AUTOMATIZOVANÉ ZPRACOVÁNÍ DOKUMENTACE

Ing. Jiří Prokop

1. Úvod

Vytvořit dokumentaci programového systému znamená popsat jednotlivé prvky systému (moduly, datové množiny) a vazby mezi nimi. Přitom všechny informace jsou poskytovány nejlépe podrobně, nejlépe je to nezbytné z hlediska potřeb toho, jemuž je tato dokumentace určena.

Vazby mezi moduly jsou dvojího typu:

1) vertikální vazby - mohou vzniknout na různých úrovních:
   a) na úrovni zdrojových modulů, na př. použití uživatelského makra, příkazu COPY, resp. % INCLUDE v PL/1.
   b) na úrovni zavádějících modulů (příkaz CALL)
   c) na úrovni provádění (makra LINK,LOAD,XCTL).
2) horizontální vazby - pod tímto pojmem rozumíme vazby mezi proveditelnými programy a množinami dat, vzniklé tím, že vstupními soubory jednoho programu jsou výstupní soubory jiných programů.

Cílem tohoto příspěvku je zabývat se hlavně popisem horizontálních vazeb, zejména možnostmi automatizovaného zpracování této části dokumentace. Ostatními úkoly při zpracování dokumentace jsem se zabýval v příspěvku /1/.

2. Vertikální vazby

Popisu těchto vazeb a využití počítače pro dokumentaci těchto vazeb byl rovněž věnován příspěvek /1/. Software, vytvořený pro účely dokumentování vertikálních vazeb, byl nezávisle rozšířen o programy, umožňující automatizovat sběr dat. To si vyžádalo rozšířit následujícím způsobem množinu evidovaných typů vazeb:

A B I Modul A obsahuje COPY B nebo % INCLUDE B.
A B M V modulu A je použito uživatelské makro B.
A B N B je jméno řídicí sekce v modulu A.
A B L Modul A obsahuje CALL B.
A B E A je entry modulu B.
A B X Modul A obsahuje XCTL, LINK nebo LOAD B.
A B J Zavádějící modul B vznikl kompilací a linkováním zdrojového modulu A.

A B D Neznamená typ vazby, ale požadavek zrušit vazbu od A k B.

Štítky typu X,J,D je dosud nutno ručně děrovat (není jich mnoho, protože vazby typu X jsou možné jen v Assembleru a vazby typu J se při dodržování konvencí platných v našem středisku vyskytovat nemají).

Štítky s vazbami typu M,I,N vytváří program, který čte zdrojové moduly, napsané v jazyce Assembler. Štítky s vazbami typu I,N jsou výstupem programu, který čte zdrojové moduly, napsané v PL/1 (tento program ovšem předpokládá dodržování určitých konvencí při děrování).

Štítky s vazbami typu L,E vytváří program, který čte zaváděcí moduly. Všechny zmíněné programy mohou číst vstupní moduly buď z knihovny nebo z magnetické pásky příslušného knihovního systému. Knihovní systém p. o zdrojové moduly, používaný v našem středisku, eviduje také příznak programovacího jazyka (viz /2/). Zpracovávat se mohou buď specifikované moduly nebo všechny, obsažené v knihovně, případně na pásce.

V souvislosti s těmito změnami bylo nutno upravit logiku programu, který čte štítky a vytváří graf vazeb.

Ostatní informace už lze nalézt v /1/. Stačí ještě poznamenat, že nyní už programátor může získat dokumentaci vertikálních vazeb svého programového systému zcela bezpracně, což značně příznivě ovlivnilo stupeň využívání softwarových prostředků pro dokumentaci vertikálních vazeb.

3. Horizontální vazby

Předchozí zkušenosti způsobily, že se hned na začátku zdálo nejdůležitější získat pokud možno co nejvíce informací bezpracně, což v tomto případě znamená získat je přímo čtením jobů nebo katalogizovaných postupů a rozborem příkazů jazyka pro řízení úloh v operačním systému OS. Není přitom nutné rozebírat všechny parametry příkazů JCL; postačí nám ty, které budeme při popisu vazeb typu "vstupní množina dat - program", případně program - výstupní množina dat" považovat za významné.

