Jiří Prokop
VVS PP Čokoládovny,o.p., Praha

VYUŽITÍ POČÍTAČE PRO TVORBU DOKUMENTACE

1. Úvod

Se vzrůstající složitostí vytvářených programových systémů roste význam podrobné, přesné a stále v aktuálním stavu udržované dokumentace. Nepříjemné přitom je, že rostoucí objem dokumentace znamená

- velkou spotřebu kapacity kvalifikovaných pracovních sil pro její vytváření
- znesnadnění údržby dokumentace v aktuálním stavu
- obtížnější orientaci v dokumentaci, nesnadné hledání toho, co právě potřebujeme vědět.

Jak lze těmto potížím čelit?

- pečlivějším výběrem informací, které jsou v dokumentaci obsaženy - méně bývá často více. Už při programování můžeme objem budoucí dokumentace zredukovat - na př. dbáme, aby zprávy programů byly samovysvětlující.
- vhodným členěním dokumentace podle profesí pracovníků, jimž je určena
- co možná nejširším využitím počítače při vytváření dokumentace.

Tento příspěvek je věnován možnostem využití počítače,a to zejména při vytváření programové dokumentace na úrovni provádějícího projektu. Některé postupy mají však obecnější platnost a použití.

2. Popis programového systému

Popsat systém (jakýkoli) znamená

a) popsat jednotlivé prvky systému

b) popsat vazby mezi jednotlivými prvky

Představe struktury systému je totožná s pojmem orientovaného grafu, jehož uzly odpovídají prvkům systému a hrany vazbám mezi nimi. Matematickým prostředkem pro popis struktury systému je teorie grafů.

V případě programového systému jsou prvky systému (na nejnižší rozlišovací úrovni) moduly (hlavní programy, externí podprogramy). Na vyšší rozlišovací úrovni jsou prvky systému jednak moduly (hlavní, proveditelné programy), jednak datové množiny (soubory, sestavy).

Na ještě vyšší rozlišovací úrovni je systém tvořen joby a datovými množinami trvalého charakteru.

Vazby mezi moduly jsou dvojího typu:

a) vertikální vazby - tímto pojmem označme vazby vzniklé použitím příkazu CALL, % INCLUDE v jazyku PL/1 a podobně
b) horizontální vazby - tak označíme vazby mezi hlavními programy a množinami dat, vzniklé tím, že vstupními soubory určitého programu jsou výstupní soubory jiných programů, atd.

Podívejme se nyní podrobněji na jednotlivé dílčí úkoly, které musíme při popisu programového systému vyřešit.

3. Popis modulů

Protože modul je základním stavebním kamenem programového systému, je nutno jeho popisu věnovat zvlášť velkou pozornost. Nejlepším způsobem (už z literatury známým) je dokumentace modulu jeho zdrojovým textem.

Jako vzor mohou posloužit na př. moduly SSP firmy IBM.

Výhody jsou zřejmé:

- při práci se zdrojovým textem není nutno hledat potřebné informace jinde
- dokumentace je udržována stále v aktuálním stavu, protože se mění současně se změnami zdrojového textu
- uložení na médiu počítače umožňuje snadný přístup nejen člověku, ale i počítači.

Každý modul je opatřen úvodním poznámkovým rámečkem, obsahujícím všechny informace potřebné pro jeho použití. Není vhodné umisťovat zde informace o vazbách s okolím modulu, protože by mohla vzniknout potřeba tyto informace aktualizovat (kvůli změnám okolí) bez nutnosti změny zdrojového textu.

Vypovídací schopnost opisu zdrojového textu je závislá na programovacím jazyku, metodice programování a formální úpravě. O tom ale bylo už dosti řečeno a napsáno jinde, proto se spokojím odkazem např. na /2/.

4. Vertikální vazby

Graf, znázorňující vazby mezi moduly (vazby vzniklé použitím příkazu CALL, % INCLUDE v PL/1, použitím uživatelem sestaveného makra) je orientovaný a acyklický (pominemе-li případnou rekursivitu procedur - což pro naše účely můžeme). Graf je definován seznamem hran; pro každou hranu si budeme vedle incidentních uzlů pamatovat také druh vazby, která je touto hranou reprezentována. Sestavíme-li graf vazeb pro všechny programy a uložíme-li odpovídající seznam hran jako sekvenční soubor, získáme dokumentaci těchto vazeb, přístupnou nejen člověku, ale i počítači.

Tím vzniká možnost řešit na počítači některé typy úloh, na př.

a) zjistit, ve kterých modulech je použit specifikovaný modul (moduly), což lze matematicky formulovat jako úlohu precedenční analysy: k dané množině uzlů najít všechny předchozí uzly.
b) pro specifikované moduly zjistit seznam všech podřízených modulů - úloha sekvenční analysy: k dané množině uzlů najít všechny následné uzly
c) pro specifikované moduly (představující na př. hlavní programy určitého systému) zjistit všechny vertikální vazby v tomto subsystému - úloha najít k dané množině uzlů subgraf, tvořený těmito uzly, následnými uzly a incidentními hranami.

