Pavel Šimoník

OKR Automatizace řízení, k.ú.o.

O NĚKTERÝCH OTÁZKÁCH VZÁJEMNÉ KOMUNIKACE MEZI SYSTÉMOVÝM ANALYTIKEM A PROGRAMÁTOREM

Úvodem poznamenávám, že se ve svém příspěvku nehodlám zabývat otázkou často diskutovanou na minulém semináři, zda je pro kvalitní návrh systému nebo subsystému a pro vypracování navrženého souboru programů vhodné či dokonce nutné nejtěsnější spolupráce systémového analytika s programátorem v rámci jednoho pracovního týmu. Vycházím ze skutečnosti, že v naší organizaci, stejně jako v řadě jiných, tvoří systémová analýza a programování samostatné útvary s rozdílnou specifikou práce, a pojednám z tohoto hlediska o problémech vzájemné komunikace. Domnívám se, že vyvozené závěry budou mít svůj význam i pro pracovníky organizací s jinou organizační strukturou, a to proto, že bezprostředně souvisejí s problematikou dokumentace.

Poněvadž mám za to, že můj referát by měl vycházet ze zkušeností, které jsme v této oblasti získali, budu předpokládat určitou úroveň zavedené standardizace v tom rozsahu, jak o ní hovořili naši pracovníci na předcházejících seminářích, včetně prostředků, které máme momentálně k dispozici. Jde zejména o metodu normalizovaného programování a generátor normalizovaných programů se všemi jeho možnostmi. V souvislosti s tematem se samozřejmě dotknu i některých otázek, které u nás dosud nejsou standardně řešeny. Poněvadž mi jde především o oblast hromadného zpracové-

ní informací a poměrně velkými soubory dat s častou frekvencí zpracování, všimnu si i některých detailů, které by při jednorázovém zpracování programu byly do jisté míry podružné, ale v našem případě nejsou zanedbatelné z hlediska maximální efektivnosti programu.

Vzájemná spolupráce mezi systémovým analytikem a programátorem bude v zásadě zahrnovat tři fáze práce na programu. První, přípravná fáze, předchází vypracování specifikace programu. Musíme předpokládat, že systémový analytik je dokonale informován o možnostech daného počítače a že zná pro svou potřebu používaný firemní software. Méně se již dá předpokládat, že bude znát stejně dobře uživatelský software, alespoň v prvním období po jeho zavedení. Rovněž tak nebude mít většinou podrobné znalosti o možnostech používaného programovacího jazyka. Je proto účelné, aby před vypracováním specifikace konsultoval koncepci programu s programátorem. U nás se tak děje formou ústní konzultace analytika s vedoucím programátorem týmu, kterému je přidělено vypracování programů dané agendy.

Druhou fázi představuje vypracování specifikace programu. Poněvadž specifikace je zároveň nutnou součástí programové dokumentace, je nezbytné, aby kromě informací nutných a postačujících pro dokonalé vypracování programu splňovala i další podmínky, kterými se budu podrobněji zabývat.

V naší organizaci máme vypracovaný standard, obsahující všechny závazné náležitosti specifikace, který mohu na požádání poskytnout zájemcům k nahlédnutí. Nebudu se proto ve svém referátě zabývat podrobným popisem tohoto standardu. Omezím se na stručný výčet jednotlivých bodů, při čemž zdůrazním ty skutečnosti, které jsou podle mého názoru pro vypracování dobrého programu a jeho zavedení do rutiny stěžejní.

V prvé řadě musí specifikace obsahovat dokonalý popis všech vstupních a výstupních souborů včetně specifikace výstupních sestav. V tomto bodě zasluhují pozornosti dvě vě-

ci. Pro úspěšné zavedení programu do rutiny je velmi důležitý spolehlivý odhad maximálního rozsahu souboru, který programátor nutně musí znát pro stahování dostatečného prostoru, pokud pracuje s diskovými soubory. U setříděných souborů je důležité uvádět kromě skutečné setříděnosti souboru rovněž ta třídící hlediska, která jsou pro daný program rozhodující. Praxe ukazuje, že při případném omylu ve specifikaci programu je možno v průběhu ladění na základě této informace rychle odhalit chybu.