Je však nutno překonat dvě překážky:

1) Informace, získané čtením DD příkazu, nepostačují vždy k rozhodnutí, zda jde o vstupní nebo výstupní soubor.
2) Bez dalších vstupních informací nezachytíme vazby, vzniklé tím, že trvalý soubor, který je výstupem nějaké úlohy, je zároveň vstupem jiné úlohy; získáme jen vazby v rámci jedné úlohy.

Jak můžeme odstranit první překážku?
Možnosti jsou dvě:

a) Umístění speciální poznámky na kterýkoli štítek takového DD příkazu, u něhož bychom nemohli na základě informací získaných rozborem parametrů UNIT a DISP správně rozhodnout, zdali jde o vstupní nebo výstupní soubor. Abychom tuto poznámku odlišili od jiných, použijeme znak #, který se v textech nevyskytuje, a bezprostředně za něm 1 znak (I - input, O - output, U - update) a přidáme ještě znak N jako označení, že DD příkaz definuje buď nezajímavou nebo dokonce vůbec žádnou množinu dat, například, když je tento DD příkaz uveden jen pro možnost se na něj v dalších příkazech odvolat.
b) Další možností je dodržování takové konvence při volbě jmen souborů, která by dovolila už z údjména rozpoznat, zdali jde o vstupní nebo výstupní soubor. Pro tuto možnost se přimlouvá i okolnost, že už výrobcem firemního softwaru jsou do určité míry takové konvence zavedeny (STEPLIB, SYSPRINT).

Nechci však žádné nové konvence zavádět; Chci jen umožnit programátorům, kteří nějakou konvenci dodržují, aby ji mohli definovat v poli PARM (případně v souboru SYSIN) a aby na ni byl vzat zřetel. V realizovaném programovém systému se definují pravidla takové konvence (i výjimky z těchto pravidel) takto: Záznamy v poli PARM jsou oddělovány čárkou. První záznam každé skupiny záznamů začíná hvězdičkou, za níž následuje některé z písmen I, O, U, N. Dalšími záznamy skupiny jsou specifikována údjména, která mají mít atribut podle prvního záznamu skupiny. Záznam může mít nejvýše 8 znaků, je-li kratší, je doplněn tečkami, jestliže poslední uvedený znak byla tečka, nebo mezerami v opačném případě. Tečka znamená, že na odpovídajícím místě může být libovolný znak, tedy například záznam
PARM='*I,..I..,*O,O..,VYSTUP,*N,..II.'
znamená, že údjména, která mají na 2. místě znak "I", budou

považována za ddjména souborů vstupních (s výjimkou těch, která mají "I" také na 3. místě), ddjména začínající písmenem "O" (i když je na 2. místě "I") a ddjméno VYSTUP jsou považována za jména výstupních souborů, ddjména, která mají "I" na 2. i 3. místě, představují soubory nevýznamné, které nebudou v tištěných schematech zpracování zobrazeny.

Obě možnosti lze kombinovat, při čemž priorita bude mít poznámka. Celý postup rozhodování bude vypadat takto: Jestliže je množině dat (ať explicitně nebo implicitně) přiřazeno takové zařízení, které umožňuje jen vstup (nebo jen výstup), rozhodne se na tomto základě. Nebylo-li možno rozhodnout tímto způsobem, využije se speciální poznámky, pokud je uvedena. Není-li, pak je možno využít konvence, pokud je pro toto ddjméno nějaká definována. Konečně, není-li dosud rozhodnuto, rozhodne první subparametr parametru DISP: NEW implikuje OUTPUT, MOD UPDATE, ostatní údaje INPUT. Kombinace NEW, DELETE však způsobí označení množiny dat jako nezajímavé.

