# ZKUŠENOSTI S VYČLENĚNOU PROGRAMÁTORSKOU SKUPINOU

RNDr. Jan Žebrok, Ing. Antonín Tomaszek
Třinecké železárny VŘSR, národní podnik Třinec

## 1. Něco historie

Výpočetní technika má v Třineckých železárnách, n.p. již velmi dlouhou tradici. Od prvních zakladačů a strojů, které zvládaly jednoduché operace, jsme se propracovali až k systému 3,5-té generace IBM 370/148. Cesta vedla přes děrnoštítkové stroje GAMMA 10 (z nichž jeden slouží dosud a stále spolehlivě), pak děrnopáskový a magnetopáskový MINSK 22. V roce 1972 byl instalován sovětský počítač MINSK 32 a ukončení jeho činnosti je plánováno koncem roku 1985. Za celou tuto dobu samozřejmě došlo k velkému rozvoji a růstu celého výpočetního střediska včetně změn v organizaci práce. Kdysi vše zvládala skupina pro věc nadšených lidí, kdežto nyní je nutný rozsáhlý štáb pracovníků od vstupní kontroly přes přípravu práce a provoz a samozřejmě tvůrce a realizátory projektů - oddělení analýzy a projektování.

Tento tým byl (jak je to běžně zvykem) rozdělen na několik skupin specializovaně zaměřených lidí z hlediska řešených oblastí. Každá měla své analytiky i programátory, kteří rozvíjeli svou oblast od úvodních analytických prací, programovali a udržovali v rámci svých možností. Neexistovala žádná servisní skupina (systémoví programátoři byli a jsou začleněni v oddělení výpočetní techniky). Platily některé základní normy číslování programů, sestav apod. a pro používání zdrojů (např. počet MGP). S instalací systému IBM 370/148 byla vypracována a zavedena celá řada norem pro zpracování dat, přičemž největší důraz byl v této době kladen na oblast činnosti programátorů. Tato byla dosti výrazným způsobem

ujednocena a formalizována (přes samozřejmý odpor) a nebyla dovolena tzv. "umělecká činnost" při tvorbě programů.

Jako jediný byl povolen jazyk PL/1 a jazyk ASSEMBLER pro systémové služební programy. V souvislosti s pronájmem databankového a datakomunikačního systému byla vyčleněna samostatná servisní skupina s názvem "Správa datové základny", která do vínku dostala současně poradenskou službu pro analytiky a programátory, neboť systém IBM 370/148 samozřejmě dal nové - dosud nepoužívané - možnosti ovšem vyžaduje i nové přístupy k řešení. Zbytek oddělení zůstal organizován postaru. Systém byl úspěšně zaplňován ovšem někdy se stávalo, že analytici ve skupině nedokázali plně zaměstnat své programátory, přičemž v jiné skupině zase nestačili programátoři. Proto bylo po důkladném zvážení a různých poradách a debatách rozhodnuto vytvořit zvláštní programátorskou skupinu, která by zajišťovala servis pro všechny skupiny, vždy tam, kde bude potřeba větší programátorské kapacity. Že však nešlo o proces zcela jednoduchý a časově krátký, chci naznačit v dalším textu.

## 2. Proces konzolidace

Bylo tedy rozhodnuto, že skupina vznikne, ale bylo ji nutno opravdu vytvořit, stanovit náplň její práce a vztahy k ostatním skupinám. Naším záměrem nebylo vytvořit skupinu ze všech programátorů oddělení, ale ponechat analytiky-programátory v jednotlivých aplikačních skupinách, aby spolu s analytiky připravovali, udržovali a rozvíjeli projekty a do oné servisní skupiny zařadit ostatní (tzv. čisté) programátory. Přišla tedy první vážná chvíle, kdy téměř z každé skupiny byl někdo ubrán a to málokdo rád vidí, neboť každý by svou skupinu viděl raději početnější. Navíc každý z těchto lidí nesl na svých bedrech dřívevytvořené nebo právě rozpracované programy. Bylo stanoveno, že všichni musí práce do půl roku ukončit tak, aby je mohli předat programátorům

