Ing. Josef Sláma
UNICHEM Pardubice

POZNATKY ZE ZAČLENĚNÍ METODY COMPOSITE DESIGN DO PROCESU TVORBY PROGRAMOVÉHO VYBAVENÍ

Úvod.

Příspěvek, který je rozdělen do tří částí, pojednává o některých poznatcích, získaných při pokusu o nahrazení obecné metody HIPO metodou COMPOSITE DESIGN ve fázi návrhu struktury jednotlivého programu. První část vymezuje pro účely tohoto příspěvku fázi "Návrh struktury jednotlivého programu" v rámci zjednodušené celkové metodiky pro tvorbu projektu subsystému ASŘ. Druhá a třetí část se zabývají metodami, použitými pro návrh struktury jednotlivého programu a zkušenostmi z jejich praktického provozování. Druhá část pojednává o obecné metodě HIPO, třetí část o specielní metodě COMPOSITE DESIGN.

Vymezení fáze "Návrh struktury jednotlivého programu"

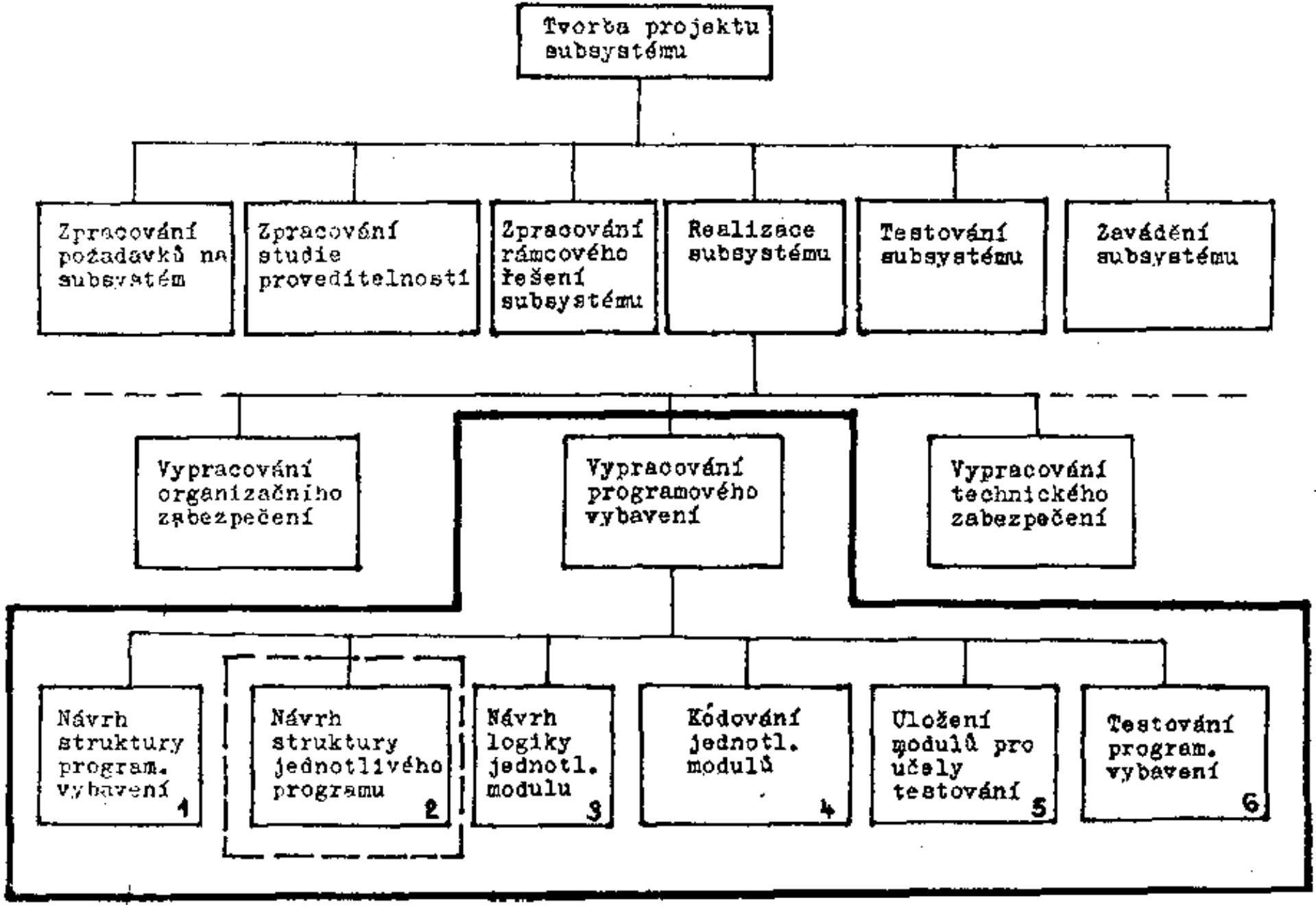
Před několika lety jsme začali postupně provádět drobné pokusy se zavedením některé z IPT technik do každodenní praxe. Od těchto pokusů jsme však poměrně záhy upustili, neboť jsme zjistili téměř naprostou marnost našeho počínání. Ačkoli nás každá technika ujišťovala, že může být použita samostatně bez ohledu na ostatní, při praktickém ověření bylo brzy zřejmé, že pouze zasazená do přesně vymezeného místa její působnosti v rámci ucelené metodiky může přinést užitek. Jelikož nepřicházelo v úvahu, abychom vedle plánovaných projektů různých aplikací pracovali ještě na projektu metodiky, rozhodli jsme se tu-

