Redakční rada

VÝBĚR Z DISKUSE NA SEMINÁŘI 1976

Při loňském semináři byla v diskusi řečena řada myšlenek a námětů, které se nám líbily a o nichž si myslíme, že by neměly zaniknout okamžikem, kdy byl ukončen seminář. Proto jsme vybrali některé názory týkající se zhruba těchto okruhů: programovací metody, vztah analytik-programátor, programovací jazyky, programátorská profese. U citátů pro technické obtíže s autorizováním jednotlivých výroků autory neuvádíme.

Redakční rada.

... Pomalejší začátek, který při tvorbě programu vyplývá z delšího přemýšlení, se ve většině případů dožene. Dvakrát měř, jednou řež. Kdo se bude hned hnát na počítač a nemá to řádně rozmyšlené tedy z těch podhledů a nadhledů, pojede zpravidla ze začátku velice rychle a pak začne zadrhávat při ladění, při chybách, zatím co dobře uspořádaný, logicky sestavený program má těch problémů daleko méně. Myslím si, že zabývat se detaily bez vymezení celku se dá charakterizovat, jako zbabělý útěk před problémem abstrakce, kterému se chtě nechtě musíme někdy věnovat. ...

* * *

Kdo dneska kreslí vývojové diagramy? Kde jste to viděli na programátorských pracovištích? Když někdo potřebuje něco logicky obtížnějšího, namaluje si to na kousek papíru a pak to stejně zahodí, protože to napíše v Cobolu, ve kterém je to daleko čitelnější. Aby se vyhovělo předpisům, tak se vývojové diagramy malují na konec.

* * *

Projekt má být funkce schopný do dvou let. Není-li to, pak pravděpodobnost, že se dokončí, se velice zmenšuje, protože v třetím a čtvrtém roce nastává nutnost reorganizace...

* * *

Velmi často se uvádí jako argument proti normalizovanému programování a proti standardizaci to, že se tím potlačuje individualita programátora. Ale podívejme se na to takhle. Co je zájmem podniku, zájmem společnosti? Jestli jenom to, aby programátor byl spokojen s tím, co dělá, aby odcházel domů s pocitem, že se dnes pěkně vyřádil na programu anebo jestli je v zájmu podniku to, aby programátor odcházel třeba lebce nespokojen, ale aby to, co odvedl, bylo v pořádku, aby program byl logicky správný, aby se namuselo plýtvat strojovým časem, aby program byl přenosný? Zde jsou tyto zájmy docela jasné a o tom se dá diskuse těžko vést ...

* * *

Nezlobte se na mne. Jestliže zavedu normované programování, tak povýším to naše krásné řemeslo, v němž spatřuji i umění, na takovou kvalifikaci, vlastně na pracoviště, kde se lepí pytlíky a místo lepení pytlíků jsou tam děrovači, kteří lepí programy. Nesmíme se potom divit, že se na tyto funkce navrhují lidi, kteří nemají dostatečnou kvalifikaci a ocitáme se potom v situaci, že musíme propagovat normované programování, protože máme pod sebou nebo vedle sebe lidi, kteří nejsou v stavu nic jiného zvládnout ...

* * *

Myslím, že přirovnání normalizovaného programování k lepení sáčků tak nějak na tu práci nesedí. Když se hovořilo o normalizovaném programování, vkládala se mu všechna "P", všech-

no nejlepší. Ale tá programátorská práce vás ta metoda nezbaví. Toho přemýšlení a těch problémů je i tak dost. Touto technologií se zbavíme jen části problémů ...