Potřebné algoritmy: ulohy typu b) a c) převládají; proto je

vhodné, aby seznam hran byl uložen uspořádaně, a to tak, že pro každý uzel U grafu platí: každá hrana, vedoucí do uzlu U, se v seznamu hran nachází dříve než kterákoli z hran, které z uzlu U vycházejí. Tuto úlohu řeší Fordův algoritmus, který je základním algoritmem síťové analysy a je popsán na př. ve /3/.

Další potřebný algoritmus řeší úlohu vyhledání všech elementárních cyklů v orientovaném grafu, kdyby v důsledku chyb graf cykly obsahoval. Vhodný algoritmus lze získat modifikací algoritmu popsaného ve /4/.

Ostatní potřebné algoritmy jsou v případě acyklického grafu velmi jednoduché; pro případ obecného orientovaného grafu jsou popsány v /1/.

Pro dokumentaci vertikálních vazeb mezi moduly byl v našem výpočtovém středisku vytvořen potřebný software, používaný téměř rok. Už při sestavování programu vydělruje programátor štítky s popisem vazeb ve tvaru A B V , kde

A je jméno nadřazeného modulu

B je jméno podřízeného modulu

V je druh vazby (1 znak); zde značí:

L v modulu A je volán modul B

I modul A obsahuje příkaz pro inklusi modulu B

M v modulu A je použito uživatelem sestavené makro B

E jméno B je vstup (entry) modulu A

O neznamená vazbu, ale příkaz zrušit v souboru vazeb vazbu od A k B.

Tyto štítky se shromažďují na jednom místě a čas od času slouží jako vstup programu SDOKUMS1, který provede kontrolu vstupních dat, a v případě jejich správnosti aktualizaci a uspořádání souboru vazeb. Soubor vazeb je uložen jako člen trvalé a stále přístupné knihovny. Slouží v případě potřeby jako vstup programu SDOKUMS2 (dalším jeho vstupem je seznam modulů), který je řízen parametricky. Na základě zadaného parametru lze vybrat některé z následujících možných funkcí programu:

a) rozšířit vstupní seznam modulů o všechny podřízené moduly,

případně jen o podřízené moduly s určitým druhem vazby
b) vytvořit pro další použití výstupní soubor (soubory), obsahující takto vytvořený výsledný seznam.
c) vypsat zdrojové texty (nebo jen poznámkové ráněčky) modulů, obsažených ve výsledném seznamu
d) uložit zdrojové texty modulů, specifikovaných ve výsledném seznamu, na magnetickou pásku, a to buď ve formě souboru SYSIN pro IEBUPDTE nebo ve formě klidové magnetické pásky knihovního systému pro zdrojové moduly (viz /5/).
e) vytvořit pro další použití subgraf vazeb pro dané vstupní moduly a moduly k nim podřízené.

Pro funkce v bodech c) a d) je nutný ještě další vstup: klidová (příp. i další úrovně) magnetická páska knihovního systému pro zdrojové moduly.
Tento knihovní systém byl převzat z výpočetního střediska NHKG a je popsán v /5/.

Subgraf, vytvořený programem SDOKUMS2 (nebo případně i celý graf vazeb) lze vytisknout v přehledné grafické formě programem SPINNE. Program SDISTRM1 umožňuje uložit ze vstupní knihovny členy specifikované ve vstupním seznamu na výstupní soubor ve formě souboru SYSIN pro IEBUPDTE.

Z uvedeného popisu je zřejmé, že použití programů je výhodné zejména tehdy, když má být do jiného výpočetního střediska předán určitý programový systém. Jediným příkazem lze vytvořit distribuční pásku, která obsahuje všechno potřebné a zároveň vytisknout informace potřebné pro její další zpracování. Programy mohou být předány ve zdrojové formě i ve formě zaváděcích modulů. Ve druhém případě navazuje na výsledné seznamy program knihovního systému pro zaváděcí moduly, který byl v našem středisku vytvořen s cílem úspory místa na discích. Jeho hlavní funkcí je umožnit ukládání zaváděcích modulů na magnetickou pásku a snadný převod do dočasné knihovny pro jejich provedení.

Protože řada programů pracuje se vstupním souborem, obsahujícím seznam jmen, je ve všech těchto programech použit jednotný způsob zpracování (týká se to nejen zde

popsaných programů, ale také programů obou zmíněných knihovních systémů, takže seznamy lze ukládat jako knihovní členy. Mohou také obsahovat komentář. Některé programy umožňují specifikovat skupinu modulů. Specifikací je skupina znaků ukončená tečkou a znamená všechny moduly, jejichž jména začínají touto skupinou znaků.

5. Horizontální vazby

Tyto vazby lze rovněž znázornit orientovaným grafem, jehož každý uzel představuje buď program nebo datovou množinu. V případě určitých zjednodušení je i tento graf acyklický. Zde je ovšem značně pracnější pořízení dat pro sestavení grafu. Proto zatím nemáme ani vytvořen software pro tento účel.