to rámcovou metodiku někde vyzískat. Metodika měla zhruba pokrývat činnost znázorněnou na obr. 1 společně s jejími vstupy a výstupy.

Obr. 1

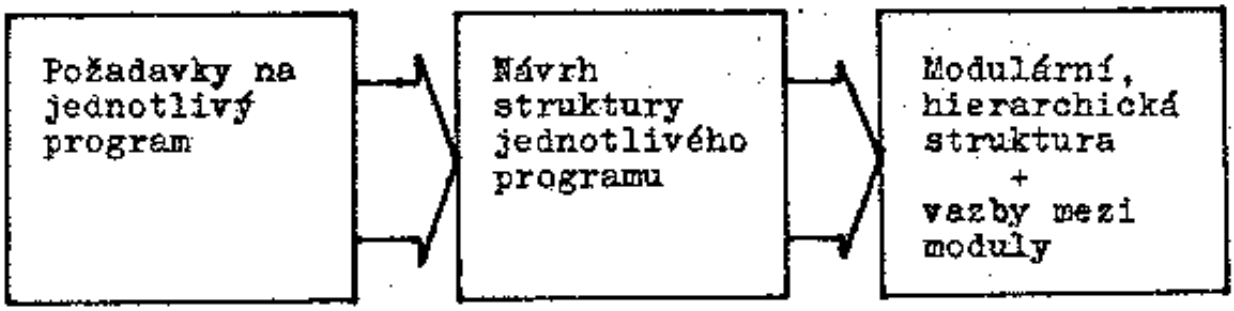
Představu jsme měli asi takovou, že tato metodika by měla být strukturovaná do snadno pochopitelných, relativně samostatných činností s přesně vymezenými vstupy a výstupy, organizovaných nejlépe do hierarchického uspořádání. Pro jednotlivé činnosti aby byly vypracovány dílčí postupy nejlépe na základě IPT technik, které by umožňovaly racionálně transformovat příslušné vstupy na požadované výstupy. Metodika jako celek by měla být podporována uceleným dokumentačním systémem. Bylo nám jasné, že takovouto metodiku vytvořit a prakticky ověřit trvá zřejmě léta. Na základě provedeného průzkumu jsme se měli možnost seznámit s metodikou vypracovanou v oddělení metodiky projektů koncernové účelové organizace CVTES Českých energetických závodů Praha, která nad očekávání úspěšně vyhovovala našim představám. Tuto metodiku jsme kompletně přebrali, při čemž nám pracovníci tohoto oddělení ochotně pomohli do začátku. Na obr. 2 je schematicky a značně zjednodušeně znázorněna část metodiky, upravená pro účely tohoto příspěvku. Zhruba silně orámovaná část byla převzata od CVTES, ostatní činnosti představují pro lepší názornost okolí, ve kterém metodika působí.

Po nějakém čase praktického používání, kdy jsme již metodiku jako celek dostatečně vstřebali a naučili se jí rutinně využívat, odvážili jsme se uvažovat nad tím, že při konkrétní aplikaci v našich podmínkách některé dílčí postupy umožňují


Tvorba projektu subsystému
Zpracování požadavků na subsystém
Zpracování studie proveditelnosti
Zpracování rámcového řešení subsystému
Realizace subsystému
Testování subsystému
Zavádění subsystému
Vypracování organizačního zabezpečení
Vypracování programového vybavení
Vypracování technického zabezpečení
Návrh struktury program. vybavení 1
Návrh struktury jednotlivého programu 2
Návrh logiky jednotl. modulu 3
Kódování jednotl. modulů 4
Uložení modulů pro účely testování 5
Testování program. vybavení 6

pracovat s velkým pohodlím, zatímco jiné již s pohodlím menším. Začali jsme si proto zaznamenávat, jak na nás jednotlivé dílčí postupy při praktickém používání působí s tím, že při negativním působení bychom třeba mohli někdy v budoucnu učinit pokus tyto postupy modifikovat či nahradit zcela novými. Nutno ovšem zdůraznit, že již samotný vznik těchto myšlenek je podmíněn existencí modulárně postavené základní metodiky s přesně vymezenými činnostmi jednotlivých "modulů" a s přesně definovanými vazbami mezi nimi.

