M O D U L Á R N Í P R O G R A M O V A C Í S Y S T É M S N A P

RNDr. Ing. Jiří N é m e c, Ing. Jan S o u č e k,
Ing. Jiří T y w o n i a k
VÚMS, k.ú.o., Loretánské nám. 3, 118 55 Praha 1

Systém SNAP byl vytvářen ve VÚMS Praha jako prostředek pro víceúrovňovou simulaci hardware a software počítače. V rámci této práce byl kladen velký důraz i na metodologii návrhu velkých softwarových a hardwarových celků. Jedním z výsledků byla filosofie programovacího systému SNAP, spočívající v rozdělení programování na dvě odlišné intelektuální činnosti. První lze nazvat programování ve velkém a tato činnost zahrnuje návrh rozdělení vytvářeného programu na moduly a návrh mezimodulárních vazeb. Druhou činností je tzv. programování v malém, čímž rozumíme implementaci jednotlivých modulů. Tomuto rozdělení odpovídá i rozdělení programovacích jazyků v systému SNAP - tj. na jazyk programování ve velkém a jazyky programování v malém.

Jazyk programování ve velkém

Jazyk programování ve velkém v systému SNAP umožňuje popsat modulární strukturu vytvářeného programu. Modulární strukturou programu zde rozumíme způsob, jakým je program rozdělen do samostatně překládaných modulů a rozhraní (interface) každého modulu. V rozhraní modulu jsou uvedeny ty objekty (procedury, data, datové typy), které modul exportuje (tzn. zpřístupňuje ostatním modulům), a moduly, jejichž exportované objekty daný modul importuje (tzn. používá). Používat proceduru znamená tuto proceduru volat, používat jisté datum znamená z tohoto data číst nebo do něho zapisovat, používat datový typ znamená deklarovat data to-

hoto datového typu atd. Současné rozhraní modulu zahrnuje popis atributů resp. vlastností každého exportovaného objektu, které jsou podstatné z hlediska jeho použití v importujících modulech. Takovou vlastností například pro proceduru je počet a datové typy parametrů, pro datum jeho datový typ atd.

Jazyk programování ve velkém může být (a obvykle bývá) přirozený jazyk. Pak je způsob popisu modulární struktury programu pouze věcí domluvy uvnitř realizační skupiny programátorů. Tato varianta se zdánlivě jednoduše zavádí do praxe, dodržování zásad popisu však záleží jen na pečlivosti a dobré vůli programátorů. Je nejvýš pravděpodobné, že v průběhu realizace programu nebude neformální popis modulární struktury odrážet její skutečný stav, což degraduje celou metodu.

Na druhé straně nelze zcela formálně vyjádřit všechny ty vlastnosti, které ten či onen objekt z hlediska modulární struktury programu má. Formálním vyjádřením vlastností objektů rozumíme nejen formální popis, ale i algoritmickou kontrolu, zda daný objekt je v souladu s touto vlastností používán.

Výhodným se jeví kompromis, který je přijat v systému SNAP. Formálně se popisují ty části rozhraní modulu, které lze snadno využít pro testování, zda jsou v následné implementaci modulů skutečně splněny. Doplnění popisu ostatních vlastností objektů lze pak provést v přirozeném jazyce - formou komentářů jazyka programování ve velkém.

Ukažme si způsob popisu modulární struktury programu v jazyce programování ve velkém v systému SNAP. Mějme realizovat program PRENOS, který provádí komunikaci s nějakou protistanicí po telekomunikační lince. Data se přenáší

z linky do nějakého souboru nebo naopak. Před každým přenosem dat do / ze souboru probíhá konverzace s operátorem, která určí podmínky přenosu (přenosový kód, jméno souboru atd.). Modulární strukturu programu popíšeme prostřednictvím tzv. obsahového modulu. Ten obsahuje výčet modulů (resp. podsystémů modulů), z nichž se program PRENOS skládá a ke každému takovému modulu je uveden popis jeho rozhraní:

```
MODULE  PRENOS
OWNER:  NEMEC
EXPORT  PROC  RUN_PRENOS
CONTENTS
   MODULE  KONVERSACE
   * komunikace s operátorem
   EXPORT  PROC  RUN_PRENOS
   IMPORT  PROC  OF  LINKA , UKLADANI
   MODULE  LINKA
   * komunikace na lince
   EXPORT  PROC  READ_LINKA ( BLOK CHAR(255) VARYING ),
                 WRITE_LINKA (BLOK CHAR(255) VARYING )
   MODULE  UKLADANI
   * ukládání do souboru
   EXPORT  PROC  OPEN_SOUBOR ( JMENO CHAR (8) ),
                 CLOSE_SOUBOR
   EXPORT  PROC  READ_BLOK ( BLOK CHAR (255),
                             DELKA BINARY ),
                 WRITE_BLOK ( BLOK CHAR (255),
                              DELKA BINARY )
   END  OF  MODULE  PRENOS
```

