Kompletování a údržba velkých programových systémů

Jan Sokol, VÚMS,kúo, Praha

Na téma technologie programátorské práce už bylo napsáno mnoho článků, existuje řada více méně propracovaných metod či technologií a i v Československu se touto problematikou zabývá několik pracovišť. O technologii programování se píší knihy a vyučuje na školách. Nicméně do praxe se výsledky této činnosti zatím příliš neprosadily. Jedním z důvodů je jistě setrvačnost, neochota učit se novým věcem. Domnívám se však, že je tu ještě jeden dosti podstatný důvod věcný. Všechny teoretické práce se zatím týkají pouze technologie programování, případně analýzy problémů, a tedy pokrývají jenom část té problematiky, kterou musí řešit každé pracoviště, produkující a dodávající rozsáhlé programové systémy.

V následujícím se pokusím
1. vysvětlit povahu a rozsah problémů, které je třeba v této souvislosti řešit;
2. stručně popsat konkrétní řešení, užívané ve VÚMS při kompletování programového vybavení DOS-4 a dalších programových celků.

1. Kompletování, distribuce a údržba

Z hlediska našeho tématu lze, domnívám se, úlohy a programové systémy rozdělit do tří velmi hrubých kategorií:

A) ty, které zvládne jeden člověk, případně několik lidí pod přímým vedením jednoho člověka;

B) ty, které se vhodnou analýzou a organizací práce dají rozdělit na několik částí s vlastnostmi podle A;

C) zbývající, které se takto rozdělit nedají.

Převážná většina úloh, které se u nás i ve světě na počítačích řeší, a většina programových systémů, které vznikají, patří naštěstí do kategorie A. Práce na takových úlohách má jistě své problémy jako ostatně každá trochu náročnější technická činnost. Vztahy mezi analýzou a programováním, testování, předávání do provozu, reklamace a opravy chyb, rozšiřování a změny funkcí - to všechno jsou obtížné problémy, do nichž se však v tomto příspěvku nechci pouštět. Rozhodující se mi zdá to, že jsou zvlá-

nutelné či přinejmenším přehlédnutelné jedním člověkem. Pravda, ten člověk nemůže být kdokoli, musí mít jisté schopnosti a vlastnosti, ale je to stále jeden člověk.

Přednosti a výhody této kategorie úloh jsou dávno známy a před několika lety se dokonce psalo o metodě s trochu honosným názvem "tým vedoucího programátora" /chief programmer's team/. Úspěšný manažer jedné anglické softwarové firmy mi před několika lety řekl, že celá jeho metoda spočívá v tom, pro každou úlohu najít takového člověka a dát mu takové podmínky, aby ji zvládl sám. Pokud na něčem zvlášť záleží, vyplatí se mu najmout takové lidi dva nebo tři a nechat je dělat totéž, aby si nakonec mohl vybrat nejlepší řešení. "Než by se ti tři dohodli, co a jak budou dělat, je každý z nich jednotlivě s prací hotov." Vynikající programátor skutečně udělá práci za deset průměrných, výsledek je často lepší a dřív hotov.

Bohužel ne všechno, s čím se setkáváme, spadá do této "ideální" kategorie. Buď proto, že úloha je pro ty programátory, které máme, prostě příliš rozsáhlá. Nebo je to "železná kráva", úloha, která se bude udržovat, upravovat a měnit deset let a víc. Nebo vyžaduje součinnost mnoha lidí s různými znalostmi a schopnostmi, a tak dále. I v tomto případě se pokusíme úlohu rozebrat a rozdělit tak, aby se vešla aspoň do kategorie B našeho předchozího dělení. Problémy s tím, jak celek udržet pohromadě, se sice objeví, ale nebudou tak velké. Budou pochopitelně tím větší a vážnější, čím těsnější budou vazby a souvislosti mezi jednotlivými částmi. Nicméně i v tomto případě můžeme kompletování, testování, distribuci a údržbu považovat za část práce řešitelů úlohy, tj. analytiků a programátorů, kteří na úloze pracují.

Další zlom nastává v okamžiku, kdy jsou splněny následující tři podmínky:

- na úkolu pracují desítky programátorů, případně na místně oddělených pracovištích;
- programový systém se musí udržovat a přizpůsobovat potřebám uživatelů mnoho let;
- systém se bude provozovat na mnoha místech a v různých podmínkách.

Tím je, domnívám se, naplněna "skutková podstata" kategorie C a řízení projektu je třeba uchopit jiným způsobem.

Pokud se včas, to jest hned na začátku rozpozná, o jak složitou věc se jedná, je aspoň v zásadě možné využít výsledků teoretických prací, o nichž jsem se zmínil na začátku, zejména metod modulárního programování či "programování ve velkém" /programming in the large/. Metoda byla i u nás několikrát popsána a propagována, z programových produktů, které ji podporují, bych rád uvedl aspoň EPS /VÚSEI-AR Bratislava/ a SNAP /VÚLS Praha/. První z nich, implementovaný na počítačích SMEP, ADT i JSEP, zahrnuje i vlastní programovací jazyk, blízký jazyku MODULA 2. Druhý, který je k dispozici na počítačích JSEP /DCS-4 i CS/ dovoluje programování v různých jazycích /Assembler, PL/I, Pascal atd./.