v aplikačních skupinách, neboť jednou z přijatých zásad bylo, že programátorské skupina bude programy tvořit na základě zadaných projektů a hotové projekty budou udržovat jednotlivé aplikační skupiny. Dá se říci, že až na jednoho pracovníka, který udržoval programy na počítači GAMMA 10, se všichni během stanovené lhůty v maximální míře od starých věcí oprostili a mohli se zapojit do nových úkolů. Další z přijatých zásad byla ta, že výuka programování v PL/1, JCL a znalosti systému bude prováděna výhradně v této skupině a budou ji procházet všichni noví pracovníci po dobu jednoho roku. Ti právě přijímaní tam byli ihned zařazeni a dále několik těch, kteří dosud s PL /1 atd. nepracovali. Jednalo se tedy zpočátku o velmi různorodý kolektiv, kde jen někteří členové měli všechny potřebné znalosti, ostatní si je doplňovali či začínali zcela na zelené louce. Společnou snahou se však podařilo kolektiv stmelit a vykázat také první výsledky. Vedoucí skupiny dostal navíc do náplně práce zodpovědnost za metodiku programování v oddělení a následnou s tím související kontrolu při předávání projektů k provozování a rovněž poradenskou službu pro analytiky při tvorbě projektů. Skupina tedy existovala, měla náplň práce, existovaly i některé zásady pro vazby na ostatní skupiny. Ne všichni byli stále přesvědčeni o přínosu existence této skupiny, neboť zaužívané metody spolupráce analytiků a programátory byly stále dost silné. Náhle bylo nutné vypracovat dosti přesné a podrobné zadání, které bylo podkladem pro práci programátorů na rozdíl od tzv. "práce přes stůl", kdy s novým dnem přicházely nové nápady a programátor stále předělával. Ale systémový přístup k projekční činnosti byl přijat jako potřebný a i tato bariera byla u těch "dosud nepřesvědčených" analytiků odbourána. Tím však nechci říci, že jsou všechny problémy zcela odstraněny a existence programátorské skupiny přináší jen klady. První období konzolidace bylo dlouhé zhruba dva roky a další rok nám přinesl nové poznatky.

## 3. Zkušenosti a náměty do budoucna

Vznik skupiny rozhodně přispěl k tomu, že je možno přelévat jisté programátorské kapacity na větší úkoly, na jejichž realizaci by aplikační skupina v daném čase neměla dostatek kapacit. Skupina má deset členů - z toho 7 žen, a pracuje vždy na dvou až čtyřech větších úkolech a občas řeší operativní záležitosti. Potřebné kapacity jsou požadovány v technickém projektu (hrubém návrhu) a přiřazovány při oponentním řízení. Hrubý návrh je pak rozpracován do detailního zadání programů, přičemž je dohodnuta dělba práce na řešení. Jako nejvýhodnější se ukazuje, že nejlepší je, když práce je udělána tzv. "na klíč". Spolupráce s aplikační skupinou však probíhá po celou dobu, ale její rozsah i úroveň jsou u jednotlivých skupin různé. Programátoři v aplikačních skupinách jsou značně zatíženi údržbou a rozvojem dosud realizovaných projektů a ne všichni mohou mít po celou dobu kontakt s řešiteli. Z toho pak vznikají problémy při předávání hotového projektu do aplikační skupiny. Objevují se bohužel i názory "co jsem neprogramoval, to nebudu udržovat" nebo "co si kdo naprogramoval, ať se také o to stará" apod. Tyto názory bereme jako krátkozraké a výhodné pro ty, kteří toho naprogramují méně. Je však také pravda, že je výhodné něco vytvořit a pak si od toho vyčistit stůl - což je teoreticky případ programátorské skupiny. Je-li však potřeba provést v programu závažnou změnu, tvůrce udržovateli vždy poradí (což je ku prospěchu věci), ovšem pokud by členové programátorské skupiny měli udržovat všechny svou vytvořené programy, došlo by nutně časem k jejich jistému zahlcení a možnost "přelévání kapacit" by se opět ztratila. Navíc by mohlo docházet ke snaze vytvořit raději méně programů, což by značně poškodilo celkovou produktivitu oddělení a to vše by bylo v rozporu s původní koncepcí při vzniku této skupiny. Pokud dochází k dělbě práce mezi programátorskou a aplikační skupinou z hlediska tvorby programů (což je verze nejvýhodnější, ale někdy nutná), je nutné přesně stanovit jednotlivé etapy, aby mohlo efektivně probíhat ladění celého