Dále bych se rád zmínil o dvou problémech, které náš standard dosud nepostihuje. Často se stává, že v popisu věty se daná položka vyskytuje pod jiným názvem než v textové části specifikace programu. Programátor, který není specialistou v příslušné oblasti, nepozná, že jde o synonyma z hlediska odborné terminologie používané systémovým analytikem. Poněvadž v zájmu pružného využívání programátorské kapacity bude vždy nutné, zejména při nárazových pracích, zadávat vypracování jednotlivých programů i mimo stabilní programátorský tým pro danou agendu, považuji za nezbytné dosáhnout v tomto ohledu naprosté jednoznačnosti.

V souvislosti s tím, že v cobolských programech představuje popis dat podstatnou část programu, považuji za vážný nedostatek, že dosud nemáme v rámci jednotlivých agend vypracovány standardní rozpisy vět včetně standardních identifikátorů, uložené na souboru vět pro příslušnou agendu ve zdrojové formě tak, aby je bylo možno snadno zkopírovat do příslušného programu. Mám za to, že bychom měli při tvorbě nových systémů využít v tomto ohledu zkušeností z jiných organizací, kde této metody používají.

Popis algoritmu daného programu má podle našeho standardu odpovídat normalizovanému programování. To nepochybně podstatně zlepšilo "čitelnost" takto psaných specifikací. Přesto se však v jednom bodě náš standard v praxi nedodržuje zcela důsledně. Jde mi o požadavek, rozdělit složitější algoritmy na několik logických celků, z nichž každý bude řešit některý částečný problém daného programu. Podle

požadavku standardu by mělo být ve specifikaci zvláště vyznačeno, které z těchto celků budou podléhat dle předpokladu systémového analytika častějším změnám během života programu. Nejde jen o to, že takto pojatý program je podstatně odolnější vůči změnám, ale též o to, že jednou napsaná specifikace programu má zůstat i po několikanásobné změně dostatečně srozumitelná. V současné době jsme vypracovali předběžný návrh na doplnění standardu v tomto smyslu: Složitý a náležitě rozčleněný algoritmus má být ve specifikaci doplněn hrubým vývojovým diagramem, kde řešení jednoho problému je zobrazeno jedním blokem diagramu. Tomuto bloku by pak v popisné části specifikace odpovídala jedna, případně i neúplná, strana textu. V případě větší změny v programu vyznačí systémový analytik v dodatku ke specifikaci programu, kterých bloků se změna týká, a nahradí příslušné stránky původní specifikace novými. Jestliže změna postihuje jen málo textovou část původní specifikace, uvede analytik v dodatku jen změnu odstavců, případně vět v původní specifikaci, s tím, že opět vyznačí, kterých bloků se změna týká. Při dodržení těchto zásad bude i u programů s velkým počtem změn ve specifikaci náležitý přehled o tom, které části programu byly změnami postiženy, takže případný další zásah do programu bude mnohem snadnější a relativně bezpečný. Rovněž možnost dobré orientace ve specifikaci takto doplňované příliš neutrpí.

Bude-li nadto vlastní program vypracován tak, že jednotlivým blokům ve vývojovém diagramu dodaném analytikem budou odpovídat subrutiny programu nebo vyvolávané moduly, bude mnohem snadnější udržovat v aktuálním stavu i podrobný vývojový diagram programu, který rovněž tvoří závaznou součást dokumentace.

Je nepochybné, že mnohé z uvedených problémů odpadnou nebo se podstatně zjednoduší, jakmile začneme ve větší míře využívat v programových specifikacích rozhodovacích tabulek spolu s překladačem rozhodovacích tabulek, který v současné době vyvíjí naše oddělení služeb. Zatím používáme ve specifikacích programů rozhodovacích tabulek jen k popisu

některých částí algoritmu. Proto se touto otázkou ve svém dnešním referátě nezabývám.

Nakonec bych se rád zmínil v souvislosti s problematikou specifikací ještě o jednom. Některé programy zpracovávající rozsáhlé soubory dat se někdy větví v závislosti na poměrně velkém počtu podmínek, které se postupně testují. Efektivnost programu bude tím lepší, čím lépe bude posloupnost za sebou následujících testů uspořádána podle klesající četnosti předpokládaného výskytu testovaných podmínek. Z informací, které má programátor obvykle k dispozici, může jen , někdy dostatečně spolehlivě usoudit, která podmínka bude splněna častěji a která méně často. Bylo by účelné, aby v takových případech byla specifikace doplněna o scházející informace, pokud to může analytik na základě znalosti věcné problematiky učinit.

