszer paraméterei a file-ok és a feldolgozás. A rendszer paramétereket kezelő outputok:

SYSTEM PARAMETER ANALYSIS	--
SYSTEM PARAMETER SIZE	--

A SET SIZE REPORT a SET-ek /vagyis a file-ok/ méretére vonatkozó információkat jeleniti meg.

A PROCESSING VOLUME REQUIREMENTS riport arról ad információt, hogy egy folyamat milyen gyakran kerül elindításra és hogy e folyamat mennyi információt kezel.

8. Rendszer-tulajdonságok outputjai

Az egyes objektumok leirása mellett szükség van olyan outputokra is, melyek megjelenitik a rendszer különféle tulajdonságait, úgy mint költség, implementálás, és bármi ami a rendszerépitités későbbi fázisaira vonatkozik. Ezek a speciális elemzések eredményeként adódó outputok munkamennyiség becsléssel, haszon/költség elemzésekkel stb. foglalkoznak. A PSA-ban jelenleg csupán egyetlen idetartozó output az ATTRIBUTE REPORT /a PAV parancs hatására/ áll rendelkezésre. Ez a riport tulajdonképpen csupán az emlitett speciális elemzések alapja. A további, a valóban érdekes riport-képességeknek a PSL/PSA rendszerhez való csatolása fejlesztés alatt áll.

9. Probléma elemző outputok

A probléma leirás bármely stádiumában szükség van arra, hogy ellenőrizzük, vajon a leirás korrekt és egyértelmü-e és hogy milyen információk szükségesek a leirás teljessé tételéhez. Talán ennek lebonyolitása az egyik legfontosabb és legnehezebb szempontja a logikai rendszertervezésnek. A rendszerépitités későbbi fázisainak eredménye csak annyira lehet jó, amennyire a logikai tervezés fázisából adódó végső leirás az. A már előzően tárgyalt riportok közül sok foglalkozik ezzel a kérdéssel. Az AS-IS SOURCE LISTING riport például jelzi az új input adatok inkonzisztenciáját, mielőtt azok az adatbázisba kerülnének. A DATA PSOCESS és a CONSISTS MATRIX

riport a leirásban lévő hiányosságokra figyelmeztet.

Van azonban egy riport, melyet speciálisan az ilyen információk megjelenitésére terveznek, és ez a

COMPLETNESS/CONSISTENCY REPORT

10. Outputok a project vezetésének

A logikai rendszertervezés folyamán bizonyos időpontokban a probléma kezelőinek jelentéseket kell készitenüük a munka állásáról a project vezetés számára. E riportok mutatják, hogy a munka - remélhetően - halad. A PSA által kifejezetten a project vezetés számára szolgáltatott outputok mennyiségi információkkal igyekeznek tájékoztatni a helyzetről. A probléma leirás minősége szintén kiolvasható ezen outputok információjából. A jelenleg rendelkezésre álló két output:

DATA BASE STATISTICS	DBS	parancs hatására
DATA BASE SUMMARY	SUM	

2. táblázat
Riport parancsok

Parancs /Zárójelben a röv./	Paraméter /Zárójelben a röv./	"Default" érték	A paraméter funkciója
FORMATTED - PROBLEM - STATMENT /FPS/	FILE/FI =dsi NAME /N/= objektum név	FILE /az NG parancs default punch file-ja/	A megadott névre /NAME/ vagy nevekre /FILE/ a FORMATTED PROBLEM STATMENT riport adódik.
	INDEX NOINDEX	NOINDEX	legyen-e a riportban szereplő nevekről index vagy sem
	PRINT NOPRINT /NP/	PRINT	megjelenjen-e a riport a sornyomtatón vagy sem
	PUNCH /P/ =dsi NOPUNCH	NOPUNCH	legyen-e punch-output vagy sem; ha PUNCH-t használunk és dsi-t nem adunk meg, akkor az output a parancs számára kijelölt default punch-filera kerül
	COMMENT /COM/ NOCOMMENT /NCOM/	COMMENT	tartalmazza-e az output a nem definiált nevekre von. megjegyz. vagy sem

2. táblázat
folytatása

	DEFINE /DEF/ NODEFINE /NDEF/	DEFINE	tartalmazza-e az output a DEFINE fejezeteket v. sem
	DESG /DG/ NODESG /NDG/	DESG	tartalmazza-e az output a DESIGNATE fejezeteket v. sem
	EMPTY NOEMPTY	lásd magyarázat	EMPTY esetén /ez a default ha PUNCH-ot használunk/ a punch-file kiüritődik mielőtt irás történik rá. NOEMPTY-nél forditva értelemszerüen.
	Van még néhány az output formátumát szabályozó paraméter, amelyet nem részletezünk.		
NAME-GEN /NG/	PRINT NOPRINT	PRINT	a parancs az objektum nevek paraméterekkel leirt részhalmazát választja ki; legyen-e printelt output vagy sem
	PUNCH =dsi NOPUNCH	PUNCH	mint az FPS parancsnál