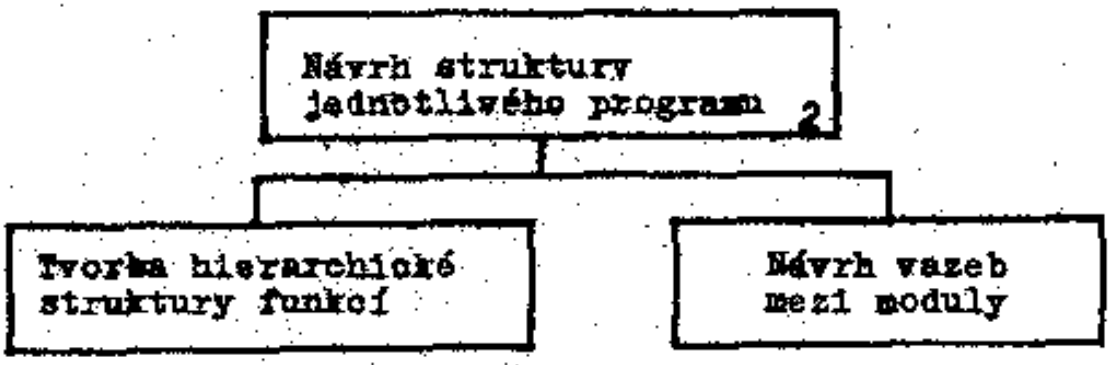
Díky šťastné náhodě se nám dostala do ruky publikace COMPOSITE DESIGN viz seznam literatury, pojednávající o dílčí, speciální metodice, podporující činnost označenou na našem schematu obr. 2 číslem 2 /Návrh struktury jednotlivého programu/ a ohraničenou čárkovaným obdélníkem. Pro úplnost uvádíme na následujícím obrázku též možné vstupy a výstupy této činnosti.



K tomuto tématu jsme měli též možnost přečíst si příspěvek otištěný ve sborníku ze semináře "Programování 80". Na základě těchto podkladů jsme se rozhodli učinit první pokusný zásah do převzaté metodiky a nahradit pro činnost "Návrh struktury jednotlivého programu" doposud používanou obecnou metodu HIPO metodou COMPOSITE DESIGN, která dle dostupných informací byla zpracována přímo pro podporu výše uvedené činnosti.

Návrh struktury jednotlivého programu pomocí HIPO metody.

Dle obecné metody HIPO sestává návrh struktury programu v podstatě z následujících dvou činností:



Přednosti použití HIPO metody jsou všeobecně známé a nebudeme je zde opakovat. Spíše se zaměříme na připomínky, které jsme si během praktického používání poznamenali a které dali vlastně popud k tomu, že jsme se rozhodli buď metodu HIPO modifikovat pro tuto činnost, nebo získat metodu jinou. Znovu opakujeme, že připomínky většinou vyplývají z faktu, že HIPO metoda je metodou obecného použití, nikoli metodou vypracovanou přímo pro návrh struktury programu. Vzhledem k omezenému rozsahu příspěvku, zaměříme se pouze na několik připomínek ze dvou oblastí - oblasti vlastního návrhu programové struktury a oblasti aktualizace programu, navrženého pomocí HIPO metody.

1. Oblast připomínek, vztahující se přímo k návrhu struktury programu.

a. Při tvorbě hierarchické struktury funkcí jednotlivého programu jsou souběžně po úrovních vypracovávány HIPO diagramy, znázorňující vazby mezi funkcemi. Tyto HIPO diagramy jsou velice pracné a zejména při návrhu prvních variant, kdy se hierarchická struktura neustále vyvíjí a přeorganizovává,

jsou mnohdy ihned po vytvoření již k nepotřebě. Pokud chceme identifikovat vazby mezi funkcemi, pak je nutné procházet svazek HIPO diagramů, což je značně nepřehledné.

b. HIPO technika nedisponuje objektivním souborem kriterií, dle kterých by bylo možno odlišit lepší návrh struktury programu od návrhu méně dobrého.

c. HIPO metoda nedisponuje konkrétním postupem, který by vedl projektanta krok za krokem k vytvoření dobré programové struktury, obsahuje pouze obecně platný postup při dekompozici jakéhokoliv systému, vlastní naplnění však nechává pouze na praktických zkušenostech projektanta.