Obsahový modul hraje tedy v programu podobnou roli jako obsah v knize. Programátor je při psaní obsahových modulů veden k tomu, aby si modulární strukturu vytvářených programů nejdříve rozmyslel a přesně popsal, tj. "naprogramoval

ve velkém". Vytvořený obsahový modul je předložen SNAPu k překladu, ten provede kontrolu, zda jsou splněna kritéria správného návrhu (např. navržená modulární struktura je hierarchická), a uloží informace získané z obsahového modulu do svého informačního souboru.

Programátor nyní rozhodne o realizaci jednotlivých modulů resp. podsystémů modulů, které v obsahovém modulu popsal.

Má k dispozici tyto možnosti:

a/ funkce vybraného modulu vyžadují rozsáhlé programování, je vhodné prohlásit vybraný modul za podsystém modulů a modulární strukturu popsat v dalším obsahovém modulu,

b/ funkce vybraného modulu nevyžadují rozsáhlé programování a tudíž je modul možné realizovat v jediném tzv. programovém modulu.

Stejný postup lze aplikovat při realizaci obsahového modulu na kterékoliv úrovni. Rozklad cílového systému na moduly vytváří tak stromovou strukturu; kořenem této stromové struktury je tzv. hlavní obsahový modul (v našem příkladě PRENOS), další obsahové moduly tvoří vnitřní uzly a programové moduly tvoří listy této struktury. Možný rozklad programu PRENOS z našeho příkladu znázorňuje obr. 1.

Pro vyjádření vztahů mezi moduly používáme obvyklé terminologie otec-bratr-syn. V příkladě na obr. 1 je modul PRENOS otcem modulů KONVERSACE, UKLADANI a LINKA, moduly KONVERSACE, UKLADANI a LINKA jsou bratři, modul LINKA je synem modulu PRENOS. Výrok "modul UKLADANI je otcem modulů POS_SOUBORY, SEQ_SOUBORY a LIOCS" tedy znamená, že modul UKLADANI je obsahový a popisuje ve svém těle rozhraní modulů POS_SOUBORY, SEQ_SOUBORY a LIOCS.

Mezi moduly cílového systému existuje ještě vztah export / import. Tento vztah je ve SNAPu podřízen následujícím pravidlům a omezením.

Exportuje-li obsahový modul jistý objekt, pak tento objekt musí být exportován současně z některého synovského modulu daného obsahového modulu. Exportuje-li programový modul jistý objekt, pak tento objekt musí být deklarován v těle daného programového modulu. Modul může importovat jen ty objekty, které exportuje nějaký jeho bratrský modul nebo které importuje jeho otcovský modul.

Na základě vztahu export / import je definována mezi bratrskými moduly relace nadřazenosti. Říkáme, že modul A je nadřazen modulu B, jestliže mezi nimi existuje alespoň jeden z následujících export / import vztahů:

A importuje proceduru exportovanou B
A exportuje data importovaná B
A importuje datový typ exportovaný B
A importuje výjimku exportovanou B

Pro relaci nadřazenosti musí platit v systému SNAP následující pravidlo: Graf relace nadřazenosti na množině synovských modulů libovolného obsahového modulu je bez cyklů.

Tímto způsobem je celý systém rozložen na postupně zjemňovanou hierarchii virtuálních strojů. Na obr. 2, který představuje možnou hierarchii virtuálních strojů, pro příklad rozkladu cílového systému z obr. 1, je rozhraní virtuálních strojů vyznačeno čárkovaně a relace nadřazenosti dvojitými šipkami.