2. táblázat
folytatása

	EMPTY NOEMPTY	mint	az FPS parancsnál
	ORDER = = { ALPHA / BYTYPE }	ORDER= =ALPHA	ALPHA esetén a kiválasztott nevek alfabetikus sorrendben adódnak; BYTYPE-nél tipus szerint és azon belül alfabetikusan
	a többi paramétert, melyek mindegyike a kiválasztandó részhalmaz körülirását szolgálja, sokaságuk miatt nem részletezzük		
NAME-LIST /NL/	ORDER= = { ALPHA / BYTYPE }	ORDER= =ALPHA	a prancs az összes nevek listáját adja a megadott rendben /mint NG parancsnál/
KWIC	FILE [=dsi]	ha dsi-t nem adunk, akkor a NG parancs default punch file-ja az input file	A FILE-ban adott nevekre a KWIC riport adódik.
	DIF=egész szám	DIF=20	output formátum

2. táblázat
folytatása

DICTIONARY /DICT/	FILE[=dsi] NAME=objektum név	mint az FPS parancsnál	A megadott névre /NAME/ vagy nevekre /FILE/ a DICTIONARY REPORT adódik.
	INDEX NOINDEX	NOINDEX	tartalmazzon-e az output a riportban szereplő nevekről indexet vagy sem
	Van még néhány output opció és output formátum paraméter, melyeket nem részletezünk.		
CONTENTS /CONT/	FILE[=dsi] NAME=objektum név	mint az FPS parancsnál	A megadott névre vagy nevekre /FILE/ a CONTENTS REPORT generálódik
	INDEX NOINDEX	NOINDEX	tartalmazzon-e a riport a benne szereplő nevekre vonatkozóan indexet vagy sem
	LEVELS= {egész szám ALL}	LEVELS= =ALL	Az "egész szám" a strukturális kapcsolat legalacsonyabb szintü előkeresendő objektumának szintje. ALL esetén minden szint előkeresendő.

2. táblázat
folytatása

STRUCTURE /STR/	{ INPUT /INP/ OUTPUT /OUT/ PROCESS /PROC/ INTERFACE /INTF/ }	PROCESS	A megadott tipusú objektumokra a STRUCTURE REPORT generálódik.
	INDEX NOINDEX	NOINDEX	tartalmazzon-e indexet a riport a benne szereplő nevekről vagy sem
	INDENT /IND/= = egész szám	INDENT=3	output formátum
PICTURE /PIC/	FILE [=dsi] NAME=objektum név	mint az FPS parancsnál	Az adott névre /NAME/ vagy nevekre /FILE/ a PICTURE riport generálódik
	INDEX NOINDEX	NOINDEX	mint eddig
	FLOW NOFLOW	FLOW	tartalmazza-e a riport az áramlási információkat vagy sem
	STRUCTURE /STR/ NOSTRUCTURE /NSTR/	STRUCTURE	tartalmazza-e a riport a struktura információkat vagy sem

2. táblázat
folytatása

	DATA /D/ NODATA /ND/	DATA	tartalmazza-e a riport a struktura és áramlási információkon kivüli egyéb információkat vagy sem
FREQUENCY /FREQ/	nincs		A FREQUENCY REPORT generálódik
CONSISTS- - MATRIX /CM/	FILE =dsi NAME=objektum név	mint az FPS parancsnál	az adott névre /NAME/ vagy nevekre /FILE/ a CONSISTS MATRIX REPORT generálódik
	{ CONTAINED /CNTD/ CONSISTS /CSTS/ }	nincs	a megadott nevekkel a jelzett kapcsolatban álló objektumok lesznek visszakeresve
ENTITY- -IDENTIFIER /EI/	FILE [=dsi] NAME=objektum név	mint az FPS parancsnál	az adott névre /NAME/ vagy nevekre /FILE/ az IDENTIFIER INFORMATION REPORT generálódik
	{ IDENTIFIER /I/ ENTITY /E/ }	nincs	

2. táblázat
folytatása

DATA- -PROCESS /DP/	FILE[=dsi] NAME=objektum név	mint az FPS pa- rancsnál	az adott névre /NAME/ vagy nevek- re /FILE/ a DATA PROCESS REPORT ge- nerálódik
	{DATA /D/ PROCESS /P/}	nincs	
PROCESS- -INPUT- -OUTPUT /PRIO/	FILE[=dsi] NAME=objektum név	mint az FPS pa- rancsnál	az adott PROCESS tipusú névre /NAME/ vagy nevekre /FILE/ a PROCESS INPUT/ OUTPUT riport ge- nerálódik
	INDEX NOINDEX	NOINDEX	
	PRINT NOPRINT	PRINT	mint az eddigiek- ben
	PUNCH[=dsi] NOPUNCH	NOPUNCH	
	EMPTY NOEMPTY		
	a további output opció és output formátum paramétereket nem részletezzük		

