Ing. Bohumil Šavčík
ČKD Praha, Výpočetní centrum

PROVÁDĚNÍ ZMĚN V ROZSÁHLÝCH PROGRAMOVÝCH SYSTÉMECH

Změna je život. Všechno živé se neustále vyvíjí, podléhá změnám. Lidé, pracoviště, prostředí jsou v neustálém vývoji, ani jeden den lidského života neprobíhá stejně, stejně tak ani jeden den na pracovišti, v závodě nebo podniku se nikdy neopakuje. Programové informační systémy jsou technická díla, která slouží k řízení "živých" objektů - provozů, závodů, podniků, správních objektů apod. Jsou úzce spojeny s nejrůznějšími obory lidské činnosti, nejen s výrobou a správou, ale i s výzkumem, vzděláváním, lékařstvím aj. A právě pro toto úzké spojení s životem nemohou tyto systémy zůstat beze změn. Bylo by jistě možno říci, že dokonalý programový systém by měl být tak univerzální, že by měl zachytit i řadu změn, které probíhají v objektech, jež jsou předmětem jeho práce. Tato námitka má sice určité oprávnění, avšak nic nelze vytvořit tak dokonale, aby to bylo stoprocentně dokonalé, nehledě na otázku nákladů na vývoj "dokonalého" systému.

1. Klasifikace změn programových systémů, příčiny změn

Abychom upřesnili význam výrazu změna programového systému, budeme za takovou změnu považovat jenom změny na dohotovených programových systémech, u kterých již skončil

vývoj a byly předány k užívání. Změny, které se odehrávaly před předáním programového systému do užívání možno začlenit do práce na vývoji systému.

Důvody ke změnám mohou být dvojího druhu - vnější a vnitřní.

Vnější důvody jsou takové, k nimž byl dán podnět z míst mimo tvůrčí programátorsko-analytický tým. Jde zpravidla o změny vyvolané novým přáním uživatele nebo zákazníka, změnou obecných předpisů, reorganizací podniku apod.

Vnitřní důvody jsou takové, které pramení ze snahy tvůrčího týmu nějakým způsobem vylepšit, zmodernizovat nebo zracionalizovat dosavadní způsob zpracování nebo jej přizpůsobit novějším hardwarovým nebo softwarovým podmínkám.

Jiné hledisko pro klasifikaci změn je hledisko četnosti změnou vyvolaných oprav programů. Je-li změnou zasažen jednotlivý program nebo velmi málo programů, lze hovořit o individuální změně. Je-li však změnou dotčen celý řetěz programů a navíc zpravidla změna u jednoho programu vyvolává nebo podmiňuje změnu u dalšího programu, pak jde o hromadnou změnu.

2. Individuální změny

S tímto druhem změn se každý programátor setkává velmi často. Důvodem ke změně může být změna výpočtového algoritmu, změna struktury vstupů, změna požadavků na výstup, snaha o racionalizaci programu apod. Charakteristickým rysem individuálních změn je nezávislost nebo velmi nízká závislost návazných programů na realizaci změny. I přes tuto nespornou výhodu individuálních změn nelze tyto změny podceňovat a zlehčovat je. Ve svém souhrnu totiž představují obrovské množství práce, které se každoročně vynakládá na jejich realizaci. Bylo již napsáno mnoho článků o tom, jak psát a konstruovat programy, aby byly nejen racionálně odzkoušeny, ale aby byly i dobře čitelné a snadno přístupné

změnám. Z tohoto důvodu se tento článek problematikou individuálních změn přímo nezabývá.

3. Hromadné změny

3.1 Problematika hromadných změn

Hromadné změny programových systémů charakterizuje vysoký stupeň závislosti realizovaných změn navzájem. Platí zde podmínka, že má-li mít programový systém po provedení změn nové funkční vlastnosti, je třeba provést všechny změny programů současně.

Jaké jsou typické příklady hromadných změn ? Je to např. změna struktury kmenového souboru dat, změna jeho organizace apod.