Jazyky programování v malém

Programové moduly jsou vytvářeny v jazycích programování v malém. V současné době do systému SMAP jsou zařazeny programovací jazyky z těchto skupin:

a/ klasické / univerzální programovací jazyky
- PL/I
- FORTRAN

připravuje se COBOL
- assembler

b/ specializované jazyky pro určité oblasti použití
- jazyk meziregistrových přenosů NEJDA

c/ předprocesory programovacích jazyků
- předprocesor umožňující používání datových abstrakcí v jazyce PL/I
- předprocesor pro strukturované programování /využívající Jacksonovy metodologie programování/ v jazycích PL/I, FORTRAN, COBOL, assembler

Každý programový modul obsahuje popis svého rozhraní, které se však obvykle prostřednictvím příkazu COPY získává z dříve přeloženého otcovského obsahového modulu. Následuje určení programovacího jazyka modulu a vlastní tělo modulu obsahující deklarace těch objektů (procedur, dat, datových typů maker), které jsou v rozhraní modulu prohlášeny za exportované nebo které jsou v daném modulu lokální. Například programový modul RIZENI vypadá takto:

```
MODULE KONVERSACE
OWNER: NEMEC
DATE: 311282
COPY ALL FROM CONTENTS
IMPLEMENTATION: PLI
  *PROC (RUN_PRENOS);
        <text v jazyce PL/I>
END OF MODULE RIZENI
```

Systém SNAP má při překladu každého programového modulu seznam všech objektů, které musí modul exportovat a seznam objektů, které model může importovat. Ke každého objektu ze zmíněných seznamů má k dispozici i jeho atributy, např. datové typy parametrů procedur, datové typy dat atd. Je tedy možné zkontrolovat, zdali překládaný programový modul odpovídá modulární struktuře programu popsané v obsahových modulech. Tato kontrola se v současné době provádí tak, že do těla programového modulu se vkládají určité příkazy příslušného jazyka programování v malém. Ilustrujme tento přístup na příkladě jazyka PL/I:

1/ pro každé exportované nebo importované datum se vkládá do těla modulu deklarace:

DCL <jméno data> <datový typ> EXTERNAL;

2/ pro každou proceduru, kterou je možné importovat do modulu, se vkládá do těla modulu deklarace:

DCL <jméno procedury> ENTRY (<seznam datových typů>)
RETURNS (<datový typ>) ;

3/ pro každou exportovanou proceduru se na základě příkazů

#PROC (<jméno procedury> (<seznam jmen parametrů>)) ;

uvedeného v těle modulu generuje záhlaví procedury odpovídající jazyku PL/I:

<jméno procedury> : PROC (<seznam jmen parametrů>) ;
DCL <jméno parametru 1> <datový typ> ;
.
.
DCL <jméno parametru n> <datový typ> ;

4/ pro každou exportovanou výjimku se vkládá do těla modulu deklarace:

DCL <jméno výjimky> EXTERNAL CONDITION;

V případě, že se importuje objekt deklarovaný v jiném jazyce programování v malém, provádí se automatický převod zápisu datových typů z jednoho jazyka programování v malém do PL/I.

Modifikovaný text těla programového modulu se předloží kompilátoru PL/I k překladu. Analogickým způsobem jako s PL/I se zachází v systému SNAP i s ostatními jazyky programování v rámci s tím, že například v assembleru je rozsah kontrol podstatně menší.

Pomocné funkce systému SNAP

Systém SNAP poskytuje celou řadu pomocných funkcí, které usnadňují vytváření modulárních programů.

Tzv. knihovní podsystém SNAPu vytváří takovou nadstavbu nad knihovními programy hostitelského operačního systému, která umožňuje udržování několika verzí jednoho modulu na úrovni zdrojového i přeloženého tvaru. Navíc tento podsystém udržuje záznamy o vztazích otec-bratr-syn mezi moduly, které se využívají pro automatizaci spojování vytvářených programů tzv. spojovacím podsystémem SNAPu. Tento opět pracuje jako nadstavba nad spojovacím programem hostitelského operačního systému. Spojování se provádí na základě určení hlavního obsahového modulu pro vytvářený program. Spojovací podsystém SNAPu pak do výsledného programu zahrne jen ty verze programových modulů, které odpovídají jediné verzi popisu modulární struktury programu prostřednictvím obsahových modulů.

V případě změny v popisu modulární struktury programu systém SNAP automaticky sleduje její šíření do jednotlivých modulů programu. Uživateli pak oznámí ty moduly, jichž se provedené změny týkají.