projektu. Osvědčily se nám krátké pravidelné (co 1 - 2 týdny) schůzky všech řešitelů včetně vedoucích obou skupin, kde je probrán postup prací, řešeny vzniklé větší problémy a určeny zodpovědné osoby za jejich řešení (včetně termínů). Spolupráci s uživatelem zajišťuje výhradně aplikační skupina, aby si uživatelé nezvykali jednat s programátory programátorské skupiny. Úroveň a obsah projektu je tak jednoznačně záležitostí aplikační skupiny a cesta přenášení informací je jediná. Změny vyplývající z požadavků uživatelů i z vlastního řešení (což vždy bylo, je a bude) jsou vzájemně konzultovány a dle nutnosti realizovány. Přitom musí být upravena příslušná projektová dokumentace (originál zadání je v aplikační skupině a kopii mají programátoři). Otázkou zůstává podrobnost zadání projektu až do jednotlivých programů, přičemž by programátor měl dobré, přesné zadání a přitom jistou volnost stavby programů. Jednotlivé fáze projektu máme stanoveny normami, jsou stanoveny formy zadání, ale pro vlastní obsah máme dány jen návody. Postupem času se však výhodné prvky zaužívaly. Používáme rovněž řadu typových prvků do projektů jako jsou konverzní programy, utility a standartní tiskové programy sjednocující, zjednodušující a zrychlující vlastní programování i když někdy i za cenu nárůstu času zpracování. Sledujeme tím standartizaci programátorských prací a jednoduchost restartů při opakování výpočtů. Ladění probíhá vždy za spolupráce obou řešících skupin a je výhodné, když ladící data zajišťuje aplikační skupina, která pomocí nich pak může otestovat jednotlivé programy, celé úlohy a následně celý průběh zpracování. Programátor bývá při tvorbě testovacích dat často "zaujatý" a netestuje dokonale - to neznamená, že při této dělbě práce by neměl testovat vůbec. Objevené nedostatky a nutné změny programátor promítne do programů a tyto může odevzdat až dokonale prověřené. Vítaná je zde maximální spolupráce uživatele, neboť jen ten dokáže vytvořit takovou kombinaci dat (zvláště chybných), která dokonale prověří celé zpracování a jeho zabezpečení proti nenadálým stavům včetně časových relací.