Mohla by být vznesena námitka, že problém hromadných změn je pouze součtem dílčích problémů individuálních změn. Tedy jestliže si umíme poradit s individuálními změnami, že s hromadnou změnou to je stejně snadné. To je však přílišné zjednodušování problému. Při provádění hromadných změn v programovém systému nutno dbát dvou požadavků:

- současné provedení změn v celém systému;
- racionalizace práce tam, kde se provádí opakovaná změna.

První požadavek je samozřejmou technickou nutností. Splnění druhého požadavku vyplývá ze snahy uspořit práci programátorů a usnadnit testování opravených programů. Vyskytuje-li se stejná změna v několika různých programech nebo na více místech jednoho programu, je žádoucí provádět ji jednotně s minimem vynaložené práce.

Programový systém pro zpracování dat je možno přirovnat k výrobnímu zařízení (linka, továrna) na výrobu hmotných předmětů. Na vstupu zařízení jsou zpracovávané materiály a polotovary, tj. procházejí výrobními operacemi, jejichž výsledkem je výrobek požadovaných funkčních a užitných vlastností. Stejně jako v jiných systémech, i zde dochází ke změnám. Mění se vstupní materiály (např. namísto

kovů plasty), mění se postup zpracování (namísto obrábění je použito tváření, leptání ap.) a mění se též samotné výrobní zařízení (z důvodů obnovy základních prostředků, náhrada méně produktivních zařízení za produktivnější).

Dojde-li na tomto výrobním zařízení k nějaké změně, může to mít rozsáhlé důsledky na čadě výrobních míst tohoto zařízení. Např. je-li nahražen ocelový odlitek výliskem z plastické hmoty, je nutno změnit obrábčcí nástroje pro tuto součástku, dále nahradit elektromagnetická upínací zařízení jinými, vyloučit operace svařování apod. To vše je třeba realizovat k jedinému okamžiku. Není možné, aby část výrobní linky byla přizpůsobena na původní, ocelovou součástku a jiná část už na novou součástku z plastické hmoty. A pokud jde o hospodárnost, je jistě žádoucí, aby se všude tam, kde dochází ke stejné nebo obdobné změně, řešila věc stejně nebo obdobně.

Toto srovnání může poněkud přiblížit problematiku změn v programových systémech. Stejně tak jako v materiální výrobě je i v programování třeba postupovat tak, abychom vyhověli požadavku na funkci a hospodárnost.

3.2 Provádění hromadných změn

Postup a organizace práce při provádění hromadných změn v programových systémech by měl zaručovat splnění dvou základních požadavků

1. požadavek na provedení změn k jedinému termínu,
2. požadavek na hospodárnost a úsporu programátorské práce.

Splnění těchto požadavků vyžaduje promyšlený postup, hlubokou analýzu navrhovaných změn, znalost programového systému, schopnost generalizace a dále organizační zajištění. Čím více času a práce se věnuje přípravě změn, tím méně práce je třeba vynaložit na vlastní provádění změn v jednotlivých programech.

3.2.1 Zajištění požadavku na provedení změn k jedinému termínu

S tímto požadavkem se zkušené pracovní skupiny setkaly již vícekrát a dovedou se s ním vypořádat. Praktické provedení spočívá v tom, že starý programový systém je třeba držet odděleně od nového. Starý systém se užívá v rutinním zpracování a na novém systému se provádějí úpravy. Pod pojmem nový systém se rozumí jen ta část programů nebo popisů práce (job control, job description), která podléhá změnám. Programy nebo popisy práce, které zůstanou nadále beze změn zůstávají v programové knihovně a používají se jak v rutinním zpracování, tak i při testování nového systému. Teprve po úpravě všech programů a po jejich úspěšném vyzkoušení se nové programy dostanou na programovou knihovnu a jsou předány do rutinního zpracování.