Podaří-li se vedení projektu tuto metodu důsledně prosadit, získá tím, aspoň v prvních fázích projektu značné výhody. Na projektu je možné pracovat "shora dolů", rozhraní mezi jednotlivými částmi jsou přesně popsána a zakotvena v programech a metoda dává i cenné diagnostické a dokumentační prostředky. Tím se dá značně urychlit vlastní vývoj a ladění programů, výsledný produkt může být spolehlivější a také následné modifikace budou snazší. Největší praktická obtíž při uplatňování těchto nesporně dobrých metod tkví právě v důsledném prosazení. To je možné u menších týmů s neformální autoritou, ale právě u velkých projektů nesmírně obtížné. Kromě toho je účinnost této metody největší na začátku, tj. ve fázi vývoje, programování a ladění. Při testování, distribuci a údržbě hotového celku vznikají ještě další problémy, které dosud známé technologie a metody neřeší.

Velký programový systém se skládá ze stovek a tisíců částí, v typickém případě modulů, maker a procedur. U systémů kategorie C už není člověka, který by o všech věděl, byl je schopen správně sestavit, odzkoušet a distribuovat. Proto je tyto činnosti třeba vyčlenit z pracovní náplně běžných programátorů a svěřit zvláštní "profesi", které /z nedostatku lepšího názvu/ říkáme "kompletátor". V jistém ohledu má činnost kompletátora hodně společného s činností provozních programátorů běžného střediska, vyžaduje ovšem značnou kvalifikaci a zkušenosti. Na svědomitosti a schopnostech kompletátorů pak hodně závisí kvalita konečného produktu.

Vlastní kompletace vychází z tzv. specifikací, tj. seznamu finálních složek systému, který zhruba odpovídá výrobnímu programu třeba strojírenského podniku. Jsou to ty programy, procedury atd.,

o nichž se zmiňuje uživatelská dokumentace. Účelem specifikací je právě to, aby se na žádný z nich nezapomnělo.

Každou z těchto finálních složek je třeba při kompletování znovu "vyrobit", v typickém případě přeložit a spojit. K tomu slouží další pomocné informace, které plní funkci kusovníku ve strojírenské výrobě: obsahují údaje o tom, z čeho, v jakém pořadí a jakými pracovními kroky se má ta která složka "vyrábět". V typickém případě nějakého programu to bude třeba zpracování předprocesorem nebo generátorem, překlady a spojování a uložení do některé knihovny. Průběh všech těchto kroků je třeba dokumentovat a archivovat kvůli identifikaci a hledání chyb.

Zkompletovaný systém nebo jeho část je třeba standardním způsobem vyzkoušet. Zkušební příklady, které si dělá každý programátor pro ladění svých modulů, se k tomu zpravidla nehodí. Vhodnější jsou zkušební příklady, vytvářené jinými osobami, třeba v souvislosti s dokumentací, které prověřují i součinnost různých složek, a to pokud možno tak, aby se výsledek testu dal zase ar vyhodnotit automaticky.

Je-li výsledek testů uspokojivý, může se vytvořit distribuční soubor /obvykle na magnetické pásce/. Distribuční soubor musí obsahovat všecky složky podle specifikace, ale neměl by obsahovat pomocné složky, které sloužily jen pro jeho vytvoření /procedury, makra, zdroje atd./. Distribuční soubor musí být bezpečně označen /verze, modifikace, datum/ zase zejména proto, aby bylo možno identifikovat následné chyby. Při větším rozsahu distribuce je výhodné distribuovat novou verzi několika vybraným uživatelům a teprve na základě jejich provozních zkušeností přikročit k distribuci v celém rozsahu.

Většina programátorů je sice přesvědčena, že jejich programy jsou správné - přinejmenším od včerejška, vždy opravili poslední chybu - ale stará zkušenost říká, že v každém programu je chyba. U malých programů pro vlastní potřebu je pomoc snadná: chyba se opraví a je to. Docela jinak u velkých systémů. Kompletování, testování, distribuce a nasazení nové verze u uživatele je pracné, zdlouhavé a někdy i riskantní procedura. Proto je pro dodavatele i uživatele naprosto nezbytné každou zavedenou verzi průběžně udržovat tzv. záplatami /patch/, které budí hrůzu u programátorských puristů, ale bez níž žádný reálný systém

nemůže existovat. Zkušený programátor kromě toho ví, že dobře udělaná záplata je asi nejbezpečnější způsob, jak opravit chybu a nezavléct do programu jinou. Mimochodem řečeno, programové vybavení kosmických sond vždycky obsahuje funkci záplatování na dálku a prý na ní už někdy závisely i lidské životy.