A jak překonáme druhou překážku?
Zdá se, že nejsnazší cesta je seřadit na vstupu joby (katalogizované postupy) tak, aby pořadí odpovídalo logickému sledu jejich zpracování. Existuje-li více takových pořadí, vybereme libovolné z nich. Pak je možno zachytit i vazby mezi programy různých úloh. Postupujeme prostě tak, že trvalý vstupní soubor ztotožníme s naposled předcházejícím výstupním souborem stejného jména (dsname). To ovšem předpokládá, že různé soubory mají různá jména. Splnění tohoto požadavku by mělo být samozřejmostí.

Zachycením všech vazeb mezi soubory a programy jsme získali všechny hrany grafu, zobrazujícího náš programový systém. Uzly tohoto grafu představují buď množinu dat nebo provedení nějakého programu (nemůžeme říci program, protože jednomu programu může v tomto grafu odpovídat více uzlů). Graf může být dále zpracován počítačem.

Můžeme vytisknout schémata zpracování jednotlivých úloh včetně návaznosti mezi nimi. Příklad takového schématu je v příloze tohoto příspěvku. V příloze je i opis katalogizovaného postupu, jehož rozborem schéma vzniklo. Struktura schématu se

podobě struktuře HIPO diagramu: vlevo jsou vstupy, vpravo výstupy, uprostřed zpracování. Každé datové množině odpovídá jeden obdélník, jehož výška je různá podle počtu potřebných řádků. V nejhořejším řádku je uvedeno zařízení, pokud ho bylo možno zjistit (chybí na příklad u katalogizovaných souborů) a údajné. Na dalších řádcích je dsname a dále informace získané z komentářových štítků. Ve spodní hraně obdélníku jsou tři dvojčísli, které představují identifikační číslo, vytvořené pro účely automatizovaného zpracování. První dvojčíslí vzniká pořadovým číslováním jobů (katalogizovaných postupů), druhé pořadovým číslováním stepů v rámci jobu, třetí pořadovým číslováním DD příkazů v rámci stepu. Kromě jiných funkcí toto číslo zahrnuje spojení od výstupního souboru směrem k obdélníku, znázorňujícímu týž soubor na vstupu. V ukázce v příloze můžeme vysledovat, že výstup třídícího programu, který má DSN=SORTMES, je vstupem programu STKD3. Vstup třídícího programu (DSN=S.STK.M(0)) bychom našli mezi výstupními soubory programu, který je třetím stepem předchozího jobu. Pro snazší hledání jsou první dvě dvojčíslí vytištěna také v obdélnících, znázorňujících programy, hned vedle jména programu.

Aby vypovídací schopnost těchto schemat byla dostatečná, jsou programově zpracovávány také speciálně označené komentářové štítky (začínají posloupností znaků //*#), nejvýše však vždy jeden takový štítek za každým příkazem EXEC nebo příkazem DD, text převzatý z komentářového štítku se pak objeví v odpovídajícím obdélníku. U souborů je mu předřazena hvězdička, aby nemohlo dojít k záměně s dsname, kdyby dsname chybělo. Text těchto komentářových štítků by měl být co nejstručnější charakteristikou obsahu souborů nebo funkce programů.

V úlohách z oblasti hromadného zpracování dat mají velký význam struktury dat. Je vhodné ukládat deklaraci struktury věty do zdrojové knihovny jako samostatný člen a do zdrojového textu programu ji inkludovat. Tím se stávají programy do určité míry nezávislými na struktuře věty. V jazyce PL/1 je k tomu účelu možno použít příkazu % INCLUDE, ale pokud nemáme optimalizační kompilátor, který umožňuje používat tento příkaz bez makroprocesoru, stojí kompilace mnoho času. V našem středisku používáme generátor pro normalizované programování, a protože

do knihovny zdrojových programů ukládáme texty obsahující makropříkazy generátoru (zatímco zdrojové texty se generátorem vytvářejí jen dočasně pro kompilaci), vystačíme s příkazem inkluze tohoto generátoru. Pro důležitost datových struktur je umožněno uvádět v komentáři na DD štítku i jméno členu zdrojové knihovny, kde je uložena deklarace struktury věty. Poznámka má tvar buď #, jméno nebo #a, jméno, kde a je jeden ze znaků I, O, U, N. Jméno členu je pak uvedeno v posledním řádku s předtištěným textem "VETA:".