3.2.2 Zajištění hospodárnosti

Při analýze požadavků na změny je třeba vytypovat všechny změny, které bude třeba v programech realizovat. Po této analýze by měla následovat klasifikace a vyhodnocení navržených změn. Zkušený programátor se schopností zobecňování problémů může nalézt ve výčtu navržených změn celou řadu stejných nebo podobných změn. Hospodárnost vyřešení změn v programovém systému spočívá právě v tom, že opakované změny se řeší stejným způsobem. Kdyby se ke každé změně přistupovalo individuálně, znamenalo by to zbytečné vynakládání práce na vyřešení stejné nebo podobné věci a zvýšení nákladů na testování programů v nové verzi. Možná, že se tento požadavek zdá zbytečně zdůrazňovat, ale je nutno si uvědomit, že v praxi řeší úkol vždy několik programátorů. Z toho vyplývá nejenom požadavek na koordinaci jejich práce, ale zároveň i nutnost centrálního vyřešení některých změn a jejich přenesení do příslušných programů.

Praktické zkušenosti z našeho pracoviště ve Výpočetním centru ČKD PRAHA nás opravňují doporučit popsaný způsob

řešení změn jako výhodný.

3.3 Příklad realizace hromadné změny

Byl zadán úkol provést změnu organizace kmenového souboru v projektu MTZ. Stávající organizace souboru byla taková, že soubor obsahoval dva druhy vět:

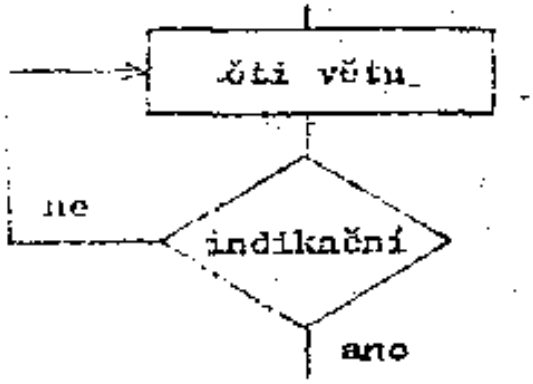
- indikační větu s popisem skladovaného materiálu;
- skladovou větu s údaji o skladování na určitém skladě.

Požadavek na úpravu projektu předepisoval rozdělení tohoto souboru na dva soubory. Kmenový soubor byl aktualizován v jediném programu, ale vstupoval do celé řady programů. Úkol tedy zněl změnit celý programový systém tak, aby byla akceptována nová organizace kmenového souboru.

Při analýze úkolu bylo zjištěno, že jsou dva druhy programů, jež mají kmenový soubor na vstupu:

- Do první skupiny patří ty programy, které pracují pouze s indikační větou.
- Ve druhé skupině jsou programy, které pracují s oběma typy vět.

Logická struktura programů z první skupiny byla v oblasti čtení velice jednoduchá (viz obr. 1).



