ing. Jiří Prikner

POLDI - SONP Kladno

ZKUŠENOSTI SE SYSTÉMEM ANALÝZY A PROGRAMOVÁNÍ ZAVEDENÉM VE VS POLDI KLADNO

Po dodávce počítače IBM 370 se analýza a programování v našem VS dlouho vyvíjely živelně. V počáteční fázi se sice stanovily určité konvence o názvu programů, souborů, deklaracích a pod., ale to bylo po dlouhou dobu vše, co bylo závazně stanoveno. Úroveň analýzy a dokumentace byla různá a závisela na schopnostech a pořádkumilovnosti autora, úroveň programů byla též velmi individuální. S nárůstem počtu lidí a prací se situace pomalu stávala nepřehlednou, vývoj nových systémů byl živelný, nekoordinovaný, nekontrolovatelný a špatně dokumentovaný. Opravy starších programů se stávaly problémem, programy byly nepřenosné mezi programátory.

Za této situace jsme v poslední době byli nuceni učinit několik různých opatření.

1) <u>Plánování akcí.</u>

Základem veškeré činnosti VS se stala směrnice FŘ - Pokyny o racionálním využívání výpočetní techniky. (Poznámka: zkratka Gm - odbor řízení a výpočetní techniky).

Jaký je obsah směrnice?

- Sestavení ročního plánu odboru
- Způsob předkládání ostatních požadavků mimo plán.
- Povinná posloupnost | etap projekční činnosti.

Sestavení ročního plánu provádí podniková komise, jejímiž členy jsou nejdůležitější složky podniku. Komise schválí plán, ke kterému dá podklady Gm na základě technicko-ekonomických studií, které vypracuje zadávající útvar a koncepce rozvoje výpočetní techniky.

Ostatní neplánované požadavky se předkládají na požadavkových listech (viz příloha 1).Tyto práce nemají mít větší rozsah než 100 hodin projektní činnosti a v případě nedostatku kapacit mohou být odmítnuty. Do této skupiny patří též všechny požadované změny v dosavadních projektech a programech.

Pro plánované práce je stanovena tato povinná posloupnost projektových prací:

- Technicko-ekonomická studie - formulace celkového záměru zadavatele. Je nutná pro přijetí práce do plánu.
- Projektový úkol - po zařazení práce do plánu stanoví rozsah a působnost projektu a nezbytné předpoklady pro jeho splnění.
- Technický projekt (analýza). Obsahuje detailní popis vstupů, logiky zpracování, výstupů, užitých číselníků a kódů, organizační zajištění akce, technické prostředky a ekonomické zhodnocení.
- Prováděcí projekt (programování). Sem patří rozdělení subsystému do programů, modulů, pomocných souborů, tabulek a pod., a vlastní programy, včetně dokonalého odzkoušení každého modulu.
- Realisace projektu, t.j. uvedení do provozu včetně provozní dokumentace a vytvoření uživatelské příručky.

Všechny etapy jsou povinně oponovány.

2) Analýza problému a programování.

V 1. fázi analýzy spolupracuje analytik pouze se zadavatelem, shromažďuje od něj informace o jeho detail-

ních požadavcích a formuluje je do jakéhosi subsystémového polotovaru. V této fázi přistupuje do řešení odpovědný programátor úkolu, se kterým analytik navrhne způsob řešení úkolu na počítači a jeho hrubou modularizaci ve formě hrubých strukturálních schémat. Je přirozené, že v této fázi se stále spolupracuje se zadavatelem, zde se upřesňují a doplňují podklady pro programování. V této fázi je možno začít programovat některé moduly, čímž se zkracuje celková doba projektční činnosti.

V okamžiku praktického vyjasnění všech podstatných požadavků na zadavatele se přistupuje k rozdělení subsystému na programy a celá činnost se jakousi zpětnou vazbou znovu opakuje s ohledem na funkci jednotlivých programů, t.zn., že určité funkce programu, které jsou již známy se mohou modulovat a tím i programovat (např. znám považ tvar vstupních dokladů, kontroly, tvar chybových sestav, třídění hlavního souboru a pod., tudíž mohu programovat vstupní modul, kontrolní modul, modul chybové sestavy, změnový či párovací modul a pod., i když v této době ještě přesně neznám tvar výstupních sestav či způsob výpočtu ceny atd.).

Vlastní kritéria modularizace

Jak jistě víte, existuje spousta návodů a teorií o modulovaném a strukturovaném programování. Základem našich úvah byl článek Structured design z IBM System Journalu 2/74. Tento i následné prameny jsme převedli do následujících pravidel návrhu:

1) Závislosti mezi moduly by měly být pokud možno funkční, např.

   výpočet ceny : cena = f (parametrů)

   tisk součtované sestavy : tvar a obsah sestavy = f (doručených vět, požadovaného detailu)

   a pod.