2. táblázat
folytatása

PRINT- -ATTRIBUTE- -VALUES /PAV/	FILE[=dsi] NAME=objektum név	mint az FPS pa- rancsnál	az adott ATTRIBUTE tipusú névre /NAME/ vagy nevekre /FILE/ az ATTRIBUTE REPORT generálódik
DATA-BASE- -STATISTICS /DBS/	NAMES NONAMES	NAMES	tartalmazza-e az output az összes adatbázisban lévő névre vonatkozóan a névhez kapcsolt NUB-ok számát
	NAMNUBS NO NAMNUBS	NONAMNUBS	tartalmazza-e az output az összes adatbázisban lévő névre vonatkozóan a névhez kapcsolt NUBA, NUBB, NUBC-k számát külön-külön
	NUBS NONUBS	NUBS	tartalmazza-e az output a NUBA-k, NUBB-k, NUBC-k teljes számát külön-külön
	SYNONYMS /SYN/ NOSYNONYMS /NSYN/	NOSYNONYMS	tartalmazza-e az output a SYNONYM-okat
SUMMARY /SUM/	nincs		A DATA BASE SUMMARY riport generálódik

4.3 Az adatbázis kezelő rendszer

Az adatbázis kezelő rendszert a CODASYL - COBOL Data Base Facility Proposal alapján készitették a michigani egyetemen 1973-ban. A fenti jelentésnek azonban nem tesz teljes mértékben eleget. /Hiányzik pl. a subschema, data-aggregate.../

A leirás öt pontból áll.
Az elsőben az alapfogalmakat tisztázzuk.
A másodikban az adatok strukturájának leirására szolgáló DDL nyelvet ismertetjük.
A harmadik pontban a DDL nyelv segitségével létrehozott adatbázis táblázat file-ról beszélünk.
A negyedikben az adatbázis file-ról lesz szó.
Végül az ötödikben az adatok kezelésére szolgáló DBMS rendszert ismertetjük.

4.3.1 Alapfogalmak

Az alapfogalmak tisztázásánál - a könnyebb egyeztethetőség végett - mindenütt megadjuk a szó angol megfelelőjét. Néhány esetben a pontos magyar szó hiánya miatt esetleg az angolt használjuk.

Tipus /type/ - előfordulás /occurence/
Különbséget teszünk egy objektum tipusa /osztálya/ és az objektum előfordulása /azaz egy konkrét pédánya, vagy aktuális megjelenése/ között. Pl. a "tanuló" lehet egy tipus, "Nagy Béla" a "tanuló" tipus egy előfordulása.

A további három fogalomra az adatbázis strukturájának leirásában lesz szükség.

Elemi adategység /item/ - mint a neve is mutatja, a legkisebb adategység, végsősoron minden szerkezeti egység ebből épül fel. /Gyakran nevezik mezőnek /field/ vagy elemnek /element/./

Rekord /record/ - elemi adategységek megnevezett gyüjteménye. Minden egyes rekordtipusnak tetszőleges számú előfordulása lehet az adatbázisban.

Készlet, halmaz /set/ - a rekordok megnevezett gyüjteménye, amely valamilyen relációt fejez ki. Minden készlettipusnak rendelkeznie kell egy vagy több gazda /owner/ rekord-tipussal és egy vagy több tag /member/ rekord-tipussal. A készlet egy előfordulásában a gazda rekord-tipusnak pontosan egy, a tag rekord-tipusnak tetszőleges számú /akár nulla/ előfordulása szerepel.

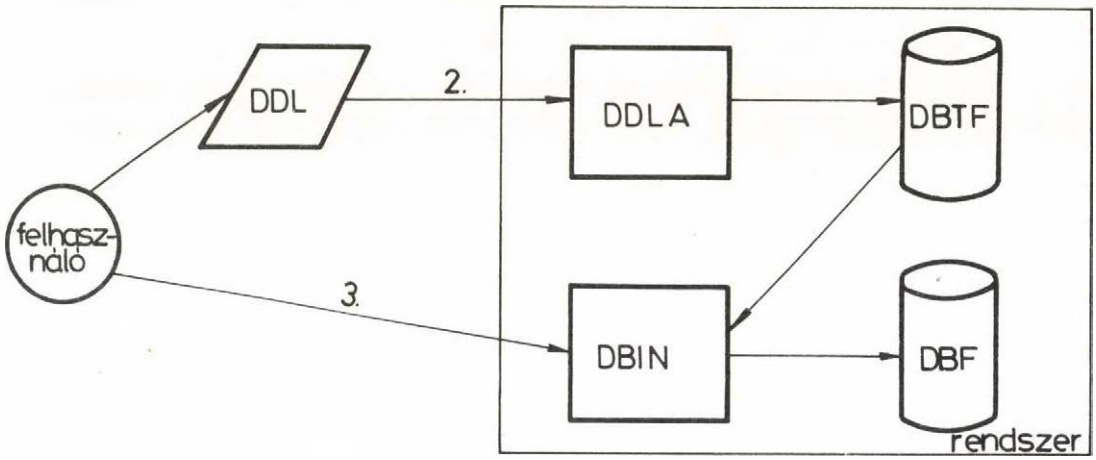
A fentieken kivül szükségünk lesz három mutatóra /currency indicator/.
A kurrens rekord-mutató /current occurence of record type/ minden egyes rekordtipushoz pontosan egy ilyen mutató van. Ez kezdetben nullára van állitva, később a rekordtipus előfordulásának kulcsát tartalmazza.

Egy készlet-tipus kurrens gazdájára mutató jelző /current owner of set type/ - az adott készleten belül egy legális gazda-rekordtipus egy előfordulására vagy nullára mutat.