Obr. 1 Výběr indikační věty z kmenového souboru materiálů

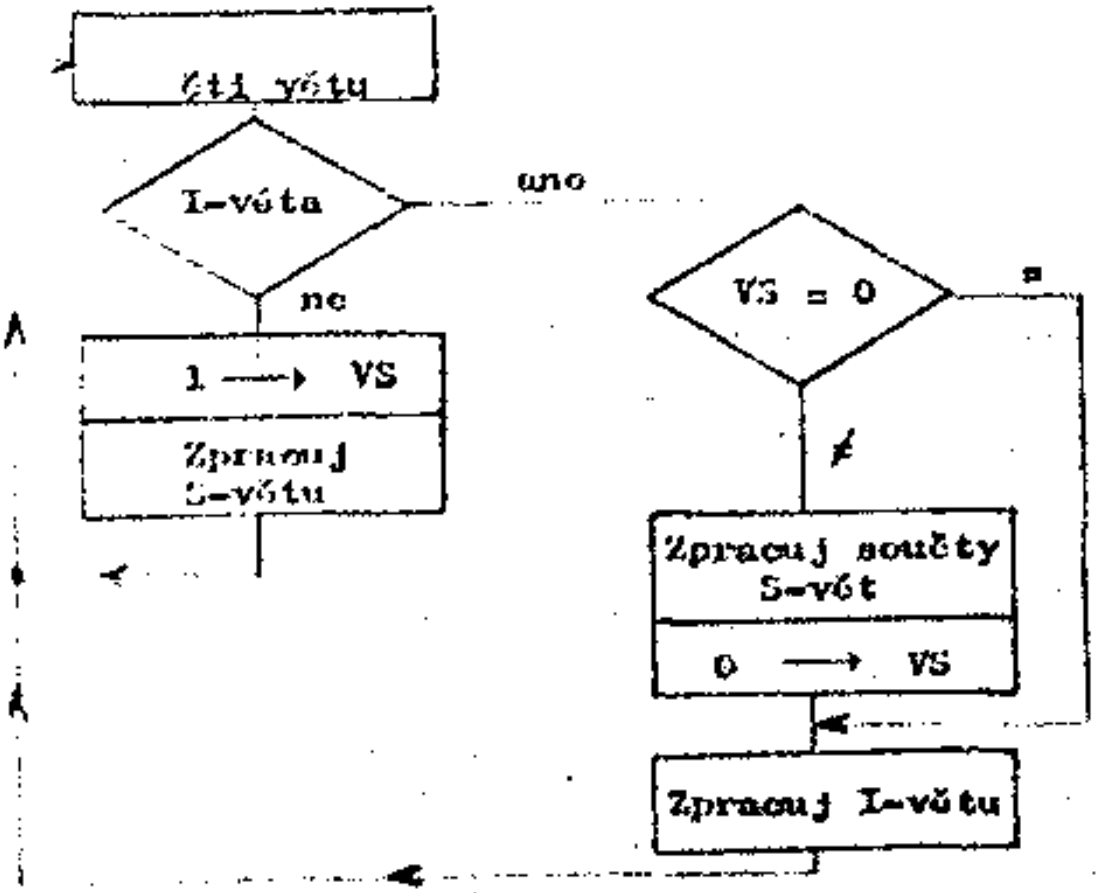
Proto i navržená úprava byla velice jednoduchá, spočívala ve vynechání výběrového bloku jako zbytečného(u některých programů s řídkým použitím se tato změna ani neprovádela), poněvadž na vstupu byl nyní soubor, který obsahoval pouze indikační věty.

Úprava programů druhé skupiny byla naopak náročnější. Každý z této skupiny programů četl kmenový soubor, v němž se vyskytovaly dva druhy vět. Logická struktura těchto programů byla v oblasti čtení kmenového souboru poměrně složitá (indikační věta je následována buď další indikační větou nebo skladovými větami, počet skladových vět k jedné indikační není předem znám). Změna organizace kmenového souboru znamenala u těchto programů značný zásah do logické struktury, i přes to, že nová organizace souboru by přinesla určité zjednodušení logické struktury.

Po zvážení situace bylo rozhodnuto vypracovat obecnou proceduru, která bude řídit čtení obou nových souborů tak, že bude v podstatě simulovat starou organizaci kmenového souboru. Aplikací této procedury v programech pak mohla zůstat zachována stejná logická struktura programů a celý problém se zúžil pouze na nahrazení příkazu READ příkazem CALL PROCEDURA. Toto opatření přineslo nejen velkou úsporu práce, ale odstranilo i nebezpečí "nabourání" složitých programů.

Obr. 2 ukazuje jednu z možných variant běžného programu pro zpracování souboru materiálů v původní struktuře. Aby bylo možno lépe vystihnout výhody popsaného řešení, byly použity obrázky 3a), b), c). Na těchto obrázcích je zjednodušeně nakresleno dvojí řešení popisovaného problému. Nutno podotknout, že obrázky používají poněkud nestandardní formy znázornění, avšak budou snadno srozumitelné, jestliže si uvědomíme, že obr. 3a) vystihuje v poněkud jiném zakreslení stejný problém jako obr. 2. Obr. 3a) tento problém zachycuje v poněkud obecnější úrovni. Z obrázků je vidět naprostou shodu struktury programu před provedením změny - obr. 3a) a po provedení změny - obr. 3c).