* * *

Budou muset mezi nás přijít analytici. Existuje teď už dostatek literatury, která říká, že v agendě hromadných dat je klíčovou a hlavní záležitostí vyřešení souborů. Programy jsou vlastně až druhořadá záležitost. Teprve až mám šikovně navržený soubor, tzn. ze všech hledisek - tedy i z hlediska jeho programového zpracování, tak se píší programy v podstatě samy. Ale jestli je ten soubor špatný, pak bude zpracování trvat dlouho, nedá se na tom realizovat spousta věcí. Podle klasického pojetí by měl soubory navrhovat analytik. Jak to může udělat, když neví, jakým způsobem se potom soubory obhospodařují? U magnetické pásky to je celkem jednoznačné, ale organizaci na disku je možno tak špatně navrhnout, že si potom programátor nad tím bude trhat vlasy a nedokáže z toho udělat rychlé a efektivní zpracování. Analytik bude tedy muset jít mezi programátory. Nenaučí-li se dělat soubory, bude si muset hledat jiné zaměstnání. Neříkám, že musí umět assembler, ale minimálně musí znát Cobol a musí rozumět struktuře operačního systému.

* * *

Na spolupráci analýzy a programování jsou různé názory. Jeden říká, že nejhodnější je, aby analytik a programátor spolupracovali v jednom týmu, v jednom oddělení. Vyskytují se takové případy, že obě profese jsou spojeny v jednom člověku. Dá se to přijmout, když výpočetní středisko začíná, když není dostatek kvalifikovaných lidí. Nedostatky, které z tohoto způsobu práce vyplývají, se obvykle kompenzují nadšením. Druhý způsob organizace práce, který se objevuje, je striktní rozdělení na práce programátorské a analytické. Jsme toho názoru, že jeden člověk nemůže znát vše. Nemůže znát například veškeré mzdové nebo účetnické předpisy, aby si mohl systém navrhnout a k tomu ještě navíc znát programovací jazyk a jazyk operačního systému. Pokud si někdo myslí, že to-

hle vše zvládne, nezvládne to v patřičné kvalitě. V našem závodě jsme prošli oběma organizačními přístupy. První jsme opustili asi po čtyřech letech, když jsme zjistili, že produktivita práce je dost nízká a že je třeba s tou věcí něco udělat. Rozdělili jsme pracovníky na dvě skupiny a ty se věnují pouze svým profesem.

* * *

Jsme středisko, které začínalo asi před deseti lety a od začátku máme oddělené programátory a analytiky. Řekl bych, že to dobře fungovalo právě v té době, kdy tam dělali ti nadšenci. V té době to fungovalo, protože programátor, když dostal podklady, které byly nějakým způsobem nedokonalé nebo v nich něco scházelo, sám si ten problém znova promyslel a dotáhl to až do toho provozovatelného stavu. Tím, že je analytik odtržen od počítače, chybí mu tam taková nějaká zpětná vazba. Ta je zprostředkována přes někoho a analytik má sklony, i když někdy programoval, což ještě není pravidlem, setrvávat na těch metodách, které on používal tehdy, když programoval a těžko se přesvědčuje, aby začal pracovat nějak jinak. Myslím si, že by to bylo ideální, i když jsme to zatím nedokázali, vytvářet ty týmy, o kterých se mluví a které se někde daří dělat a někde ne. V tom týmu by hned od začátku spolupracoval analytik a programátor v tom smyslu, že na začátku by měl převahu analytik a programátor by mu radil, jak to udělat nejvhodněji pro počítač. Ale aby analytik neztrácel styk s počítačem, mohl by v závěrečné fázi programovat některé jednodušší programy, na které by stačil. Chce to, aby se všichni podíleli na výsledku po té programátorské i analytické stránce.

* * *

S rozvojem softwaru pro počítače třetí generace dochází k tomu, že vznikají určité oblasti dříve neznámé. Jde o komunikační programy, databanku, údržbu operačního systému a pod. A tak je těžko možné, aby všichni dělali všechno. Zejména, a naše zkušenost to ukazuje, když průměr pracovníků v této oblasti z hlediska schopností klesá. To neznamená, že speci-