2. Oblast připomínek, týkajících se otázky aktualizace programu, navrženého pomocí HIPO metody.
Pod pojmem aktualizace budeme zjednodušeně chápat pouze přidání nové funkce do stávajícího programu, nebo změnu v jeho modulech.

a. Přidání nové funkce do programu.
Přidání celé nové funkce do programu je poměrně snadné a nečiní zvláštních potíží, není však obvykle možno použít modulů stávajícího programu, neboť tyto nejsou již od počátku pro tento účel konstruovány.

b. Změna do modulů stávajícího programu.
Změna do modulů stávajícího programu může být vyvolána buď změnou některé datové struktury, nebo změnou logiky modulu.

A. Změna datové struktury.
Návrh struktury programu pomocí HIPO metody má za následek, že příslušná datová struktura vystupuje obvykle ve všech modulech, které pracují s některou její položkou. Proto je nutné provést změnu do všech modulů, ve kterých je struktura obsažena. Tyto moduly je ovšem velice nesnadné identifi-

kovat, neboť diagram hierarchické struktury v metodě HIPO neobsahuje schema datových vazeb mezi moduly a tak je nutné procházet celý svazek HIPO diagramů a nedopustit se přehlédnutí. Poněkud pomáhá vytvořit již při návrhu programu tabulku, ve které jsou společné datové struktury přiřazeny modulům, ve kterých se vyskytují. Dále je nutné mít na zřeteli, že datová struktura je přístupná jako celek všem modulům, ve kterých je obsažena, třebaže modul pracuje pouze s některou její položkou. Při změně struktury může dojít k posunu jednotlivých položek a pokud modul pracuje "tvrdě" dle relativní adresy položky ve větě, dochází k přepsání některé části struktury, což vede k chybám, které se projevují až v úplně jiných modulech. Odstranění těchto chyb je vždy značně pracné a časově náročné.

B. Změna logiky modulu.
Změna v logice modulu představuje změny do jeho kódu. Pokud se modul vyskytuje ve struktuře pouze jedinkrát, pak provedení změny nečiní problémy, neboť dokumentace modulu pomocí HIPO techniky je velice názorná a přehledná. Vyskytuje-li se však ve struktuře víckrát, pak je dle konvencí HIPO znázorněn všude tam, kam logicky ve struktuře patří, což může být u větších programů i na různých diagramech. Při změně je tedy nutno všechny tyto diagramy pečlivě projít, zjistit odkud všude je modul volán, za jakým účelem a pak teprve uvážlivě provést úpravu. Případným přehlédnutím některé vazby dochází k těžko identifikovatelným chybám. Jako určitou pomůcku jsme již při návrhu programu sestavovali tabulku pro moduly obecnějšího použití, ze které je zřejmé, ze kterých modulů je ten který modul volán.

Návrh struktury jednotlivého programu pomocí metody COMPOSITE DESIGN.

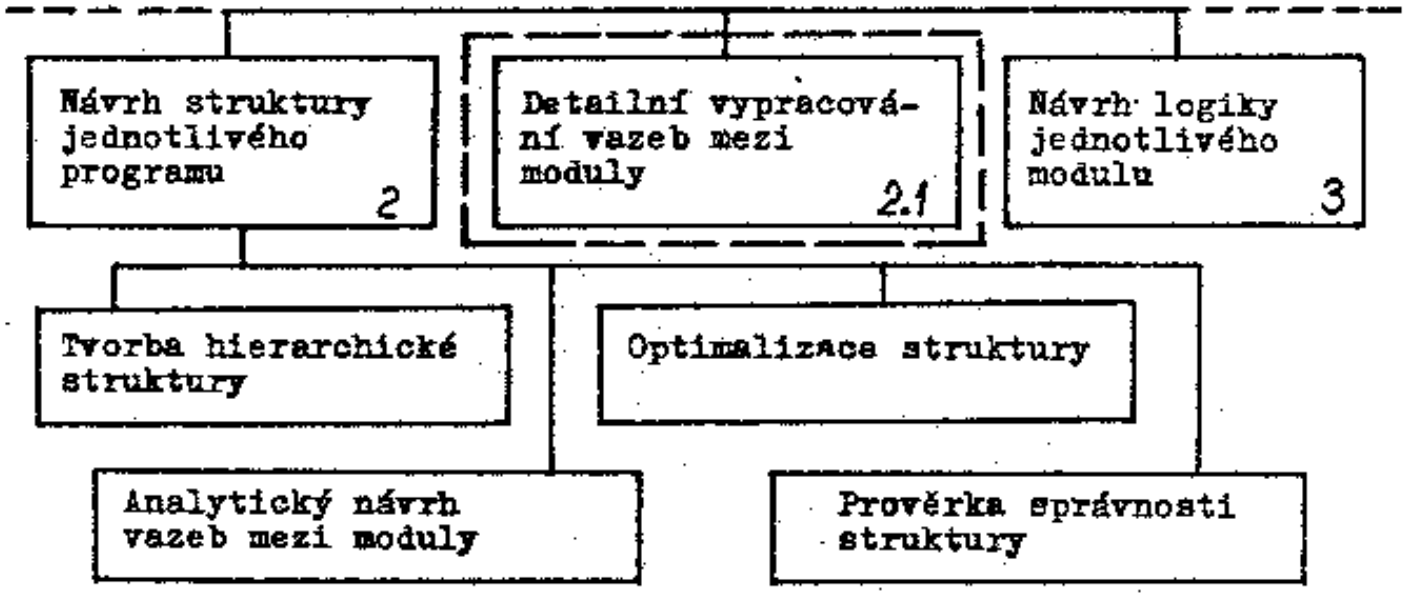
Z dostupných informací jsme si učinili představu, že na rozdíl