2) Rozsah modulu by neměl překročit stránku výkonných příkazů.

3) Předávání dat modulu se musí dít přes argumenty a parametry; parametry mají být pokud možno jednoduché (elementární) proměnné.

4) Modul ai musí poradit s jakýmikoliv daty, t.zn. že musí být vždy přesně definována akce při dání jakýchkoli dat, třeba i STOP programu.

5) Pro každý tiskový soubor a sestavu musí být vytvořen zvláštní modul.

Přirozeně, tyto zásady se nechápou jako pracovní příkazy, ale jako doporučení. V některých případech, hlavně při změnách se neubráníme, aby program měl více než 60 řádek.

3) Systém zadávání programů

Program zadává analytik na formuláři (příloha 2) - Zadávací formulář programu. Hlavními informacemi je jméno programu a číslo interní zakázky. Tento formulář se po schválení denně děruje, číslo interní zakázky se kontroluje proti tabulce plánovaných interních zakázek či požadavkových listů a v případě nesouhlasu je odmítnut. Tímto vstupem se také provede zápis do souboru jmen programů. Přítomnost jména a neměnného čísla interní zakázky povoluje práci s programem. V kolonce zadání úkolu - stručný popis činnosti je stručný popis, který je dále rozepsán v přílohách.

Obsah příloh je zpravidla:

- modulové schéma programu se vstupy, výstupy, funkcemi modulů (které programátor může dále rozepsat)
- popisy vět souborů (standartní díl), se kterými se pracuje
- tvar sestav, třídění, součty na rastrovaném papíře.

. tvar vstupů, kontroly
. algoritmy výpočtů tam, kde je to potřeba (pro jednotlivé moduly).

4) Systém zadávání modulů

Odpovědný programátor, který dostane daný program prostuduje vazby, navrhne detailní modulové schéma, sestaví schéma vazeb mezi moduly. Na takto vytvořené moduly napíše zadávací list modulu (viz příloha 3), s tímto formulářem se provádí podobné akce jako s formulářem na zadávání programů a tím je povolena práce na modulu. Zadávací list modulu musí být vystaven i na změny již hotových a odladěných modulů. Zadání modulu má zpravidla tyto přílohy:

. popis vstupních a výstupních parametrů
. algoritmus řešení
. popis souborů, tiskových sestav, vstupů v modulu užívaných
. užité standartní deklarace z knihovny deklarací

Modul má mít tyto náležitosti:

.Každý modul (včetně řídícího hlavního programu) musí obsahovat popis modulu. Všechny informace popisu jsou povinné a pokud se nevyskytují, musí být proškrtnuty (popis viz příloha 4).
.Úprava modulu musí být čitelná a přehledná dvojice then-else, do-end a pod. psány pod sebou, obsah do-skupiny musí být obsazen vpravo od do-end, v programu musí být výrazně odděleny skupiny:
popis
deklarace
on-jednotky
vlastní program

a pokud pro to nejsou zvláštní důvody, nesmí se vyskytovat jinde.

- Je-li v modulu prováděna nějaká inicialisační akce při 1. vstupu do modulu, pak se na začátku oddílu vlastní program musí vyskytovat skupina příkazů:

```
if  PRVNI
    then do
        : PRVNI = '0'B;
          akce
        end;
```

kde proměnná PRVNI je povinná.

- Skoky goto jsou povoleny, ale jen tam, kde je zřejmá logika programu, na př.

```
CTI: READ ...
     IF podmínka
       THEN GOTO CTI;
```

Jinak se snažíme příkazu goto vyhnout a to z toho důvodu, že programu bez zbytečných goto jsou lépe promyšlené.

5) Ladění modularisovaného programu

Každý programátor modulu je povinen odladit svůj modul tak, aby prošel všemi větvemi programu, což u jednostránkového modulu nečiní zpravidla žádné potíže.

Ladící data k odladění modulu je povinen vymyslet si programátor modulu, v některých případech, kde je to účelné, může programátor ladit více modulů najednou (i když se vystavuje nebezpečí prodloužení ladění, např. ladí-li čtení DŠ a tisk chyb a pod.).

Jinak modulový programátor je povinen postupovat tak, že napíše testovací prográmek, kde

čte z DŠ parametry vstupní
tiskne je
volá testovanou proceduru
tiskne výstupní parametry

Modul musí správně fungovat,i ve speciálních případech (např. : vstupní soubor, užívaný v modulu je prázdný, chybová sestava prázdná, dodány vstupní parametry označující poslední vstup do modulu a pod.).

