JE PARAMETRICKÝ PROGRAM EFEKTIVNÍ ?

Vít Vojáček, Jiří Pechlát

1. Úvod

Parametrické programy patří dnes mezi nejrozšířenější programovací prostředky. Dosažené výsledky, dokumentované např. v /4/ a /6/ ukazují, že jejich obliba je oprávněná. Snadnost a pružnost zadávání je /měla by být!/ charakteristickým rysem parametrických programů. Jejich přínos je téměř hmatatelný, pochopitelný i laikovi. Je neporovnatelně snazší obhajovat zavedení parametrického programu než databanky nebo dokonce věci na pohled tak abstraktní, jako je strukturované programování. Přitom náklady na získání hotového parametrického programu jsou dosud zanedbatelné ve srovnání s cenou standardního programového vybavení. Při vytváření vlastních parametrických programů je celá otázka složitější, ale programátor raději přistoupí na šibeniční termín, protože se domnívá, že se mu to později vyplatí, a protože ho tvorba parametrického programu láká jako ne zcela vědní problém. Tady asi je jeden z kořenů rozporného stavu, kdy na jedné straně aplikační programátoři houfně vytvářejí více či méně obecné parametrické programy a na druhé straně si více či méně oprávněně stěžují na nedostatek kapacit. To však je možné téma na jiný příspěvek.

2. V čem je problém?

Stručně řečeno: v tom, že parametrický program je mnohdy hodnocen pouze z hlediska bezprostředního přínosu. Méně pozornosti se věnuje dalším důsledkům, které sebou nese zavedení a používání parametrického programu. A ještě méně si uvědomujeme, kolik kapacit bylo a je spotřebováno na jejich tvorbu.

Nejlepší situace je tam, kde parametrický program přestal být módou a stal se rovnocenným programovacím nástrojem, který lze organicky začlenit do systémově pojatých řešení, jak dokládají /1/, /2/ a /3/. Tento přístup však vyžaduje dobrou organizaci a podporu ze strany vedení a to není všude obvyklé. Častěji se chytáme parametrických programů jako onoho pověstného stébla ve

chvíli, kdy už nevidíme jiné východisko. Přesto se tohle nouzové řešení obyčejně ukáže jako efektivní, což je jen dalším důkazem životaschopnosti parametrických programů. Problémy se objeví později: nedostatečná garance a údržba programu, nedostatečná nebo žádná dokumentace. Zjišťujeme, že program neumí zdaleka všechno, co jsme si od něho slibovali a co od nás uživatel požaduje. Občas dělá něco navíc, např. tvrdošíjně tiskne vlastní hlavičku, a uživatel stejně tvrdošíjně trvá na tom, že ji na sestavě nechce.

Dalším nepominutelným faktem, doprovázejícím zavedení obecného parametrického programu je snížení nároků na kvalifikaci programátora. Jde o stejný psychologický efekt, jaký přinesla zavedení výpočetní techniky do administrativní činnosti: ztráta pocitu vlastní důležitosti. Necitlivé "násilné" prosazování může vést ke ztrátě kvalitních programátorů, nedůslednost v prosazování vede jen k částečnému používání. S tím souvisí i tříštění programovacích prostředků.

Zvláštní skupinu představují problémy, spojené s dalším rozvojem výpočetního střediska a ASŘ. Je žádoucí vědět předem, že použitý parametrický program nebude brzdou dalšího rozvoje třeba proto, že je nepřevoditelný do vyššího operačního systému nebo na další počítač.

3. Vytváření parametrického programu

Otázka na jeho přínos je v tomto případě daleko akutnější, problémy spojené s jeho provozováním zůstávají.

Můžeme to dokumentovat na našem programu Tabulky. Účelem tohoto programu je podle zadání vytvořit jednu či více matic /tabulek/, přičemž při splnění zadaných podmínek pro sloupce a řádky se pro příslušný prvek matice provede zadaný výpočet. Vlastní realizace výpočtu je rozdělena do tří kroků:

- generace řídící části v jazyce assembler
- překlad řídící části
- vlastní provedení výpočtu

Použitý algoritmus při výpočtu je poměrně komplikovaný, pro-

tože základem při testu není řádek či sloupec matice, ale testovaná informace zpracovávaného souboru.

To byl hlavní důvod, proč padlo rozhodnutí o vytvoření parametrického programu.