Dalším zpracováním můžeme získat výstupní soubory, obsahující

- seznam jmen všech programů, tvořících náš programový systém
- seznam jmen všech členů, obsahujících deklaraci struktury věty

Tyto soubory (jejich výstup je volitelný tím, že uvedeme odpovídající DD příkaz) jsou určeny pro další automatizované zpracování, na př. seznam programů může být použit na vstupu programového systému pro dokumentaci vertikálních vazeb.

Další možnosti automatizovaného zpracování, o kterých se zmíním již jen velmi stručně, jsou však zatím ve stadiu programování nebo ověřování. Je možno vytvořit graf návaznosti mezi jednotlivými joby, z něj pak pro obsluhu počítače vytisknout operogram zpracování. Konečným cílem je vytvořit propojení mezi systémem automatizovaného zpracování dokumentace a systémem automatizovaného řízení provozu, který je v současné době rovněž ve stadiu ověřování.

4. Závěr

Programový systém pro dokumentaci horitontálních vazeb je v popsaném rozsahu v našem středisku k dispozici od začátku letošního roku. Během té doby posloužil ke zpracování dokumentace dvou subsystémů, které byly dokončeny koncem minulého roku. Jeden je svým rozsahem menší (6 jobů, 11 programů), druhý je průměrného rozsahu s 10 joby a 16 programy (odtud jsou naše ukázky). V obou případech byla získaná schemata zpracování správná, aniž bylo potřeba doplňovat jakékoli informace. Jen kvůli větší sdílnosti byly použity komentářové štítky za každým příkazem EXEC. Také možnost definování konvence pro ddjména byla využita jen k tomu, aby se netiskly příkazy STEPLIB

a DD příkazy, zařazené pro možnost zpětných odvolávek, a před tiskem schematu zpracování úloh také DD příkazy SYSPRINT, SYSOUT a SORTLIB.

Zpracování na počítači s volbou všech popsaných výstupů trvalo 10 minut. Programy vystačí s 52 K paměti. Dále byl zpracován celý obsah knihovny PROCLIB (co se týče obsahu, představuje rozšíření SYS1.PROCLIB) a knihovny X.PROCLIB (obsahuje "agendové" JCL postupy). V případě knihovny PROCLIB je potřeba zhruba v polovině procedur doplnit označení, že jde o výstup, v případě knihovny X.PROCLIB je potřeba oprav podstatně nižší (asi 10 %).

U programových systémů z oblasti hromadného zpracování dat je tedy pro programátory velice přitažlivá možnost téměř bezpracného pořízení větší části dokumentace. Pro pracovníky provozu, kteří si museli na počítačem kreslená schemata zvyknout, je sice na jedné straně nevýhodou menší "čitelnost", způsobená nemožností odlišit v počítačem tištěných schematech různá zařízení použitím různých schematických značek (to je ostatně v operačním systému OS dosti problematické), na druhé straně však oceňují nižší, téměř nulový výskyt chyb a také jednotnost zpracování.

V současné době vstupují v platnost normy pro programovou dokumentaci (viz /3/). Bylo by možno s návrhy norem v mnohém polemizovat, ale v okamžiku, kdy vstoupí v platnost, stejně takové polemika ztrácí smysl. Naštěstí jsou požadavky uvedené v normách natolik všeobecné (ani tomu nemůže jinak být, jestliže normy mají platit pro všechny druhy výpočetních systémů), že se lze spokojit konstatováním, že automatizované vytváření dokumentace, jak jsem je zde popsal, není s žádnou z norem v rozporu, až na jedinou výjimku, kterou představuje norma "Specifikace". Zde je však možné vypracovat specifikaci s použitím přílohy, kterou představuje vytištěný graf, uvedený v příloze článku /1/. Jeho vypovídací schopnost je ostatně vyšší než u přílohy, uvedené v /3/.