Systém SNAP především podporuje strategii tvorby programu shora-dolů, program je však možné prostřednictvím aparátu tzv. standardních modulů vytvářet i zdola nahoru. Stejně tak je možné snadno začleňovat do nově vytvářeného

programu pod systém SNAP i moduly ze starších programů, které pod systémem SNAP vytvářeny nebyly.

Na základě informací uvedených v obsahových modulech se vytváří přehledná dokumentace modulární struktury programů.

Závěr

Systém SNAP je zatím implementován a používá se v operačním systému OS/VS1, provádí se jeho přenos do operačního systému DOS-3. Systém SNAP je koncipován jako obecný nástroj, jehož výhody se projeví zvláště tehdy, je-li použit pro tvorbu rozsáhlých programů se značnou mírou vnitřních závislostí. Nutí tvůrce programů k takovému postupu při návrhu, který ústí ve výsledný program s přehlednou, dobře dokumentovanou modulární strukturou, která je jen v malé míře ovlivněna zvyklostmi jednotlivých tvůrců. Další přínos systému SNAP spočívá v automatickém provádění kontrol mezimodulárních vazeb v programu již v době překladu jednotlivých modulů (tzv. separátní překlad modulů).

Literatura:

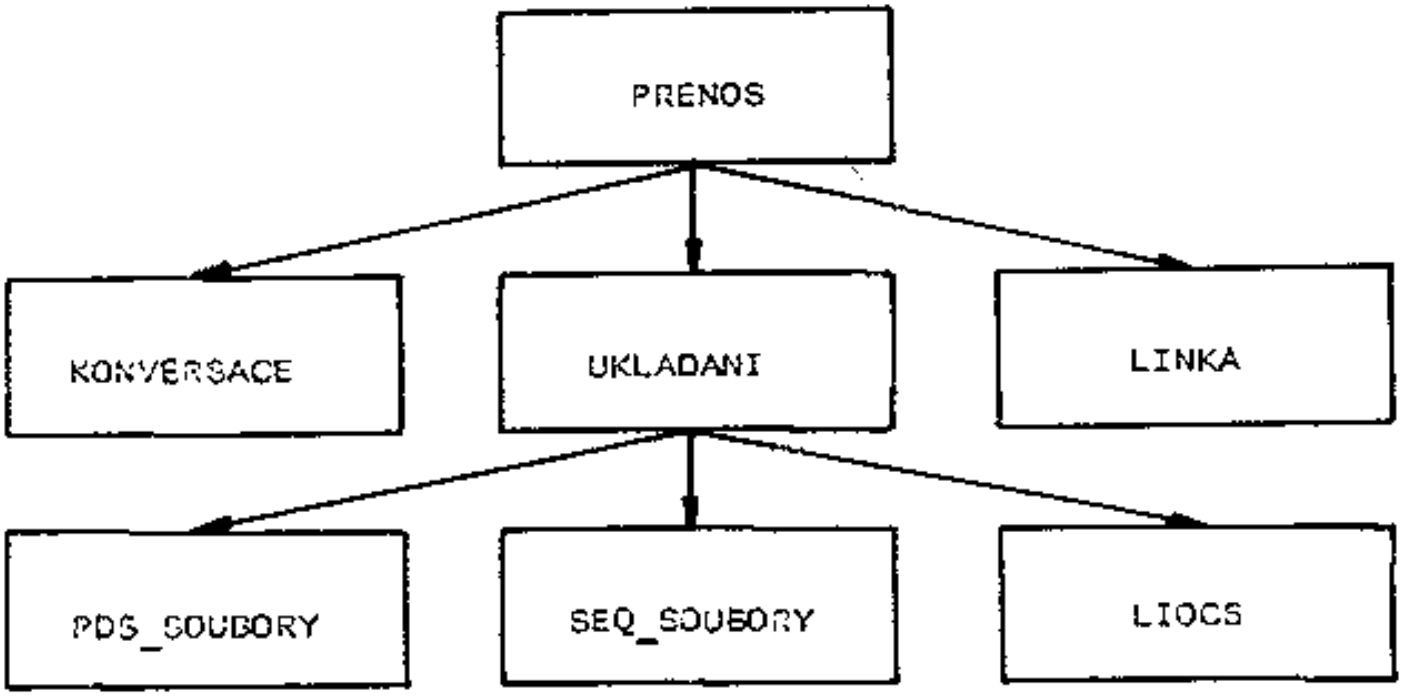
[1] Němec, J., Součck, J., Tywoniak, J.: Programovací systém SNAP (uživatelská dokumentace), VÚMS, 1982