Jak však probíhala realizace? Program vznikal v době převodu našeho ASŘ z počítače druhé generace na počítač generace třetí. Aby byl v našem středisku maximálně využíván, nebylo možno jeho realizaci pozdržet. Informace o počítačích třetí generace pocházely výhradně ze školení a pro vybavení programu uživatelským komfortem naprosto nedostačovaly. Z tohoto důvodu bylo nutno přistoupit na dost značná omezení:

- vstupní soubor smí být pouze na magnet. pásce,
- maximální délka bloku vstupního souboru je omezena,
- nejsou k dispozici žádné pomocné seznamy pro tvůrce zadání,
- řízení činnosti programu není možno provádět pomocí systémových prostředků, ale speciálními štítky zadání /máme na mysli např. potlačení opisu zadání/,
- není ošetřena chyba při výpočtu vznikající zadáním chybné věty /příčinu je pak nutné hledat ve výpise paměti/.

Další problém, který se nám nepodařilo vyřešit, bylo vyčlenění pracovníků pro vytvoření programu. Situace nakonec dopadla tak, že program řešil a vytvořil výhradně jeden pracovník, systémový programátor, který však nemohl zcela opustit své další úkoly. Z toho však vznikly další problémy:

- nebyla provedena dostatečná analýza,
- program vznikal překotně /byl napsán v průběhu jednoho měsíce, laděn rok/,
- řádek matice byl zredukován na délku řádku na tiskárně,
- všechny řádky matice musí mít stejný formát,
- při vkládání nebo vynechání řádků či sloupců v matici je třeba v zadání tyto ručně přečíslovat,
- neexistuje prakticky žádná dokumentace,
- a v neposlední řadě utrpěla vlastní práce se systémem.

V době, kdy byl program Tabulky již plně využíván, se obje-

vil další velmi závažný problém. Program byl napsán a odladěn pod systémem µOS. Při přechodu na systém OS je třeba program celý přepracovat, neboť porušuje některé základní zásady programů v OS /je to např. práce s operační pamětí, způsob využívání registru 13 ap./. Tím se tedy program Tabulky stal brzdou převodu zpracování z µOS do OS. A i když k přepracování nakonec přistoupíme, nemůžeme se vyhnout v žádném případě chybám, kterých jsme se dopustili již při zavádění do µOS. Opět bude vznikat program pouze na podkladě znalostí ze školení a na jeho přepracování bude zase možno vyčlenit pouze jednoho necelého pracovníka.

4. Některé zkušenosti s používáním

Na druhou stranu jsme si však udělali přehled rozsahu používání tabulek a byli jsme celkem příjemně překvapeni. Zjistili jsme, že v současné době se u nás tímto programem vytváří 80 různých typů pravidelně zpracovávaných tabulek pro nejrůznější oblasti hromadného zpracování dat. Nejvíce tabulek se vytváří pro agendu JEP /34/ a pro agendu evidence výrobního programu /23/. Navíc má používání tohoto programu další příjemnou vlastnost, a tou je snadná zvládnutelnost zadání. Proto nový programátor projde obvykle vývojovým stupněm "zadávání tabulek", což mu zajistí rychlé uplatnění v programátorském oddělení, ale na druhé straně zpomalí jeho odborný programátorský růst.

Vedle tabulek používáme Printer, výběrový, výpočtový, sumarizační a tiskový program s vestavěným sortem.

Také ten byl zaveden opožděně v době, kdy už některé agendy fungovaly, jiné byly rozpracovány. Řešitelé si zvykli na odlišné přístupy a parametrické programy u nás vlastně nikdy neměly význam jako jednotící prvek. Jsou používány jen některými programátory a jen v některých agendách. Přesto je efekt značný. Pro ilustraci: v agendě JEP z bezmála 80 výstupních sestav jen 8 nevyužívá Tabulky ani Printer!

Výrazným nedostatkem je nesourodost obou programů. Mají zcela odlišnou filozofii a techniku zpracování a samozřejmě diametrál-

ně odlišné zadávání. To vede ke zbytečnému tříštění programovacích prostředků. /Mimochodem, odstrašující příklad: v jistém období pracovalo ve VS 5 aplikačních programátorů, kteří používali pěti různých prostředků k vytváření výstupních sestav./

Vedle zmíněných dvou obecných parametrických programů existují u nás už od počátku parametrické programy "šité na míru" určité agendě. Jde v zásadě o 4 typy programů:
- výběrové /výběr vět a výběr polí/,
- sumarizační /víceúrovňové/,
- výpočtové /jednoduché aritmetické operace v jedné větě/,
- tiskové.