Nabízí se sice možnost využít vlastností jazyka JCL v operačním systému OS a snažit se sestavit graf automaticky z informací obsažených v jobech a procedurách JCL, ale ani v tomto případě nelze vystačit jen s těmito informacemi.
Hlavním problémem je otázka, do jaké míry výsledný efekt vyváží vloženou práci.

Některé úlohy, které by bylo možno na základě horizontálních vazeb řešit pomocí počítače, můžeme řešit i existujícími prostředky: na př. úlohu, které programy je nutno přepracovat, resp. rekompilovat při změně větné struktury určitého souboru, můžeme řešit za předpokladu, že deklarace struktury byla uložena jako samostatný modul a do programů byla inkludována, takže jde vlastně o vertikální vazbu.

6. Textová část dokumentace

Od dokumentace se nelze odmyslit části, které nutně mohou mít jen charakter slovního popisu. I zde nám však může být počítač užitečným pomocníkem. Výzkumný ústav matematických strojů na př. dodává uživatelům počítače EC 1021 programový ediční systém, kterým lze provádět ediční úpravy textu.

U nás k tomu účelu používáme program knihovního systému pro zdrojové moduly. Neprovádí sice ediční úpravy, ale umožňuje snadnou aktualizaci a vkopírování částí jiných textů. Výhody jsou zřejmé:

- snadné udržování v aktuálním stavu
- snadný přístup k informacím
- menší pravděpodobnost výskytu chyb

7. Závěr

Některé výhody i nevýhody tohoto přístupu k dokumentaci jsou zřetelné už z předchozího textu. Zbývá snad ještě dodat, že veškerý obecný uživatelský software (tedy i ten, který byl vytvořen pro účely tvorby dokumentace) představuje vždy také podporu snahy o jednotnost, automaticky hlídá dodržování standardů a konvencí zavedených ve středisku. Je-li " čtenářem " části dokumentace nejen člověk, ale i počítač, zvyšuje se pravděpodobnost včasného odhalení případných chyb, protože počítač bývá obvykle přísnějším kritikem.

Po dosavadních zkušenostech můžeme konstatovat, že pro programové systémy, představující rozšíření základního programového vybavení (uživatelský software) je možné pro celý rozsah dokumentace využít počítače. U takových systémů jsou totiž v důsledku větší hloubky modularity významnější vertikální vazby.
Opačně je tomu u subsystému ASŘ, kde lze popsané metody využít jen zčásti, protože zde jsou významnější horizontální vazby. Konečný cíl - automatizovat tvorbu dokumentace v celém rozsahu je tedy dosti vzdálen a vyžádá si ještě mnoho úsilí. Aby tato snaha nevyzněla neprázdno, je nutno se už teď snažit také o propagaci tohoto přístupu, jinak by se mohlo stát, že budeme muset zůstat u klasické formy prostě proto, že to bude na nás vyžadováno.

Cílem mého příspěvku je proto také podnítit zájem

dalších, a pokud se to podařilo, splnil svůj účel.

8. Literatura

/1/ Vepřek J., Bažantová J., Ježowicz E., Nedoma J., Pražák I.: Metody analýzy struktury systému, EML, Praha 1975

/2/ Ing. P. Miklica, Ing. V. Čimbura: Poznámky ke standartizaci v programování, dokumentaci a evidenci, sborník semináře Programování 78 Havířov 1978

/3/ Zuchovickij S.I., Radčíková I.A: Matematické metody síťové analýzy, SNTL, Praha 1972

/4/ Tiernan J.C.: An Efficient Search Algoritnm to Find the Elementary Circuits of a Graph, Comm. ACM, 1970, Vol. 13, Num. 12, str. 722-726

/5/ Vrbenský K.: Programový systém pro opravy zdrojových programů na zařízení DASO, Sborník Metody programování počítačů III. generace, Havířov 1977

/6/ Doporučené metodické pokyny - Dokumentace ASŘP FMTIR, Praha 1977

Příloha : Ukázka grafu vazeb programu SDISTRM1 vytištěná programem SPINNE.

```
ARACUNVD (M) => NUMBCD
         (M) => PRIME
         (M) => REQUATE
         (M) => TENME

MRACTIST (M) => HEAD
         (L) => ARACUNVD (M) => NUMBCD
                         (M) => PRIME
                         (M) => REQUATE
                         (M) => TENME

KRASORT  (M) => PRIME
         (M) => TEAME

SDISTRM1 (M) => HEAD
         (L) => SMARTIST (M) => HEAD
                         (L) => ARACUNVD (M) => NUMBCD
                                         (M) => PRIME
                                         (M) => REQUATE
                                         (M) => TENME
         (L) => KRASORT  (M) => PRIME
                         (M) => TENML
         (M) => NUMBCD
         (M) => PRIME
         (M) => KAMUL
         (M) => REQUATE
         (L) => SPGET
         (M) => TENME
```

VYSVĚTLENÍ POUŽITÝCH SYMBOLŮ

```
A (L) => B     MODUL A VOLÁ MODUL B
B <= (L) A

A (M) => M     V TEXTU A JE POUŽITO VOLÁNÍ MAKRA M
M <= (M) A
```