od metody HIPO, která je metodou obecného použití, metoda COMPOSITE DESIGN je metodou vypracovanou přímo pro podporu návrhu jednotlivého programu a tak jsme učinili pokus nahradit metodu HIPO pro tuto činnost metodou COMPOSITE DESIGN. Zkušenosti z prvního praktického použití přinesly následující poznatky v oblastech, které v tomto příspěvku sledujeme.

1. Oblast připomínek, vztahující se přímo k návrhu struktury programu.

a. Při tvorbě hierarchické struktury je vytvářena jednoduchá analytická tabulka vazeb mezi moduly na místo HIPO diagramů, čímž se dosáhne značného snížení pracnosti.

b. Metoda obsahuje jasný a kompletní postup při návrhu struktury programu na základě vypracovaných zásad pro tvorbu dobré programové struktury/ viz příspěvek Programování 80/. Základní činnosti jsou uvedeny na následujícím obrázku.



c. Součást metodiky tvoří následná optimalizace struktury na základě zásad pro hodnocení.

d. Velkým přínosem je rozdělení návrhu vazeb mezi moduly do dvou částí. Analytický návrh se provádí pouze v rozsahu nutném pro tvorbu hierarchická struktury, detailní návrh pak před návrhem logiky modulu. Tato činnost byla včleněna do činností vyšší úrovně jako činnost 2.1 mezi činnost 2 a 3/viz obr./

2. Oblast připomínek, týkajících se otázky aktualizace programu, navrženého metodou COMPOSITE DESIGN.

a. Přidání nové funkce do programu.
Velice snadné a navíc vzhledem k tomu, že jednotlivé ENTRY v modulech pracujících na jednotlivých datových strukturách představují jednoduché, samostatné funkce, je možné většinu z nich použít i pro novou funkci.

b. Změna do modulů stávajícího programu.

A. Změna datové struktury.
V případě dodržení metody a provedení optimalizace programové struktury, vyskytuje se jedna datová struktura v celém programu pouze v jednom modulu, ve kterém se změna provede.

B. Změna logiky modulu.
Vzhledem ke konvencím použitým pro znázornění programové struktury, vyskytuje se každý modul ve struktuře pouze jednou, spojnice vedoucí k tomuto modulu představují jeho volání z jiných modulů. Tím získáváme dobrou pomůcku pro stanovení rozsahu a účinnosti změn. U modulů vyšších úrovní, představujících v podstatě pouze řízení znamená změna logiky pouze změnu sekvence volání výkonných funkcí. Změna do modulů nižších úrovní znamená přidání či změnu ENTRY v modulu, což při elementárnosti funkcí nečiní problémy.

Závěrem těchto několika poznatků bychom chtěli konstatovat, že metoda, ačkoliv si při praktickém používání vynutila některé dílčí úpravy do základní metodiky, splnila naše očekávání a bude nadále pro návrh struktury jednotlivého programu používána.

Literatura:

1. MYERS G.J., Composite/Structured Design
2. NEPOVÍM a kol., Metodické pokyny OVTES k.ú.o
3. CIMBURA V., Problémy návrhu modulárních programů, sborník seminář "Programování 80" Ostrava DT CSVTS
4. IBM, HIPO - A Design Aid and Documentation Technique