Zadání je jednoduché vzhledem k omezené obecnosti. Možnost výběru polí do jednotné věty délky 100 bytů však dává nepřeberné bohatství variant. Omezení obecnosti na jednu agendu a na výpočty statistické povahy přineslo výrazně nižší pracnost ve srovnání s obecnými programy. Nároky na tvorbu jsou srovnatelné s vytvořením jednoúčelového programového vybavení. Zvolené řešení je přitom daleko pružnější a umožňuje vypracovat zadání i složitých sestav během jednoho dne. Další předností je snadná přenositelnost do jiné agendy podobné povahy. Stačí změnit deklaraci vstupu ve výběrových programech.

Ani tento způsob tvorby parametrických programů však není obecným jevem v našem VS. Jde opět o iniciativu jednoho programátora.

5. Závěr

Jaká je tedy efektivnost parametrického programu? Zřejmě o tom rozhodují hlavně tři faktory: stupeň obecnosti programu, náklady na jeho získání nebo vytvoření a další údržbu a šíře použití. Budeme to dokumentovat na dvou obecných parametrických programech, které používáme v našem středisku.

Program Tabulky byl vytvořen v našem středisku a je používán výhradně u nás. Zvážíme-li investice vložené ve formě lidské práce do jeho vytvoření, pak z hlediska využívání programátorských kapa-

cit nepřinesl valnou úsporu. Jiná situace by ovšem nastala, kdyby program i jeho dokumentace byly na také úrovni, aby ho bylo možno bez obtíží využívat i v dalších střediscích.

Druhým hlediskem je množství strojového času, potřebného k odladění takového programu. Je značně velké, což je přijatelné pouze u programu s vysokou četností používání, jakou nemůže zajistit jedno VS.

Třetím hlediskem je rychlost, jakou program pracuje. Parametrický program je vždy pomalejší než jednoúčelový. Bohužel v řadě případů roste spotřeba času nad únosnou mez. Tabulky jsou v tomto směru odstrašujícím příkladem.

Jiná situace je u programu Printer, který nebyl vytvořen v našem středisku. Proto je pro nás zajímavé pouze třetí hledisko, rychlost zpracování, která je u Printeru vyhovující.

Srovnáme-li tedy program Tabulky s programem Printer, můžeme konstatovat, že program Printer je efektivní pro naše středisko hlavně proto, že jsme ho získali hotový a nemuseli ho sami vytvářet. Tuto zkušenost lze zobecnit, a říci, že obecný parametrický program by nemělo vytvářet středisko ve vlastní režii.

Nechceme tím říci, že by vůbec neměly vznikat obecné parametrické programy. Musí však existovat někdo, kdo zajistí využitelnost obecného parametrického programu ve více střediscích. To znamená zjistit požadavky na program, stanovit omezení, určit operační systémy, pod kterými má program pracovat, zajistit vydání dokumentace, zajistit garanci atd. Přitom tento koordinátor nemusí program sám vytvořit. Toho se může ujmout určité středisko formou úkolu, pokud bude mít volné řešitelské kapacity. V našich podmínkách může být tímto koordinátorem jedině organizace NOTO. První kroky tímto směrem byly již učiněny. Je to např. zakoupení a převedení programu CULPRIT, který má navíc proti všem tuzemským parametrickým programům výhodu v úzké vazbě na IDMS.

Jedině tímto způsobem mohou vznikat obecné parametrické programy, které šetří programátorské kapacity a nejsou brzdou dalšího vývoje využívání výpočetní techniky.

Parametrické programy obsluhující jednu egendu nebo provádějící jednoduché funkce se jeví efektivní i při omezeném využívání v jednom středisku. Svoji roli tu zřejmě hraje, kromě jejich omezené obecnosti, také skutečnost, že tvůrce takového programu /obvykle garant agendy/ vychází z hluboké znalosti řešeného problému. Ale ani zde se nevyplácí přílišná obecnost zadání, která neúměrně zvyšuje pracnost naprogramování, zadání a prodlužuje zpracování na počítači.

Literatura:

/1/ Streit Dušan : Parametrické programy uživatelského software v systému zpracování dat, sborník Programování 80

/2/ Streit Dušan : K problematice parametrických programů, MAA 7/1981

/3/ Horyna Jaroslav, Horynová Eliška : Parametricky řízené typové programové prvky, sborník Programování 80

/4/ Štěpánek Zbyněk: Zkušenosti s používáním parametrických programů v oblasti zpracování dat, sborník Programování 80

/5/ Vojáček Vít : Zadávání tabulek, rukopis

/6/ Tomka Zdeněk : Srovnání progresivních metod k vytváření tiskových sestav, sborník Programování 81

/7/ Burda Karel : Stručný popis funkce a možností programu Printer, rukopis