Egy készlet-tipus kurrens tagja /current member of set type/ - egy készleten belül egy gazdarekord előforduláshoz több tagrekord tartozhat. Ezek közül egyet kitüntetünk, és ez lesz a fenti.

4.3.2 DDL /DATA DEFINITION LANGUAGE/

Mielőtt ismertetnénk a strukturát definiáló nyelvet, nézzük át az adatbázis táblázat és az adatbázis file létrejöttét.



1. A felhasználó DDL nyelven leirja az adatbázis strukturáját.

2. Ezután meghivja a DDLA programot, amely a fenti leirás alapján létrehozza az adatbázis táblázatot /DBTF - Data Base Table File/

3. A DBIN program a táblázat segitségével kialakitja az adatbázis file-t /DBF - Data Base File/, ami ekkor még tulajdonképpen üres.

Az adatbázison történő manipulációt az ötödik részben ismertetjük.

A nyelv leirása

A DDL nyelvben az egyes tipusok /record, set, item/ leirása kártyaképszerüen történik. A DDL nyelvü leirást is általában kártya-file formában szolgáltatjuk inputként a DDLA programnak.

1. RECORD - egy rekordtipus név csak egyszer fordulhat elő
   - a rekordhoz tartozó elemi adategységek kártyáinak közvetlenül a rekord kártyát kell követnie
   - az elemi-adatmezőtipus nevének csak a rekordon belül kell egyértelmünek lennie
   - az utolsó elemi adategység ismétlődő lehet
   - ha az ismétlések számát egy másik elemi adategység tartalmazza, annak az adategységnek is ebben a rekordban kell lennie

Van egy kitüntetett rekord: SYSTEM. Ezt sem létrehozni, sem törölni nem szabad.

A RECORD kártya leirása:

1 - 6 oszlop	RECORD
8 - 13	rekord tipus név

Az item kártya leirása:

1 - 4 oszlop	ITEM
8 - 13	elemi adategység tipusának neve
15 - 20	milyen fajta az elemi adatmező /INTEG, REAL, BINARY, LOGIC, CHAR/
22 - 24	nagyság /INTEG és REAL esetében bitekben, BINARY és CHAR esetében karakterekben, a fennmaradó két

esetben mindig egy szó/
/Megjegyzés: a fizikai helyfoglalás a nagyságtól függetlenül mindig egész szó, vagy annak többszöröse./

26 - 31 melyik elemi adatmezőtől függ az ismétlésszám

33 - 35 maximális ismétlésszám

2. SET - legalább egy gazda és egy tag rekordnak kell lennie
- ha SYSTEM /a kitüntetett rekord/ a gazda, több gazdarekord nem lehet
- SET kártya után az OWNER kártyák, utánuk a MEMBER kártyák jönnek
- az itt szereplő rekordtipusoknak már előzőleg elő kellett fordulni egy RECORD kártyán
- ha rendezési kulcsot /sort key/ definiálunk, akkor annak elemi adategységnek kell lennie, mégpedig CHAR v. INTEG-nek. Ezenkivül minden tagrekordban ugyanazon a helyen és nagyságban kell előfordulnia.

SET kártya leirása

1 - 3 oszlop SET
8 - 13 set tipus név
15 - 20 tagrekordok rendezése /FIFO, LIFO, NEXT, PRIOR, IMMAT, SORTED/
26 - 31 rendezési kulcs /csak akkor, ha a rendezés SORTED/

OWNER kártya:

1 - 5 OWNER
8 - 13 rekord tipus név

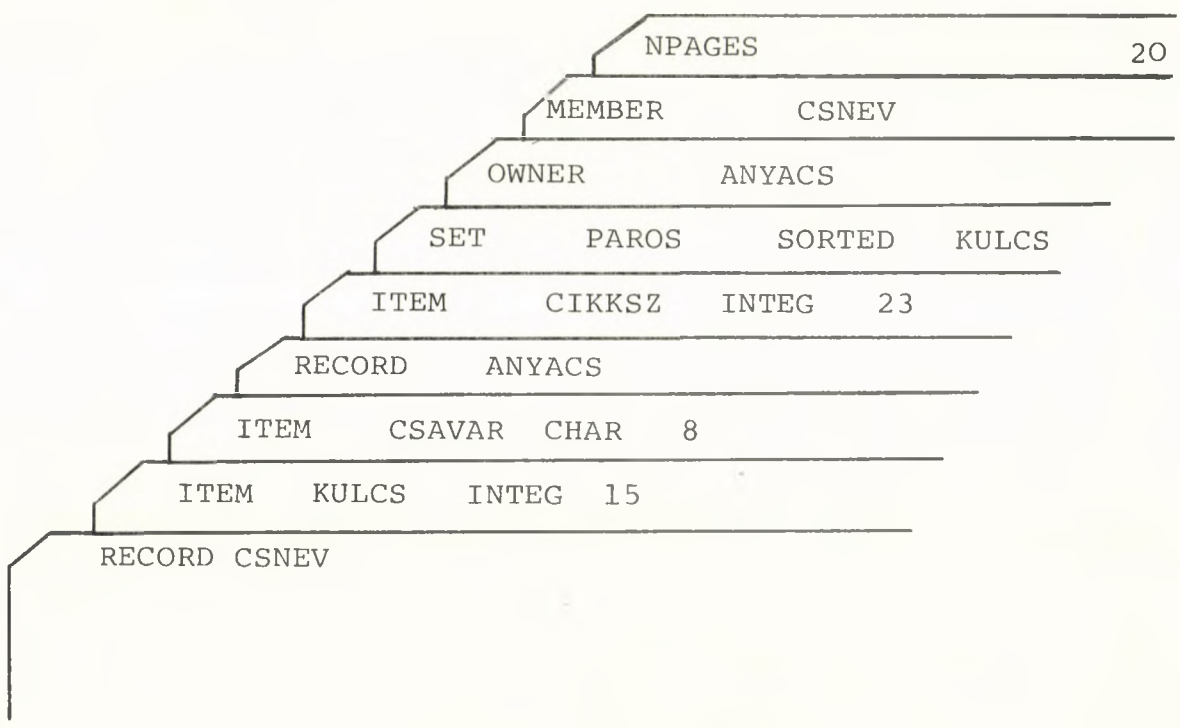
MEMBER kártya:

1 - 5 MEMBER
8 - 13 rekord tipus név

A fentieken kivül még egy kártya létezik.

1 - 6 oszlop NPAGES
22 - 24 az adatbázis nagysága page-kben
/Nálunk egy page 320 szó./

Példa egy DDL nyelvü kártya-file-ra



4.3.3 Az adatbázis táblázat file /DBTF - Data Base Table File/

A DDLA program a DDL nyelven megirt inputból szerkeszti meg a DBTF-t. A táblázat file ismerete nem szükséges a rendszer használatához, ezért nem is irjuk le teljes részletességgel.
A táblázat file négy részből áll
1/ ötszavas paraméter terület
2/ rekord táblázat - RECTAB
3/ készlet táblázat - SETTAB
4/ névtáblázat - NAMES

1/ Az ötszavas paraméter-terület tartalmazza
az adatbázisban lévő page-k számát
a page-k nagyságát
a másik három táblázat hosszát

2/ A RECTAB-ban minden rekord tipusra:

rekord leiró blokk	item leiró blokk	. . .	item leiró blokk

A rekord leiró blokk hat szóból áll. Ebben tárolódik többek közt a kurrens rekord kulcsa, a rekord maximális hossza, item-ek száma ...

3/ A SETTAB-ban minden set tipusra:

set leiró bl.	owner leiró bl.	...	owner leiró bl.	member leiró	...	member leiró

A set leiró blokk hét szóból áll, a többi 3-3-ból. A set leiró blokk tartalmazza a kurrens gazdarekordot, a hozzátartozó kurrens memberrekordot stb.

4/ A NAMES-ben a rekord tipus nevek, az elemi adategységek nevei és a set tipus nevek találhatók. /Az azonos neveket csak egyszer tárolja./

4.3.4 Az adatbázis file inicializálása

A DBIN /Data Base Initializer/ program az adatbázis tábla file alapján létrehozza az adatbázisfile-t. Ez annyit jelent, hogy beszámozza a pege-ket, regisztrálja az üres helyeket, betárolja az első rekordot, a SYSTEM-t.

4.3.5 Az adatábzis vezérlő rendszer /DBCS/

Az adatbázis file feltöltését, az adatok visszanyerését a DBCS /Data Base Control System/-ből vett szubrutinokkal végezhetjük. Ezek a szubrutinok mind FORTRAN-ból, mind COBOL-ból elérhetők.

A page-k kezelése

A memóriában egyszerre maximum 12 page tartózkodhat. Ha új page-re van szükség, akkor a rendszer a legrégebben használt page helyett hozza be az újat.

A rekordok kreálása és törlése

Először a memóriában lévő page-ken történik a helykeresés. A page-ken a "lyukak" növekvő sorrendben láncra vannak füzve. Az első megfelelőt foglalja le a rekord számára /best fit/. Egy rekord törlésénél a kialakuló helyet nullával tölti fel, és - ha lehet - hozzáfüzi az előtte vagy mögötte álló lyukhoz.

A DBCS Szubrutinjai /az aláhuzottak a visszatérő paraméterek/

Vezérlő szubrutinok

OPEN /2,3, nbuf, ierr/

Minden esetben ez az első meghivandó rutin. A 2 és 3 az adatbázis file és az adatbázis táblázat dsi-je. Az "nbuf" a fizikai rekordok I/O-jára szolgáló pufferek száma.

CLOS /switch,array,ierr/

Ez az utolsó meghivandó rutin. A "switch" értékétől függően különböző statisztikákat kaphatunk az "array" tömbbe.

DBST /switch,array,ierr/

Az utolsó megnyitás óta történt adatáramlásokról ad statisztikát.

switch	array	statisztika
1	array/1/	az adatbázisból történt olvasások száma
2	array/2/	az adatbázisból történt irások száma
4	array/3/	a létrehozott rekordok száma
8	array/4/	a törölt rekordok száma
16	array/5/	DBKFND rutin hivásainak száma

Egy rekord előfordulásra négyféleképp hivatkozhatunk:

az adatbázis kulcsa alapján	K
mint egy set kurrens gazdájára	O
mint egy set kurrens tagjára	M
mint egy rekordtipus kurrens rekordjára	R

A rutinok úgy vannak megkonstruálva, hogy nevük utolsó betüje a fenti négy közül valamelyik, aszerint, hogy hogyan azonositjuk a kivánt rekordpéldányt. A rövidség kedvéért a parancsokban e betük helyett x-et használ-

lunk. /Az R-t néha nem kell beirni./

Rekordok létrehozása és törlése

CR /rekord-tipus,db-key,ierr/
CRS /rekord-tipus,data,db-key,ierr/

Mindkettő helyet foglal a rekordnak, a második rögtön fel is tölti.