Reklamace a hledání skutečných i domnělých chyb je denní chleba programátora. Říká se, že program, na který nechodí reklamace, asi nikdo nepoužívá. Podle zkušeností velkých světových firem, které můžeme našimi skromnými jen potvrdit, počet reklamací na začátku života velkého programového systému klesá, až se ustálí na jisté pevné hodnotě. Mění se pouze závažnost reklamovaných chyb.

Reklamace přicházejí v nejrůznějších podobách, od stručného sdělení, že to a to "nechodí", přes bohatě dokumentovaná hlášení o váze několika kilo až po detailní /a často správné!/ návrhy, kde a jak chybu opravit. Každou reklamaci je třeba nejdřív ověřit a identifikovat, tj. usoudit, zda jde skutečně o chybu a ve které části systému je, přesněji řečeno, kde se má opravit. To je velice obtížný úkol, v extrémních případech neřešitelný; u skutečně složitých systémů vznikají chyby, které nelze identifikovat. Pokud se chybu nepodaří reprodukovat, nezbývá než počkat, až se objeví v příznivější konstelaci okolností, které identifikaci umožní. Identifikovanou chybu je třeba opravit a je-li závažná, vzniká tlak, aby se opravila hned, to znamená záplatou. K tomu jsou potřeba přesné a spolehlivé informace, v jaké podobě byl např. modul distribuován někdy před rokem. Právě k tomu jsou určeny doklady a protokoly, vzniklé při kompletování verze. Záplata sama se ovšem stává součástí příslušné verze a musí se s příslušným komentářem zanést do distribučního souboru, aby se na ni nezapomnělo. Distribuce záplat probíhá častěji, někdy i "na počkání" po telefonu.

2. Kompletovací systém

Na závěr bych rád stručně popsal technologii kompletování, jak ji provozujeme ve VÚMS. Způsob, který používáme, není ideální: nevznikl najednou na základě promyšleného návrhu, ale vyvinul se postupně podle potřeb a zkušeností. V provozu se však celkem osvědčil a právě proto může být zajímavý jako příklad.

Hlavním produktem, který takto vzniká, je operační systém OS-4 a programové vybavení, které k němu patří. Je to poměrně silně provázaný celek s mnoha sty finálních složek, na němž pracovalo a pracuje 60 až 100 programátorů na několika místně oddělených pracovištích. Vývoj systému začal zhruba před deseti lety a stále pokračuje, i když jeho těžiště se přesouvá ke "koncovým" složkám, jako je řízení báze dat a telekomunikační a interakční systémy. Na operační systém navazuje u každého uživatele značný objem vlastního aplikačního vybavení, do něhož uživatelé investují značné prostředky. Systém jim proto musí zaručit poměrně dlouhou životnost a provozní spolehlivost aplikací.

Kompletování je samostatná pracovní činnost, kterou vykonává asi 5 programátorů jako svoji hlavní pracovní náplň. V současné době vydáváme jednu verzi za rok pro všeobecnou distribuci /asi 300 instalací/ a v průběhu roku ještě jednu nebo dvě verze, které slouží k provoznímu odzkoušování novinek u 10 až 20 vybraných uživatelů. Provozní opravy /záplaty/ se shromažďují průběžně a předávají asi jednou měsíčně do organizací NOTC. S kompletováním úzce souvisí vyřizování reklamací a údržba dokumentace.

Programátoři, kteří pracují na vývoji jednotlivých složek systému, předávají kompletační skupině hotové a vyzkoušené zdrojové soubory /knihy/ a procedury, které slouží k překladu a spojování finálních složek. Předávání probíhá stále na zvláštní knihovnu, z níž pracovníci kompletování čas od času /asi jednou za měsíc/ tyto "polotovary" přebírají. Přebírání se dokumentuje, takže lze kdykoli zjistit, kdy byl daný prvek naposledy převzat atd. Při přebírání se také kontroluje, na které vyšší složky bude mít změna prvku vliv, které bude nutno znovu přeložit atd. K tomu slouží m.j. "kusovníkový" soubor, který vzniká automaticky při překladech a spojování a který obsahuje přesné údaje o skutečné skladbě vznikajících složek systému. Protokoly o překladech se netisknou, ale ukládají v komprimovaném tvaru na disky; tisknou se až když je to potřeba pro nějakou reklamaci.

Vlastní kompletování a standartní testování je do značné míry automatizováno, vyžaduje však přesto několik desítek hodin strojového času. Proto se vyplatí věnovat i podpůrnému programovému vybavení pro kompletování velkou péči.

Kompletování, distribuce a údržba velkého programového systému představuje, jak je vidět, značný objem práce. Je to však práce, která se vcelku mnohonásobně vyplatí, protože dobře kompletovaný a udržovaný systém ušetří spoustu času a starostí na každé instalaci. Platí to zvlášť v podmínkách, kdy technické vybavení často nedosahuje potřebné úrovně spolehlivosti a uživatelé přesto chtějí přecházet na modernější způsoby interaktivního zpracování. Domnívám se, že popsaná technologie má značný podíl na tom, že programové systémy VÚMS si získaly jistou důvěru uživatelů, což je pro nás zároveň zavazující i do dalších let.