Závěrem lze říci, i přes zatím jen nevelké počáteční zkušenosti, že jakmile jsou jednou vytvořeny s dostatečnou pečlivostí a důkladností softwarové prostředky pro vytváření doku-

sentace, je naprosto nemožné vrátit se zpět k pracnému psaní a kreslení, které se nám už teď zdá být dávnou minulostí.

Literatura:

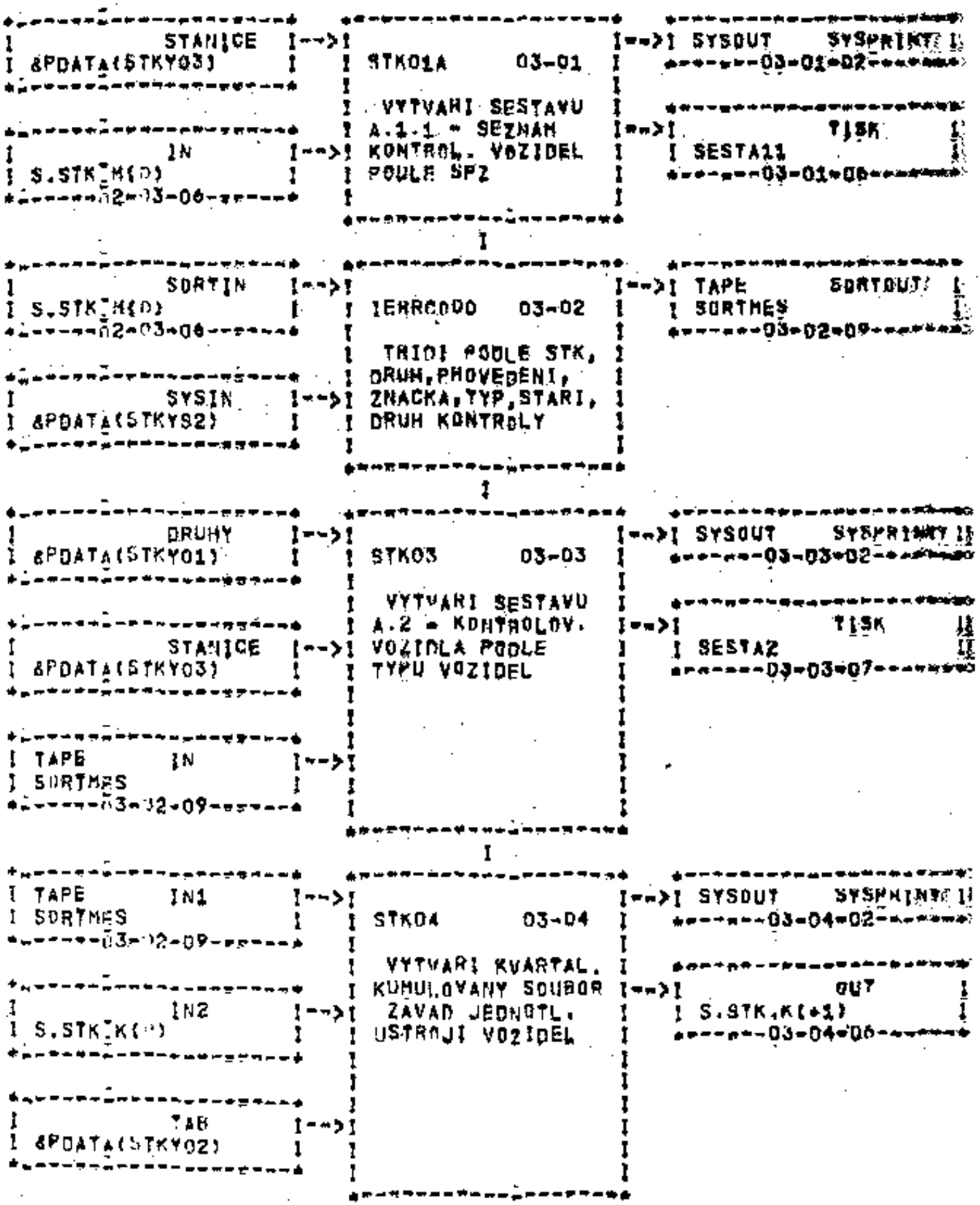
/1/ Prokop, J.: Využití počítače pro tvorbu dokumentace
Sborník Programování 79, Havířov 1979