DRx /rekord-azonositó,ierr/ x = K, O, M, v.bl.
letöröl egy rekordot az adatbázisból.

DELS /set-tipus,ierr/
Az illető set-tipusban a kurrens gazdarekordhoz tartozó összes tagrekordot törli.

A fenti két parancs esetén: ha a törölt rekord egy gazdarekord, akkor a hozzátartozó tagrekordok ugyan bent maradnak az adatbázisban, de már nem fognak az illető set-tipus egyetlen előfordulásához sem tartozni. Ha a rekord kurrens gazdarekord volt, a kurrens gazdarekord és kurrens tagrekord mutatókat nullára állitja a rendszer. Ha kurrens tagrekord volt, akkor a logikailag következő tagrekord lesz a kurrens. /Ha nincs ilyen, nullára állitja./

Rekordok hozzáadása set-hez, rekordok eltávolitása set-ből

AMSx /set-tipus,rekord-azonositó, ierr/
Az adott rekordot hozzáfüzi a kurrens gazdarekordhoz, és ezt teszi kurrens tagrekorddá. /Csak egy példányban fordulhat elő abban a set-példányban./

RM /set-tipus,ierr/
A set-példányból eltávolitja a kurrens tagrekordot, helyette a logikailag következő tagrekord lesz a kurrens.

RS /set-tipus,ierr/
Az illető set-tipus kurrens gazdarekordjához tartozó összes tagrekordot eltávolitja a set-ből.

Rekordból elemi adategységek nyerése, beállitása
SETx /rekordazonositó,adat,ierr/
A rekord összes mezőjét beállitja az "adat" segitségével.

GETx /rekordazonositó,adat,ierr/
A rekord összes mezőjét visszakapjuk az "adat"-ban.

SFx /item-tipus,rekordazonositó,adat,ierr/
Az adott rekordban a megfelelő mezőt beállitja.

GFx /item-tipus,rekord-azonositó,adat,ierr/
Az adott rekordból a megfelelő mezőt megkapjuk az "adat"-ban.

ICFx /item-tipus,rekord-azonositó,adat,ierr/
Az adott rekord megfelelő mezőjét összehasonlitja az "adat"-tal.

Rekordról való érdeklődés
GTx /rekord-azonositó,rekord-tipus,ierr/
Az adott rekord tipusát kapjuk meg. Nyilván x ≠ R.

GKx /rekord-azonositó,adatbázis-kulcs,ierr/
Az adott rekord kulcsát kapjuk meg. x ≠ K.

COx /set-tipus,rekord-azonositó,sw,ierr/
Az adott set-tipusban lehet-e a rekord gazda.

CMx /set-tipus,rekord-azonositó,sw,ierr/
Az "sw"-ben visszatérő érték azt jelzi, hogy az adott rekord tagrekord-e abban a set-ben.

A kurrens gazdarekordról való érdeklődés

COT /set-tipus,rekord-tipus,sw,ierr/
Az "sw"-értéke azt mutatja, hogy a set-ben a kurrens gazdarekord tipusa megegyezik a "rekord-tipussal".

CMT /set-tipus,rekord-tipus,sw,ierr/
A kurrens tagrekord tipusát ellenőrzi.

CARD /set-tipus,adat,ierr/
A kurrens gazdarekordhoz tartozó tagrekordok számát adja meg.

Keresés egy set-ben

FyM /set-tipus,ierr/

y = F első tagrekord
L utolsó tagrekord
N a következő tagrekord
P az előző tagrekord

Az adatt set-ben a kurrens gazdarekordhoz tartozó megfelelő /első,utolsó.../ tagrekordot keresi meg.

FMSK /set-tipus, adat,ierr/

FNSK /set-tipus,adat,ierr/
Csak rendezett /sorted/ set-ben alkalmazható. Megkeresi azt a tagrekordot, amelyben a rendezési kulcs megegyezik az "adat"-tal. Abban különböznek, hogy az első a keresést az első tagrekordtól kezdi, a második a kurrens tagrekordtól.

A gazdarekorddal azonositott, rendezett set-ben való keresés

SxFM /set-tipus,rekord-azonositó,adat,ierr/
A set-tipusban az adott rekordot - ha lehet - kurrens gazdarekorddá teszi, azután a hozzátartozó tagrekordo-

kat nézi végig a rendezési kulcs alapján, s ez lesz az új kurrens tagrekord.

A kurrens rekordokra mutató jelzők állitása

SRx /rekord-tipus,rekord-azonositó,ierr/
Az adott rekordot a "rekord-tipus"-on belül kurrenssé teszi.
SOx /set-tipus,rekord-azonositó,ierr/
Az adott set-tipuson belül a fenti rekord lesz a kurrens gazda.