[2] Němec, J., Součck, J., Tywoniak, J.: SNAP a metodologie programování v automatizaci, sborník CAD 81, Dům techniky Praha, 1981

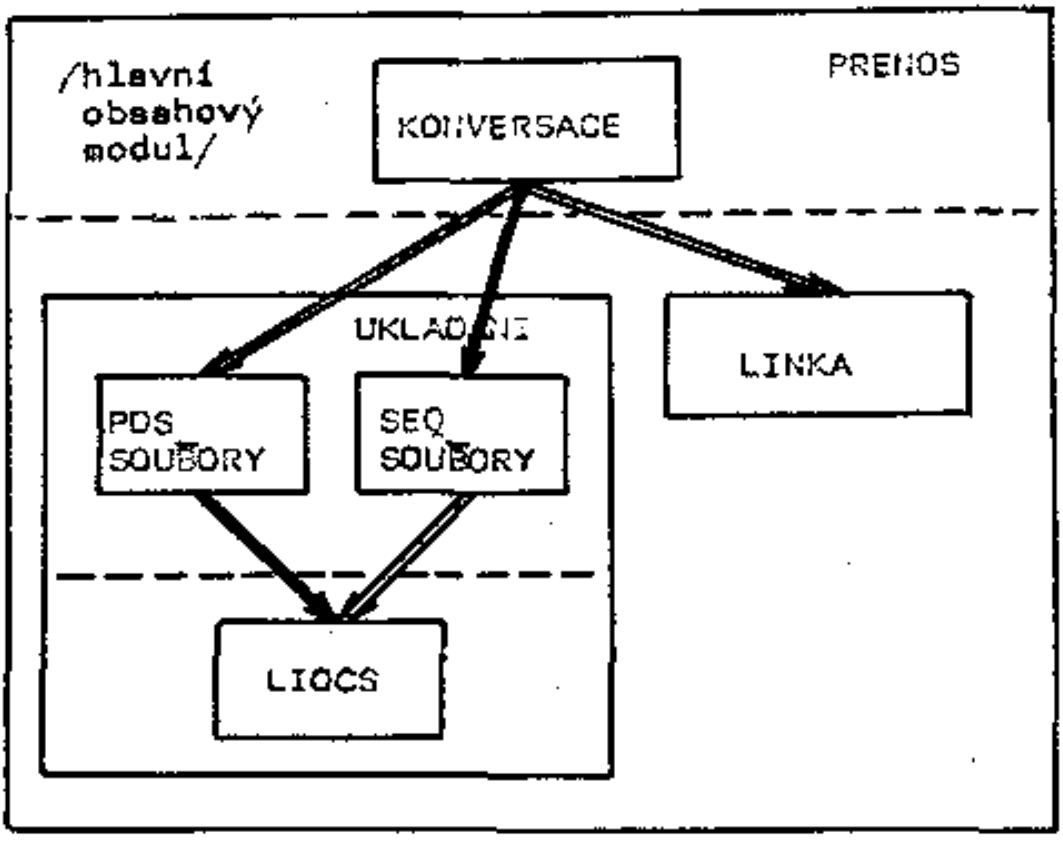
[3] Němec, J., Součck, J., Tywoniak, J.: Programování v systému SNAP, sborník Moderní metody ladění programu, Dům techniky České Budějovice, 1982

[4] Tywoniak, J., Součck, J., Němec, J.: Separátní překlad modulů v jazyce PL/I, sborník SOFSEM 81, VVS Bratislava, 1981

[5] De Remer, F., Kron, H.: Programming-in-the Large versus Programming-in-the Small, International Conference on Reliable Software, Los Angeles, 1975


PRENOS
KONVERSACE
UKLADANI
LINKA
PDS_SOUBORY
SEQ_SOUBORY
LIOCS

Obr. 1


/hlavní
obsahový
modul/
PRENOS
KONVERSACE
UKLADANI
LINKA
PDS
SOUBORY
SEQ
SOUBORY
LIOCS

Obr. 2