alista dejme tomu přes komunikační systém by se nemohl později stát třeba specialistou přes databanku, ale v dané chvíli málokdo je schopen do dostatečných podrobností zvládnout vše zároveň. Potom myslím, že jsou asi tři oblasti a v jedné z nich se každý z nás cítí nějak doma. Jednak je to analytik systému, který provádí průzkum u uživatelů, zjišťuje tedy algoritmy, obsah dat, které jsou schopni dát, cíle systému a potom zajišťuje zavedení nějakého systému. Jako další je tu navrhovatel systému, který výsledky tohoto průzkumu syntetizuje do nějakého návrhu systému spolu s příslušnou datovou základnou, rozdělením do programů a modulů. Tento už musí mít velice blízko k programování. Musí znát operační systém a je to taková, řekl bych, mezní oblast mezi programováním a tím, co se teď rozumí analýzou. A konečně je zde programátor, který u programů rozdělených do určitých modulů zajišťuje jejich zakódování a otestování. Těžko lze očekávat, že bude někoho, kdo rozumí systému a je schopen dělat tyto náročné práce, těšit takové dohadování do omrzení s uživateli systému o tom, co chtějí a proč se jim to nelíbí, když je to uděláno tak, jak to původně chtěli atd.

* * *

Otázka jazyků. Jsme toho názoru, že je nutno mít jednotný jazyk. Obvykle tam, kde se začíná, se jazyky volí tak, že se seženou programátoři a podle toho, jaký jazyk kdo zná, takový užívá. Nebývá potom řídkým zjevem, že se v jednom podniku programuje ve třech i více jazycích. S přesností programů to pak vypadá velmi špatně. Jeden programátor obvykle nezná více jazyků. A když zná, tak pouze přehledně ... Měli jsme možnost srovnat assembler, Cobol i PL/I a zjistili jsme, že pro takové běžné použití ve výrobě, dá se říci ve velkém, kdy neprostá většina programátorů je na průměrné úrovni, je nejvhodnější Cobol. Nechci nijak napadat jazyk PL/I, je bezesporu tím nejvyšší a nejlepším, co programovací jazyky dávají. Pro běžné použití je však příliš náročný. Je to pro programátora dobrý, ale příliš rozsáhlý nástroj a průměrný programátor je schopen tímto nástrojem, pokud jej nezvládne zcela dokonale, napáchat značné škody.

* * *

Velice rád bych upozornil budoucí uživatele ať již třicítek nebo čtyřicítek, ať velmi bedlivě uváží výběr jazyka. Pokud se ještě nerozhodli nebo pokud dříve programovali v jiných jazycích. Je to otázka velice důležitá a neradil bych, aby ji podceňovali.

* * *

Myslím si, že středisko, které dělá hromadná data, bude těžko dělat ve více jazycích. Myslím, že se dá používat pouze jeden jazyk, u nás je to bohužel PL/I a souhlasím s tím, že ta cesta přes Cobol je šťastnější a my, protože máme s PL/I určité zkušenosti, jsme narazili na to, že je třeba vytypovávat určité postupy, instrukce, deklarace, které není doporučeno používat. Např. není příliš dobré pracovat ve větším rozsahu s bitovými řetězci, protože se výpočty velice prodlužují.

* * *

Jazyk PL/I je zde proto, aby se dal použít pro všechny typy výpočtů a proto je nesmírně komplikovaný. Souhlasím s tím, že v mnoha případech je nevhodný ve srovnání buď s Cobolem nebo Fortranem. Neposkytne větší možnosti a přitom rychlost chodu je několikanásobně pomalejší.

* * *

Chtěl bych říci něco na obranu PL/I. Myslím si, že PL/I je velmi dobrý jazyk, že je lepší než Cobol, ale musí se to s ním umět. Je to vysoký jazyk a svým charakterem velice dobře znázorní to, co se ve stroji děje. Já bych tu PL/I vůbec nezatracoval. V našem podniku Cobol vůbec nemůžeme z principielních důvodů používat, protože vůbec nerespektuje skutečnost, která je někdy podstatná pro možnost dokonalého řešení na místech, kde to dokonalé řešení je potřeba.