SMx /set-tipus,rekord-azonositó,ierr/
Az adott set-tipusban a fenti rekord lesz a kurrens tag.

Az ierr-ben visszatérő értékek

00	minden rendben
01	az adatbázis nincs megnyitva
02	érvénytelen set-tipus
03	érvénytelen rekord-tipus
04	ebben a rekordtipusban érvénytelen ez a mező
05	ebben a setben érvénytelen gazdarekord-tipus
06	ebben a setben érvénytelen tag-rekord tipus
07	érvénytelen adatbázis kulcs
08	a set tipusban nincs kurrens gazda
09	a set tipusban nincs kurrens tag
10	a rekord tipusban nincs kurrens rekord
11	a rekord már a set-tipus tagja
12	a rekord nem tagja a setnek
13	az ismétlésszám túl nagy vagy negativ
14	az adatbázis már nyitva van
15	DDL inkonzisztens
16	az adatbázis file inkonzisztens

17	nincs több hely az adatbázisban
18	a set nem rendezett
20	a pufferek száma érvénytelen
99	katasztrófa
-1	a set végét elértük

5. AZ ISDOS RENDSZER IMPLEMENTÁLÁSA ÉS HASZNÁLATA A CDC 3300-AS GÉPEN

5.1 Az implementálás

5.1.1 A rendszer igényei

Az ISDOS rendszer 2.1-es verzióját Michigan-ből, az egyetem Industrial and Operations Engineering részlegétől kaptuk mágnesszalagon /69 FORTRAN nyelvü file, 12 teszteléshez szükséges program és egy implementálási füzet N° 111/.

A rendszer kialakitásánál két dolgot tartottak szem előtt: átvihetőség, hatékonyság.
Az elsőnél figyelembe kellett venni az egyes gépek különbségeit /szóhossz, I/O kezelés .../. Emiatt és a nagyobb hatékonyság miatt néhány szubrutint nem irtak meg magasabb szintü nyelven, hanem azok feladatát és IBM kódját adták meg. /Mi természetesen compass nyelven irtuk meg./

Az operációs rendszerrel szemben támasztott igények a következők:

FORTRAN compiler
Library generation /nálunk a GLIB/
Linkage editor /nálunk RLDR/
SORT

5.1.2 A felépités segédeszközei

A file-okat két részre oszthatjuk
a/ a PSL/PSA felépitéséhez szükséges file-ok /az u.n. BETA rendszer/

b/ a tulajdonképpeni PSL/PSA rutinok

Az anyagot nem a compiler számára érthető formában tárolják: a többször előforduló részek külön vannak, helyüket csak egy sor jelzi. Ennek előnye, hogy igy kevesebb helyet foglal el, és ez garantálja a változtathatóságot. Az anyag felépitése a BETA rendszerrel történik, mely a következő programokból áll:

INCA, INCL
MACH
PRMF
UPDT

Az első kettő /INCA, INCL/ ugyanazt a funkciót látja el, azzal a különbséggel, hogy az INCA inputról, az INCL pedig egy adatbázis file-ról dolgozik. Mindkettő a common területek beszúrását végzi.
Például *CALL,LIO sor helyett a LIO nevü commont teszi be.
A MACH a szerkesztés második fázisát látja el: az egy szóban lévő karakterek számától - mely gépfüggő - függően kialakitja az integer tömböket, a DATA utasitást és a FORMAT-ot.

pl. *CHAR,NAME /szám/-ból
INTEGER NAME $\left(\left\lceil \frac{\text{szám+NCW-1}}{\text{NCW}} \right\rceil\right)$ ahol NCW az egy szóban lévő karakterek száma

A PRMF az adatbázisban lévő common - k listázására az UPDT pedig az új common-k felvitelére vagy a régiek kicserélésére szolgál.

5.2 Az adatbázis-kezelő rendszer használata

A rendszer a 854-es tipusú 766-os discen található. Ennek megnyitása

```
$*DEF /O,,dsil, ISDOS,DBLIBRARY/
```

a/ Az első lépés az adatbázis tábla file allokálása és megnyitása outputra 3-as dsi-vel /ennek használata kötelező/. /Blokkméret 512char/

```
$*DEF /A,,acc.szám,filename,...
$*DEF /O,,3,...
```

b/ Ezután a DDLA taskot hivjuk meg. Az inputot az 5-ös dsi-ről várja és end-of-file-ig olvas. Egy "block data" szubrutint lyukaszt, erre a file-on való manipuláláskor lesz szükség.

```
$DDLA,dsil
```

c/ Az adatbázis file-t allokáljuk, és 2-es dsi-vel megnyitjuk outputra. A blokkméret 1280 character.

```
$*DEF /A,,acc.szám,filename,...
$*DEF /O,,2,...
```

d/ A DBIN a táblázatot csak inputra használja, az adatbázis file-t azonban outputra

```
$DBIN,dsil
```