Ladění celého programu zaslouhuje zvláštní pozornost. I když jsou všechny moduly dokonale odzkoušeny, dochází k chybám typu : odpovědný programátor chybně zadal parametry, jsou rozpory v modulárních schématech, nepromyslel dobře speciální stavy.

Ladění celého programu provádí odpovědný programátor. Zde záleží na jeho zkušenostech, jak rychle dokáže chyby odhalit a odstranit.

Na ladění celého programu se zúčastní také analytik. Posuzuje, zda funkce programu je taková, jak bylo v analyse požadováno.

6) Podpůrné prostředky pro aplikaci modulárního programování

. Programovací jazyk. Je to základní prostředek, jazyk PL/I i COBOL umožňují modulární stavbu programů.
. Použití knihoven objekt-modulů. Modulové stavebnicové prvky ve tvaru object - modulů se snadno sestavují ve výsledný load-modul.
. Pružné vlastnosti linkage-editoru, který právě provádí spojování object-modulů ve výslednou load-formu. S volbou OVERLAY může linkage-editor dosáhnout toho, že v paměti je obsažena vždy jen aktuální, právě aktivní volací sekvence modulů.
. Třídění uvnitř programu. Můžeme použít několik SORTů za sebou v rámci jednoho programu.

. Systém práce se SYM a OBJ knihovnami a k tomu speciálně vypracované procedury, např.:
  UPDS - udd, change symbolické knihovny
  PL1KFC - překlad modulu za symbolické do object knihovny.
  PL1V - výpočet s loaderováním (t.j. sestavením load-programu z object-modulů do paměti).
  LISTDA - výpis rejstříku knihovny
  atd.

. uložení nepoužívaných symbolických a objektivních programů na klidových páskách s podpůrným systémem procedur na obsluhu tohoto systému.
. programové (systémové) zajištěné jedinečnosti jména modulu, který je ukládán na symbolickou knihovnu.
. zjišťování nákladů na strojový čas na jednotlivé interní zakázky a na spotřebu času jednotlivých programátorů podle zakázek.
. zajištění přehledu o produktivitě programátora, je to zajištěno z děrovaných zadávacích formulářů programu a modulu.
. Automatické vkládání data poslední změny na počátek symbolického programu. U každého příkazu je pak kódované datum opravy příkazu či datum zápisu příkazu (na sloupci 73-75).
. Automatické pořizování přehledu vazeb nadřazená - podřazená procedura a naopak.
. Automatické sledování úplnosti popisů modulu dle autorů modulů.
. Parametrizovatelné výpisy popisů modulu (možné výběry).
. Na některé funkce (tisky, vstupy. výběry, párování a pod.) je možno nasadit vyvinuté šablony, po případě generátory modulů.

Všechny tyto podpůrné prostředky usnadňují práci s modulovaným programem na všech úrovních analytik -

odpovědný programátor - modulový programátor a současně "dohlíží" na dodržování zásad modulárního programování.

7)Klady a zápory tohoto systému modulárního programování.

Klady:

. Urychlení termínu programovacích prací. Program se může dát psán několika lidem, kteří nemusí být hlouběji informováni o podstatě problému. Tím se lépe vytíží programátorské kapacity. I méně zkušení a kvalifikovaní programátoři mohou být zapojeni do složitých systémů.
. Zlepšení odhadu pracnosti jednotlivých modulů a tím i celého systému.
. Bezpečné a lacinné testování. Podle zkušeností nyní při záběhovém a rutinním provozu nedochází k závažným chybám a obecně se počet chyb nápadně menší.
. Snadné změny. Modul rozsahu stránky se mění velmi snadno, logika modulu je triviální. Také zde velmi pomáhají popisy modulu.
. S předchozím bodem souvisí i snadná přenosnost modulů mezi programátory.
. Snížení času kompilací při ladění a změnách programu. Oprava jediného modulu spotřebuje čas na překlad 2 - 3 min, zatímco monoliticky napsaný program 20 - 30 min (i více).
. Dokumentace programu. Popsané programy a jejich moduly jsou po stránce programátorské samodokumentující.
. Možnost použití modulů obecného určení z modulů z jiných programů bez jejich změny.
. Program je možno kdykoliv overlayovat podle potřeby.

Nevýhody.

. Větší pracnost ze strany odpovědného programátora. Odpovědný programátor musí být na mnohem vyšší úrovni

aby dokázal správně vystihnout všechny vazby, detailně je popsat a odhalit chyby při testování celého programu. Takovýchto programátorů nebývá ve středisku nadbytek.