* * *

Organizace, které začaly programovat v PL/I, dnes, když mají v chodu 600-700 programů, přicházejí k hledisku provoznímu. A to je nejdůležitější. Při rutinním zpracování se dostáváme do takového stavu, že nejsme vůbec tyto programy schopni v daných terminech zpracovat. Stává se nám, že programy se nemohou sdílet, protože každý z nich potřebuje 100% CPU. Na to

bych chtěl potencionální uživatele systémů EC včas upozornit.

***

Řešili jsme na výrobní poradě problém, že nás programátorů je málo. Přemlouval jsem jednoho zkušeného analytika: "Hele, zamysli se nad tím, nauč se to a pojď dělat k nám." A on mi na to řekl, že je ochoten se to naučit, že to může být zajímavé, ale že programovat nebude, protože se nenechá diskvalifikovat. Bude tedy třeba vymyslet způsob, jak ty lidi mezi nás dostat, aby to necítili jako diskvalifikaci.

***

U poctivého programátora se projevuje jedna zvláštnost. Má k počítači takový vztah, jaký má dobrý žokej ke svému koni. Když si uvědomíte důsledek tohoto tvrzení, tak je to velice závažná věc. A proto nemám rád lidi, kteří sedí v nějaké výborné kanceláři, v životě počítač neviděli a vyrábějí vývojové diagramy.

***

Je řada programátorů, kterým dělá dobře, když mají svůj program nepřenosný. Oni jsou jeho pánem! Jenom oni dokáží do něj udělat opravu. Zvyšuje to možná u nich pocit důležitosti - mít takový program. Myslí si o sobě: "Napsal jsem takový program, že jenom já, já génius, ho dokážu udržet při životě, jenom já ho dokážu opravovat." To je názor velice mylný a vymstí se nejvíce na těch, kteří ho mají. Nejde jen o to být pánem programu, ale často to znamená být otrokem programu. Znamená to být buzen ve tři hodiny v noci, být volán o sobotách a nedělích, v době, kdy to nejméně potřebují. To je jeden aspekt. Druhý aspekt je, že pracuje-li takový člověk v programování určitá léta, táhne už za sebou takový chvost svých "vynikajících" děl, že se určitě k jiné práci nedostane. A takové případy se dají vyloučit nebo aspoň omezit, když se programátor ještě naučí něco málo k tomu moc, co se už musel naučit, když ještě totiž zvládne, jakým způsobem volit identifikátory a jakým způsobem program formálně upravit.

***

Programátor nemůže pracovat jenom těch osm nebo osm a půl hodiny denně v práci. On musí přemýšlet o těch programech i doma večer i o tom víkendu a třeba i o té dovolené.

* * *

Teď jsme v takovém období, kdy dochází k důležité změně. Přecházíme od období, kdy se programovalo systémem nadšenců, kterým se programování líbilo a nacházeli v tom určitou životní náplň. Teď přicházíme do období, kdy většina lidí programuje proto, že je to pro ně zaměstnání. Bylo to již řečeno v diskuzi několika řečníky, souvisí to i s výrokem, že úroveň programátorů klesá - to je nepochybně pravda. Ale bylo tady i několik diskusních příspěvků, kde se vytrvale mluví o tom, že je třeba programovat se stále větším nadšením po večerech, sobotách a pod. Já si myslím, že takhle to už nejde. Když člověk chvíli dělá a přibývají mu do střediska noví a noví lidé, tak vidí, že většina těch přibývajících lidí nejsou nadšenci, kteří touží dělat po nocích, po sobotách, ale jsou to lidé, kteří si chtějí vydělat nějakým způsobem peníze a dělat to co nejméně pracně. Od těchto lidí nemohu očekávat, že budou vymýšlet nejlepší řešení. Tato změna, ke které nyní dochází, má mnoho průvodních jevů. Speciálně se to, myslím, projevuje v tom, že se začíná přecházet na jiné metody práce, že se vynořují metody typu normované programování, strukturované programování, dochází k jiným vztahům mezi programátorem a analytikem.

* * *

Skutečnost, že programu rozumí a může ho udržovat jen jeho tvůrce, musí být vždy považována za hrubý nedostatek řešení.

* * *