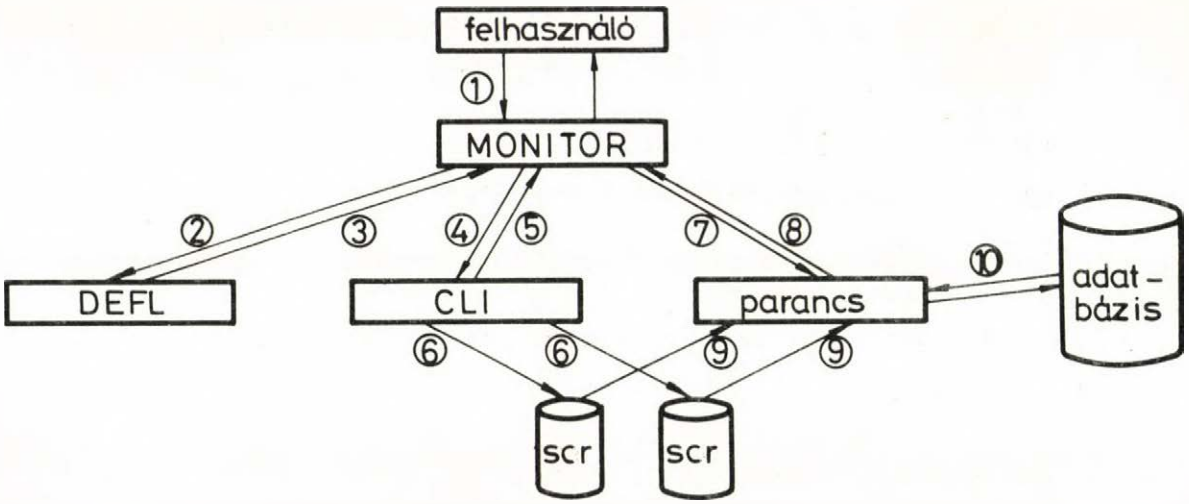
A fenti lépéseket csak egyszer, az adatbázis létrehozásakor kell elvégezni.

e/ A file-on való manipulálás szubrutinok segitségével történik, amelyek FORTRAN-ból és COBOL-ból hivhatók. Egyszavas integereket használ, ezért a $FTNU kártyán az S opció használata kötelező. A felhasználói programhoz hozzá kell füznünk a DDLA által lyukasztott block data szubrutint és a következő file-on lévő block data szubrutin:

```
$*DEF /O,,dsi2,ISDOS,DBBLK/
```

5.3 PSL/PSA

5.3.1 Felépitése



1 A felhasználó a monitort hivja meg.

2-3 A monitor először a DEFL taskot hivja meg, ami a kezdeti értékek beállitását végzi, és visszaadja a vezérlést.

4-5-6 A következő task a CLI /command language interface/. Ez szedi le a parancsok paramétereit, amelyeket azután két scratch file-ra ir ki.

7-8-9-10 A monitoron keresztül átadódik a vezérlés a megfelelő parancsnak. A parancsok felhasználják a scratch file-okat, ennek alapján irnak vagy olvasnak az adatbázisról. A hibás parancsokat a monitor kihagyja, STOP utasitásra terminál.

5.3.2 A PSL/PSA használata

Mint a 3.2 pontban emlitettük, az első lépés az adatbázis file létrehozása és inicializálása. Az adatbázis file-t kötelezően 2-es dsi-vel kell megnyitni.

```
$*DEF/A,,...        /    a DB file allokálása
$*DEF/O,,2,...      /    a DB file megnyitása

$*DEF/O,,3,ISDOS,PSADBT/
$*DEF/O,,dsi,ISDOS,DBLIBRARY/
$DBIN,dsi                          inicializálás
```

A PSA/PSL használatakor a következő task-hivásokat kell megtenni.

```
$*DEF/O,,2,.../          DB file megnyitása
$*DEF/O,,dsi,ISDOS,PSA/  PSA/PSL rendszer megnyi-
                         tása
$PSA,dsi
     vagy
$PSA,dsi/n/
```

Ez utóbbi hivást akkor használjuk, ha az adatbázisban lévő objektum-nevek száma /n/ > 420.

Az 1-21-es dsi-ket a jobon belül ne használjuk, kivéve a 2. és a 3. dsi-ket.

A PSA parancsoknál az INPUT=dsi, FILE=dsi és a PUNCH= dsi, paraméter megadások a leirással ellentétben egyelőre nem használhatók, a rendszer csak a megfelelő default értékekkel tud dolgozni.

A rendszer memóriaigénye:

```
IP utasitásnál 124 qp /62 K szó/
DPSL           120 qp
```

többinél max. 88 qp

A scratch igény az adatbázis file nagyságától függ.
/A belső SORT igényt is tekintetbe kell venni./

IRODALOMJEGYZÉK

1. Transcript of Subgroup Reports at the "State-of-the-Art of Systems Building" Workshop
ISDOS Working paper /IWP/ N° 70

2. An introduction to Computer-Aided Documentation of User Requirements for Computer Based Information Processing Systems
IWP N° 72

3. Automatic File and Module Design - A Description of Proposed Research
ISDOS Newsletter Vol. 7. N° 3

4. PSA Users Manual
IWP N° 90

5. PSA Command Descriptions
IWP N° 91

6. PSL/PSA User Manual
IWP N° 98

7. PSA Outputs
IWP N° 99

8. A Data Base Management System for PSA based on DBTG 71
IWP N° 88