- Pokud programátor modulu neotestuje modul dostatečně kvalitně nebo si změní podmínky zadání, může pro něj "odladěný" modul být zdrojem chyb v programu. Modul se pak musí narychlo upravovat, přepisovat a pod., což jeho kvalitě zrovna napomáhá.
- Tento systém vyžaduje spolupráci celého týmu. Nezodpovědným přístupem jednoho či více lidí může být zmařena práce celého týmu.
- V případě hledání chyb (je-li např. přepsána část paměti či podobné lahůdky) je situace odpovědného programátora nezávidění hodná. Podle našich zkušeností je většina takovýchto chyb způsobena nesouladem při předávání argument - parametr.

Celkový dojem:

Modulární programování je nespornou výhodou, je efektivní, i když náročnější na logické promyšlení problému.

ODBOR RIZENI A VYPOCETNI TECHNIKY POLDI-SONP KLADNO **PŘÍLOHA 1**

POZADAVKOVY LIST NA - NEPLANOVANE PRACE
- ZMENU NEBO DOPLNEK PROJEKTU
- ZPRACOVANI PODLE HOTOVYCH PROGRAMU

ZADAVATEL	ODBOR OBLAST	PODNIKOVA ADRESA

NAZEV UKOLU

STRUCNA CHARAKTERISTIKA POZADAVKU

(POZN.: PRIPADNY PODROBNY POPIS UVEDETE V PRILOZE.)

DULEZITOST :	*EK. PRINOS	KCS	PRAC. SIL
PRIKAZ GR C.			
PRIKAZ P C.	*ABSOLUTNI		
OSTATNI			
	*RELATIVNI		

POVERENY PRAC. ZADAVATELE	TIL.	PODP. ZAMEN.

POCET PRILOH	DATUM ZADANI	PODPIS VED.ODB.(ODD.)

DOSLO ONF CISLO

VYJADRENI ODBORU GM

POVERENY PRAC. GM	SUBSYSTEM

DATUM	SOUHLAS VED. ODBORU GM

PRIDELOVANE VYSLEDKY PREVZAL - JMENO:

ODDELENI: DATUM: PODPIS:

PŘÍLOHA 2

ZADAVACI FORMULAR PROGRAMU	AGENDA:	POZAD.
		LIST:

DRUH DS	JMENO PROGRAMU	ZMENA	INTERNI ZAKAZKA	PRIV	ZADAVATEL (ANALYTIK)	ZADANO R/M M/D D
#EB	|_|_|_|_|_|_|_|	|_|	|_|_|_|_|_|	|_|	|_|_|	|_|_|_|_|_|_|
1-3	4 - 11	12	13 - 17	18	19 - 20	21 - 26

NAZEV PROGRAMU

|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|

27 - 59

NAZEV PROGRAMU (POKRACOVANI)	PROGRAMATOR	UKONCENO R R/M M/D D	POR. CISLO
|_|_|_|_|_|_|_|_|_|	|_|_|	|_|_|_|_|_|_|	|_|_|_|_|
60 - 68	69 - 70	71 - 76	77 - 80

POZADOVANY TERMIN:	UCET:	SOUCET KNIHOVNA:

UZIVATEL PROGRAMU:

ODPOVEDNY VEDOUCI UKOLU:

ZADANI UKOLU - STRUCNY POPIS CINNOSTI:

ODHAD PRACNOSTI:	PLAN. TERMIN UKONCENI:

ZADAL:	PREVZAL:	VED. ANALYTIK:	VED. PROGRAMATOR:

PŘÍLOHA 3

```
 DRUH     J M É N O      ZMĚNA     INTERNÍ     PRTY ZADAVATEL Z A D Á N Í
  OS      M O D U L U               ZAKÁZKY        (PROGR.) A Č/P H/C L

 REC
 1-3        4 - 11        12      13 - 17     18   19 - 20     21 - 26
=========================================================================
                     N Á Z E V   M O D U L U

                            27 - 59
=========================================================================
      J M É N O                                  U K L Á D Á N Í
   P R O G R A M U                PROGRAMÁT.P     N Č/P H/D L

        61 - 68                    69   70          71 - 76
-------------------------------------------------------------------------
 POŽADOVANÝ TERMÍN:               , UKOL:         PŘEDPOKL. PRIORITA:
-------------------------------------------------------------------------
 ODHAD PRACNOSTI:
=========================================================================
 ZADAL:              VED.PROGRAMÁTOR:      , PŘEVZAL:

-------------------------------------------------------------------------
```