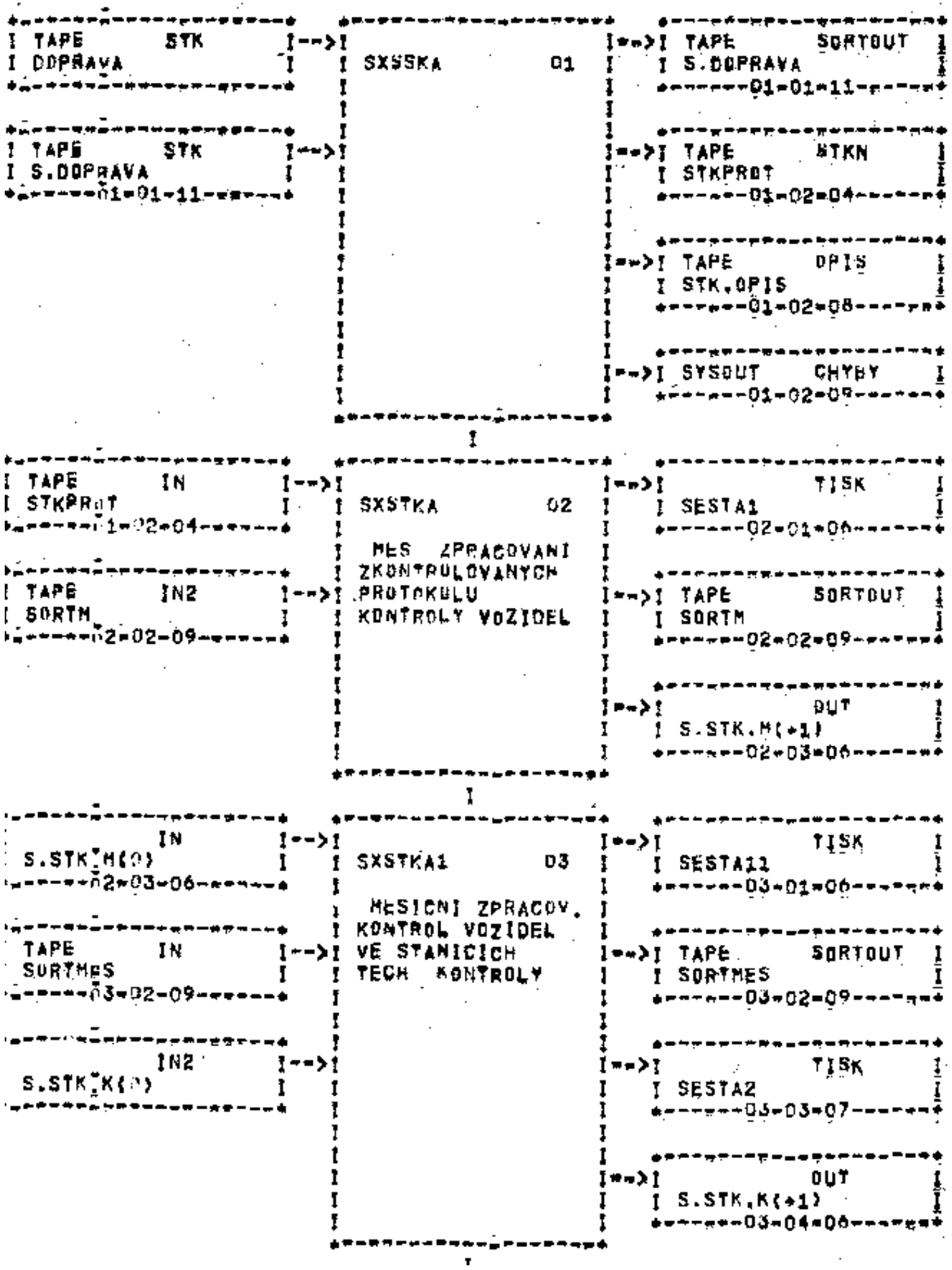
/2/ Prokop, J., Tloušť, V.: Knihovní systém pro zaváděcí moduly a data v operačním systému OS
Výběr informací z organizační a výpočetní techniky, č. 6/1979