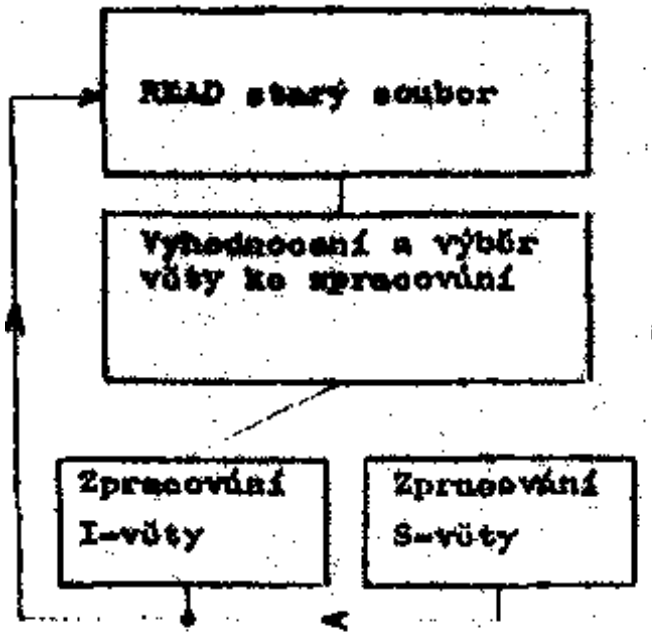
Z uvedeného příkladu je možno vidět, že k tomu, aby byl s úspěchem vyřešen i takovýto, poněkud složitější problém, je nutná nejen počáteční hlubší analýza, ale i pevné centrální řízení změn projektu, stejně jako jistá úroveň abstrakce a zobecnění řady podobných problémů.



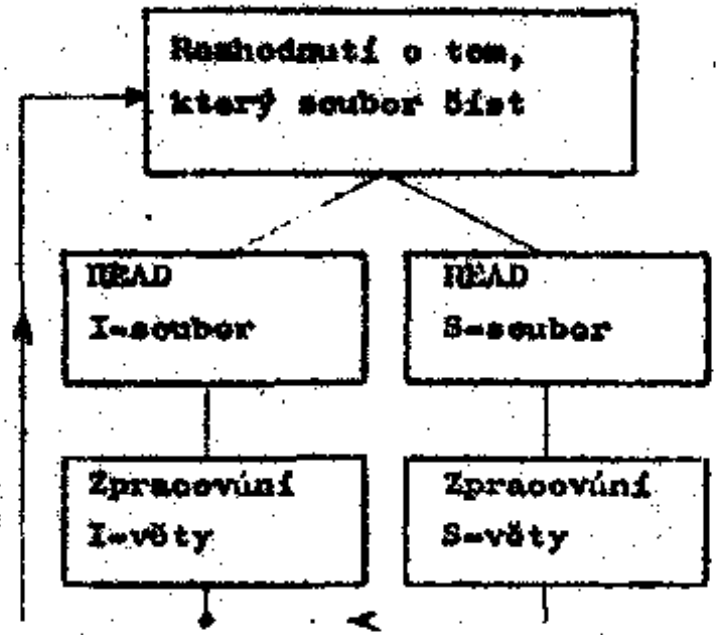
Obr. 2 Struktura programu pro čtení kmenového souboru materiálů o dvou typy vět.