PŘÍLOHA 4

```
         ZG    121  VYPIS CLENU (BAREI    ) Z KLIENTE  NG  PACKY

                 VLOZEN:07.07.77       VLOZIL:

*/*       POSLEDNI ZMENA CLENU BAREI     DNE 77.179 JOHN PUPAT          */
 BAREI:PROC(PUCET ,UST,UCET,SKLAD) RETURNS(FIXED BIN(15));
-/*            URCENI INDEXU PRO SOUCTOVANI PODLE UCTU A SKLADU

OPROGRAMATOR: ADELSBERGER                                   DATUM: 14.11.197
-NADRAZENA PROCEDURA: BAREK
 PODRAZENA PROCEDURA: -
OFUNKCE: PROCEDURA UCHOVAVA NEJVYSSI POUZITOU HODNOTU INDEXU PRO POLE
         STRUKTURY REKGET  PRI VOLANI ZVYSI TUTO HODNOTU O 1, PREDA
         VYSLEDEK VOLAJICI PROCEDURE A ZAROVEN ZAZNAMENA TENTO INDEX
         DO PEVNU POLE STRUKTUR UST.ST SPOLU S ODPOVIDAJICIM SKLADEM
         (SKLAD DO UST.SKL)
         PUCET(UCET) - UDAVA INDEX POLE STRUKTURY UST,T.J. UDAVA, KTERY
                       PRVEK UST ODPOVIDA POZADOVANEMU UCTU
         UST.UI(PUCET(UCET)) UDAVA POCET OBSAZENYCH PRVKU ST A SKL,
                             T.J. POCET SKLADU S TIMTO UCTEM
         UST.SKL(PUCET(UCET),*) JSOU SKLADY S TIMTO UCTEM
         UST.ST(PUCET(UCET),*) JSOU INDEXY TECHTO SKLADU V POLI
OVSTUPNI PARAMETRY: PUCET - POLE INDEXU V ZAVISLOSTI NA UCTU
                    UST - POLE STRUKTUR ST SKL  A INDEXU SKLADU
                    UCET - UCET,PRO KTERY HLEDAME INDEX
                    SKLAD-SKLAD,PRO KTERY HLEDAME INDEX
 VYSTUPNI PARAMETRY: FUNKCNI HODNOTA   INDEX POLE REKGET
OPOUZITE SOUBORY: -
OPOUZITE DEKLARACE: -
```

```
-/*****************************************************************/
 /*              DEKLARACE                                        */
 /*****************************************************************/
ODCL PUCET(*) FIXED BIN(15);
 DCL 1 UST(*),
       2 UI FIXED BIN(15),     /* POCET SKLADU,KTERE SE DRI MOST VYSK  */
       2 SI(*) FIXED BIN(15), /* INDEXY SKLADU V POLI DESKET           */
       2 SKL(*) FIXED(5);     /* SKLADY                                */
 DCL UCET FIXED(5);
 DCL SKLAD FIXED(5);
 DCL J FIXED BIN(15) STATIC;  /* PLATNY INDEX POLE STRUKTUR V KUCI     */
 DCL UJ FIXED BIN(15) STATIC; /* NEJVYSSI POUZITY INDEX POLE UST       */
 DCL I FIXED BIN(15) STATIC;
/*****************************************************************/
 /*              VLASTNI PROGRAM                                  */
 /*****************************************************************/
OIF SKLAD = 0                 /* PRI VOLANI S NULOVYM CISLEM SKLADU   */
 THEN DO;                     /*   PLATNY INDEX POLE [illegible]      */
        J = 0; PUCET = 0; UJ = 0;  /* MISTA V STRUKTURE JSOU VOLNA     */
        RETURN(J);
      END;
 J = J + 1;
 I = PUCET(UCET);             /* INDEX STRUKTUR UST                    */
 IF I = 0
 THEN DO;                     /* UCET SE VYSKYTL [illegible]           */
        UJ = UJ + 1;
        IF UJ > HBOUND(UST,1)
        THEN CALL BAREL('UMAX','BAREK');
        I,PUCET(UCET) = UJ;   /* [illegible]                           */
        UI(UJ) = 0;           /* [illegible]                           */
      END;

UI(I) = UI(I) + 1;            /* DALSI SKLAD [illegible]               */
 IF UI(I) > HBOUND(UST,2)
 THEN CALL BAREL('SMAX','BAREK');
 SI(I,UI(I)) = J ;            /* ULOZENI DO POLI                       */
 SKL(I,UI(I)) = SKLA ;        /* ULOZENI SKLADU                        */
 RETURN(J);
END BAREL;
```