/3/ Filinger, Č.: Normy pro programovou dokumentaci
Informační služba KP NOTO, č. 13 až 16/1981

Další aut. řeš. úlohou je přechod na vyšší rozliš. úroveň. Prvky systému jsou mysl jobs a trvalé soubory. Ze přejmenovaných vstupních souborů vybereme jen poslední, z výst. poslední. Identifikační čísla mnaše umožňují přechod na vyšší úroveň.

```
//*       PGM=SXSTKA1 ,SERIAL=005 M 25.02.82  PROKOP     X.PROCLIB
//SXSTKA1  PROC TAPE1=IP1204
//*# MESICNI ZPRACOV.KONTROL VOZIDEL  VE STANICICH      TECH. KONTROLY
//K1  EXEC PGM=STK01A
//*# VYTVARI SESTAVU A.1.1 - SEZNAM   KONTROL. VOZIDEL PODLE SPZ
//STEPLIB  DD DSN=&GOSET,DISP=(OLD,PASS)
//SYSPRINT  DD SYSOUT=A,SPACE=(TRK,2,RLSE)
//STANICE  DD  DSN=&&PDATA(STKY03),DISP=(OLD,PASS)
//IN       DD     DSN=S.STK,M(0),DISP=OLD
//REFER1   DD DSN=S.STK.K(-1),DISP=OLD
//TISK     DD     DSN=SESTA11,DISP=(NEW,KEEP),LABEL=2,
//                VOL=(,RETAIN,REF=*,REFER1),
//                DCB=(RECFM=FBA,LRECL=121,BLKSIZE=3509)
//K2       EXEC   PGM=IERRCOOO,PARM='CORE=70000'
//*# TRIDI PODLE STK,DRUH,PROVEDENI,  ZNACKA,TYP,STARI,DRUH KONTROLY
//SYSOUT   DD     SYSOUT=A,SPACE=(TRK,(1,1),RLSE)
//SORTLIB  DD   DSN=SYS1.SORTLIB,DISP=SHR
//SORTIN   DD     DSN=*.K1.IN,DISP=(OLD,KEEP),
//                DCB=(RECFM=FB,LRECL=170,BLKSIZE=1700)
//SORTWK01  DD  UNIT=5050,SPACE=(TRK,200,,CONTIG)
//SORTWK02  DD  UNIT=5050,SPACE=(TRK,200,,CONTIG)
//SORTWK03  DD  UNIT=5050,SPACE=(TRK,200,,CONTIG)
//SORTWK04  DD  UNIT=5050,SPACE=(TRK,200,,CONTIG)
//SORTWK05  DD  UNIT=5050,SPACE=(TRK,200,,CONTIG)
//SORTOUT  DD  UNIT=TAPE,DISP=(NEW,PASS),VOL=SER=&TAPE1,LABEL=2,
//                DSN=SORTHES,
//                DCB=(RECFM=FB,LRECL=170,BLKSIZE=1700)
//SYSIN    DD     DSN=&&PDATA(STKYS2),DISP=(OLD,PASS)
//K3       EXEC   PGM=STK03,COND=(9,LT,K2)
//*# VYTVARI SESTAVU A.2 - KONTROLOV. VOZIDLA PODLE     TYPU VOZIDEL
//STEPLIB  DD   DSN=&&GOSET,DISP=(OLD,PASS)
//SYSPRINT  DD SYSOUT=A,SPACE=(TRK,2,RLSE)
//DRUHY    DD     DSN=&&PDATA(STKY01),DISP=(OLD,PASS)
//STANICE  DD   DSN=&&PDATA(STKY03),DISP=(OLD,PASS),VOL=REF=*.DRUHY
//IN       DD     DSN=*.K2.SORTOUT,DISP=(OLD,PASS),
//                DCB=(RECFM=FB,LRECL=170,BLKSIZE=1700)
//REFER1   DD DSN=S.STK.K(-1),DISP=OLD
//TISK     DD     DSN=SESTA2,DISP=(NEW,KEEP),LABEL=3,VOL=REF=*.REFER1,
//                DCB=(RECFM=FBA,LRECL=121,BLKSIZE=3509)
//K4       EXEC   PGM=STK04,COND=((9,LT,K2),(9,LT,K3)),TIME=20
//*# VYTVARI KVARTAL.KUMUL.OVANY SOUBOR ZAVAD JEDNOTL.  USTROJI VOZIDEL
//STEPLIB  DD   DSN=&&GOSET,DISP=(OLD,DELETE)
//SYSPRINT  DD  SYSOUT=A
//IN1      DD     DSN=*.K3,IN,DISP=(OLD,DELETE)
//IN2      DD     DSN=S.STK,K(0),DISP=OLD
//REFER    DD     DSN=S.STK,K(-2),DISP=OLD
//OUT      DD     DSN=S.STK,K(+1),VOL=REF=*.REFER,
//                DISP=(NEW,CATLG,DELETE),DCB=FMODEL
//TAB      DD     DSN=&&PDATA(STKY02),DISP=(OLD,DELETE)
//*        KONEC PGM
```