3.4 Další aplikace

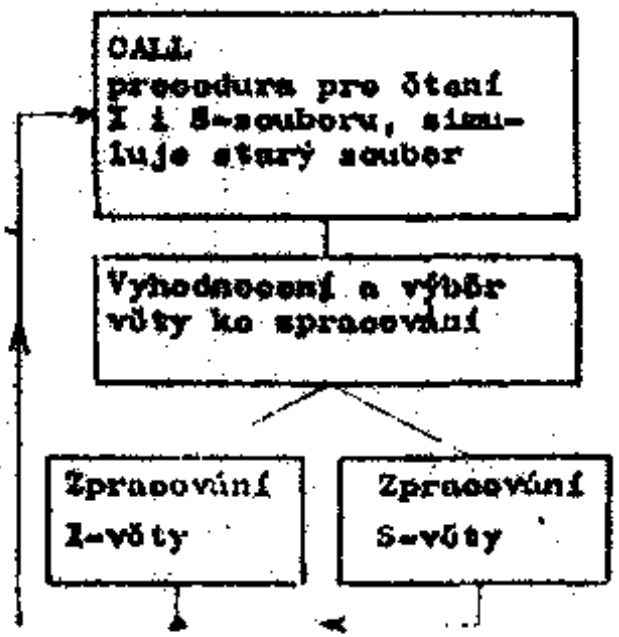
Podobné případy se v praxi mohou vyskytnout častěji, zejména při změnách struktury věty souboru nebo při změně organizace souboru. Velmi často dochází ke změně organizace souboru tehdy, přechází-li se k uložení souboru na magnetických páskách k uložení na disky. Při diskovém uložení se nabízejí nové formy organizace, které na magnetických páskách nejsou možné. Velmi častým jevem je zde sdružování souboru. Při práci s takovým souborem se pak přímo nabízí nahradit původní příkaz READ příkazem pro přečtení a expanzi věty do definované struktury, jak to ukazuje obrázek 4.



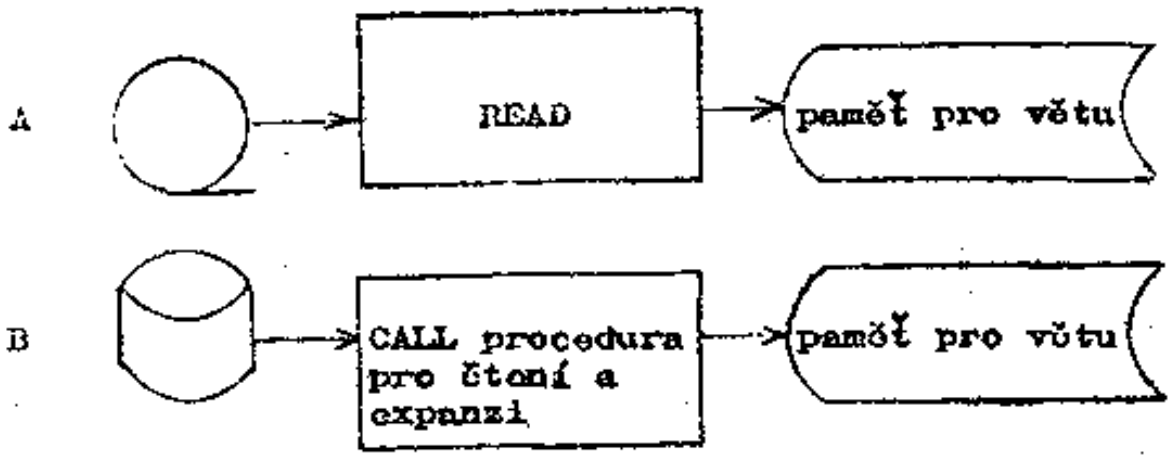
Obr. 3a)
Původní struktura programu, čtení souboru s dvěma typy vět.
Tento obrázek vyjadřuje jinou formou stejný problém jako obr. 2.



Obr. 3b)
Struktura programu pro čtení dvou souborů, jak by vypadala při navrhování nového programu.



Obr. 3c)
Nová struktura programu pro čtení dvou souborů, jak vypadá po aplikaci procedury pro simulaci staré organizace souboru.



Obr. 4 Přechod z MP souboru (A) k diskovému uložení souboru s komprimovanými větami.