Třetí a poslední oblast vzájemného styku mezi programátorem a systémovým analytikem představuje odzkoušení programu na vhodném vzorku dat dodaných analytikem a zavedení programu do rutinního zpracování. Z našeho hlediska bude zajímavý zejména poslední bod u programů zpracovávaných na počítačích, které pracují pod operačním systémem. Pro účely rutinního zpracování je nutno celé zpracování rozdělit na jednotlivé pracovní postupy (joby), které by měly tvořit jednak logicky ucelené úseky celého zpracování agendy, jednak by měly vyhovovat požadavkům provozu, a to jak požadavkům zadavatelů zpracování, tak požadavkům vlastního provozu počítače. Pro programátora z toho vyplývá úkol, vypracovat příslušné procedury s využitím všech možností jobového jazyka tak, aby zpracování jobů bylo provozně jednoduché a co nejméně citlivé na možné omyly obsluhy. Zatímco máme poměrně dobrý standard pro tvorbu procedur, pro jejich popis a dokumentaci, zůstává samotné řešení problému ponecháno případ od případu vzájemné konzultaci všech zúčastněných, aniž byly u nás zatím stanoveny pevné zásady pro tento úsek práce. To je po mém soudu na škodu věci, protože někdy dochází ad hoc k připomínkám, které v tomto stadiu práce mohou být již jen částečně uspokojeny. Mám za to, že systémový analytik, který

řeší svůj úkol komplexně včetně zavedení rutinního zpracování a včetně organizačních otázek, které jsou s tím nezbytně spojeny, by měl mít pro tuto poslední fázi společné práce s programátorem zcela ujasněnu svou vlastní koncepci a dostatečně přesnými požadavky na její realizaci. Rovněž se domnívám, že i v této fázi práce by měl zůstat systémový analytik jediným partnerem programátora, který na něho klade požadavky a konzultuje s ním případné problémy. Na podporu svého stanoviska si dovolím uvést tyto důvody:

Již při koncipování programu musí být jasno, v jaké souvislosti bude program zpracováván, zda bude pracovat v jediné pevné verzi, či zda bude v závislosti na zadaných parametrech možno nebo nutno jeho činnost poněkud měnit. Dále je již v této fázi práce třeba mít vyjasněnu formu a způsob zadávání parametrů, zda a v jaké formě budeme od programu požadovat informace, zda je třeba některé momenty jeho činnosti kontrolovat pomocí případného návratového kódu. Máme-li tyto otázky ujasněny, máme zároveň v ruce prostředek, který umožňuje zadávat ke zpracování a také v průběhu zpracování do jisté míry kontrolovat job, v němž je náš program zařazen spolu s jinými programy agendy.

Zůstává zatím otevřená otázka, v jaké formě zadávat požadavek na vytvoření procedury. S ohledem na to, co bylo řečeno v předcházejícím odstavci, je zřejmé, že k vypracování postačí doplnit informace tam uvedené takto: Stanovit jednotlivé kroky v rámci procedury, popsat parametry a určit způsob zadávání, stanovit ,za jakých podmínek se provedou respektive přeskočí jednotlivé kroky v rámci procedury, stanovit způsob vstupu děrnoštítkových souborů a způsob výstupu tiskových souborů, rozhodnout o katalogování výstupních souborů a také o tom, které údaje mají být případně zadávány pomocí symbolických parametrů v rámci procedury.

V podrobnostech by měly být konzultovány před zadáním s programátorem některé speciálnější otázky, na př. možnost využití zřetězených souborů na vstupu, zajištění možnosti opakovat job případně jeho část od jistého kroku

při dodržení minimálních požadavků na zásah pracovníků provozu. Poslední požadavek je důležitý pro případ neúspěšného zpracování jobu vlivem chybné činnosti systému, chybné obsluhy nebo jiné vnější příčiny. Domnívám se, že prostý písemný stručný výčet informací a požadavků shora uvedených může dobře posloužit jako svého druhu specifikace pro vytvoření procedury.

Netroufám si bez předběžné diskuse posoudit, zda je vhodné předepisovat některé častěji užívané postupy při tvorbě procedur jako standard, či je vhodnější ponechat navrhovateli i tvůrci procedury poměrně značnou volnost, tak aby procedura byla co nejtěsněji přizpůsobena svému účelu.

Ve svém referátě jsem se dotkl některých otázek komunikace mezi analytikem a programátorem, zejména takových, jejichž zanedbávání je zdrojem častěji se vyskytujících chyb při tvorbě programů. Budu považovat svůj úkol za splněný, jestliže případná diskuse přinese alespoň několik nových podnětů ke zlepšení tohoto úseku naší práce.