STANICE
&PDATA(STKY03)
IN
S.STK,M(0)
SORTIN
S.STK,M(0)
SYSIN
&PDATA(STKYS2)
DRUHY
&PDATA(STKY01)
STANICE
&PDATA(STKY03)
TAPE
SORTMES
TAPE
SORTMES
IN1
IN2
S.STK,K(-1)
TAB
&PDATA(STKY02)
STK01A 03-01
VYTVARI SESTAVU
A.1.1 - SEZNAM
KONTROL. VOZIDEL
PODLE SPZ
SYSOUT SYSPRINT
TISK
SESTA11
IERRCODO 03-02
TRIDI PODLE STK,
DRUH,PROVEDENI,
ZNACKA,TYP,STARI,
DRUH KONTROLY
TAPE SORTOUT
SORTMES
STK03 03-03
VYTVARI SESTAVU
A.2 - KONTROLOV.
VOZIDLA PODLE
TYPU VOZIDEL
SYSOUT SYSPRINT
TISK
SESTA2
STK04 03-04
VYTVARI KVARTAL.
KUMULOVANY SOUBOR
ZAVAD JEDNOTL.
USTROJI VOZIDEL
SYSOUT SYSPRINT
OUT


TAPE STK DOPRAVA
TAPE STK S.DOPRAVA
01-01-11
SXSSKA 01
TAPE SORTOUT S.DOPRAVA
01-01-11
TAPE STKN STKPROT
01-02-04
TAPE OPIS STK.OPIS
01-02-08
SYSOUT CHYBY
01-02-09
TAPE IN STKPROT
01-02-04
TAPE IN2 SORTM
02-02-09
SXSTKA 02
MES ZPRACOVANI ZKONTROLOVANYCH PROTOKOLU KONTROLY VOZIDEL
TISK SESTA1
02-01-06
TAPE SORTOUT SORTM
02-02-09
OUT S.STK.M(+1)
02-03-06
IN S.STK.M(0)
02-03-06
TAPE IN SORTMES
03-02-09
IN2 S.STK.K(0)
SXSTKA1 03
MESICNI ZPRACOV. KONTROL VOZIDEL VE STANICICH TECH KONTROLY
TISK SESTA11
03-01-06
TAPE SORTOUT SORTMES
03-02-09
TISK SESTA2
03-03-07
OUT S.STK.K(+1)
03-04-06