4. Některé další poznatky a zkušenosti

4.1 Strukturované programy

Strukturované programy mohou v mnohém usnadnit celou práci spojenou s prováděním hromadných změn. Platí zde zkušenosti řečené již mnohokrát, že strukturovaný program je čitelnější a snadno přístupný změnám. Programátorské kolektivy, kterým tento poznatek není neznámý, by si měly uvědomit, že uplatňováním zásad strukturovaného programování při tvorbě nových programů usnadňují do budoucna jakékoliv změny programů, tedy i hromadné změny celého systému.

Určitá standardizace postupů rovněž usnadní práci při změnách programů. Je-li celý programový systém "ušit" podle stejného střihu a navíc dobře dokumentován, pak je mnohem snadnější orientovat se v jednotlivých programech a navrhovat řešení změn. Tento aspekt je třeba zvlášť zdůraznit u rozsáhlých projektů, u kterých se na programování podílí větší počet programátorů.

4.2 Problémy s jazyky nižší úrovně (assembler aj.)

Ve Výpočetním centru ČKD PRAHA je instalován počítač ..RA 1305 a ICL 1903T. Oba počítače jsou navzájem plně ..ompatibilní a jsou mezi sebou úzce hardwarově propojeny.

Programovacími jazyky vhodnými pro oblast zpracování dat jsou COBOL a PLAN, což je jazyk třídy assembler. První počítač serie ICL 1900 byl v našem středisku instalován před 13 lety a tehdy byl po několik let jediným programovacím jazykem pro zpracování dat assembler. Tak se stalo, že největší projekty byly naprogramovány v assembleru. Dnes je situace taková, že noví programátoři se učí pouze COBOL, který je pro výuku i programování rychlejší, levnější a produktivnější. Nových programátorů přibývá a ti, kteří umějí alespoň na základní úrovni jazyk PLAN, jsou již dávno v menšině.

Vedle starších projektů existuje již celá řada nových, programovaných v COBOLu. Srovnáním obtížnosti a pracnosti provádění změn v cobolských a assemblerovských programech vychází jasně najevo přednosti COBOLu. Mezi hlavními možno jmenovat

- jednodušší realizace změn, protože programy jsou mnohem čitelnější a přehlednější;
- možnosti využívání centrálního popisu souborů;
- snazší personální zajištění.

Z vyjmenovaných předností stojí za pozornost otázka centrálního popisu souborů. S jeho využíváním jsme sice nemáme velké zkušenosti, avšak zdá se nám, že s jeho pomocí bude v mnohém usnadněna i problematika hromadných změn.

4.3 Centrální popis souborů

Je to knihovna popisu souborů uložená na disku. Je centrálně spravovaná pro každý programátorsko-analytický tým jedním knihovníkem, který jediný smí dopisovat nové soubory a provádět změny. Knihovna je dostupná všem programátorům, kteří si z ní jednoduchým příkazem kopírují požadované popisy souborů do svých programů.

Dojde-li ke změně ve struktuře věty souboru, pak se celá záležitost realizace této změny do programů usnadní. Předně se změna promítne do knihovny centrálního popisu

souborů a pak se znovu přeloží všechny programy, které s daným popisem pracují. Tím se jednak ušetří programátorská práce s přepisováním popisu souborů, jednak se zajistí důsledné a naprosto shodné promítnutí úprav při změnách.

Závěr

Každá analyticko-programátorská skupina se při své práci čas od času dostane před úkol provést v programovém systému hlubší změny. Řešení takového úkolu je vždycky náročné a vyžaduje důkladnou přípravu, zvážení možných cest řešení a tvůrčí sjednocení pracovního kolektivu.

Zkušenosti popsané v této stati mohou napomoci při řešení podobných úkolů. Bylo by ku prospěchu věci, kdyby tento příspěvek vyvolal širší diskusi a dal podnět k publikování zkušeností z jiných pracovních kolektivů.