V předávání programů nemáme zatím vypracovanou zcela jednotnou metodiku a tento proces probíhá několika formami. Jsou skupiny, které přeberou programy do své správy po uplynutí ověřovacího chodu (do té doby je za vzniklé chyby a jejich odstranění zodpovědný tvůrce), přičemž příslušný programátor se již dříve seznámil s jejich funkcemi (většinou spolupracoval s analytikem při zadání a následně i při ladění). Také se stává, že skupina formálně práci převezme (hlavně bez zjevného nadšení) a jen se čeká, kdy to upadne, aby mohli rychle zavolat autora a ten to pak opravit. Málokdo v této krizové chvíli dokáže odmítnout pomoc, i když často nejde o programovou chybu, ale o v projektu nepředpokládanou kombinaci dat. Spoléhat se však na tento způsob je do jisté míry pohodlné a ne vždy je autor ihned k dispozici nebo nezná všechny souvislosti. Někteří přebírající programátoři jsou zvyklí na svůj styl práce a jim zase vadí odlišný způsob zápisu instrukcí a nejraději by vše měli po svém. Je nesporné, že při způsobu programování "na zakázku" musí být zajištěna jistá společná struktura programů. Je nepřípustné, aby si každý programátor dělal svou strukturu a odmítl se přizpůsobit. Jak bylo uvedeno, platí pro programování u nás řada norem a máme i model zdrojového textu, což zajišťuje dostatečnou jednotnost tvaru programu. Je však nemyslitelné, aby se všechny programy navzájem podobaly jak vejce vejci a všichni programátoři používali stejné instrukce. Maximální jednotnost ano, ale identita není možná. Vždyť i úroveň jednotlivých programátorů je různá. Vždy a všude však musíme prosazovat zájem o maximální spolupráci při cestě za výsledkem bez ohledu na vlastní výhody.

Hodně jsme si slibovali od jednotné výuky, kterou by procházeli noví pracovníci (i ti, kteří byli plánováni na místa analytiků). Doba jednoho roku se nám zdá dostatečně dlouhá na to, aby každý zvládl základy PL/1, JCL a systému,

zásady práce v oddělení (programátorské) a zásady spolupráce s přípravou práce a provozem. Za tuto dobu se musí rovněž pod vedením zkušeného programátora podílet na realizaci řešeného projektu včetně tvorby příslušné provozní a programátorské dokumentace. Ne všichni však za tuto dobu zvládli všechny potřebné znalosti a některé jim ani nebyly předány dosti podrobně, protože by jich ihned nevyužívali. Každý nový pracovník však nebyl (a nemůže být) přínosem pro programátorskou skupinu, ale po jistou dobu spíše brzdou, neboť se mu musí někdo věnovat a tím se kapacita skupiny snižuje. Na druhé straně to však šetří čas aplikačním skupinám a svým způsobem přispívá k jednotnosti výuky. Navíc takový pracovník nezapadne okamžitě do úzké sféry problémů dané aplikační skupiny, ale může pracovat na jiné problematice a tak si rozšířit svůj obzor a také spolupracovat krátký čas s jinou skupinou. Ne všichni vedoucí skupin však souhlasí s dobou jednoho roku a zvláště v poslední době zase převládají tlaky na co nejdřívější zařazení pracovníka na plánované místo (např. po třech měsících) a okamžité řešení úkolů dané skupiny. S otázkou výuky souvisí i školení ostatních programátorů i analytiků v oddělení, zvláště na poli standartních programů a postupů. Naopak někteří špičkoví programátoři z aplikačních skupin jsou pověřováni výkladem o jimi používaných a ověřených postupech, metodách a možnostech systému. Dlouhou tradici mají tzv. "programátorské čtvrtky", které organizujeme spolu se systémovými programátory. Na nich jsou všichni seznamováni s novinkami zaváděnými ve středisku, s novými standartními moduly, se zásadami tvorby projektů, spolupráce s provozem apod. K nabízeným modulům jsou vydávány "Systémové informátory" s přesným popisem funkce a použití. Často se však setkáváme s tím, že v odborných časopisech či na přednáškách jsou publikovány jako novum takové věci, které u nás musí zvládat běžný programátor či analytik ve své každodenní práci.

Je nutné konstatovat, že za dobu existence programátorské skupiny a hlavně s tím související zákonné povinnosti tvořit podrobnou a propracovanou dokumentaci výrazně stoupla úroveň technických a prováděcích projektů. Téměř nikdo si nedovede představit programování bez předchozí technické dokumentace a "nošení hotového projektu na ruku" bez předání do ověřovacího chodu a rutiny spolu s prováděcím projektem. Pro vychytání nejasností, nepřesností, ale i nátlakových snah a snah o svérazná řešení, či řešení technicky v daných podmínkách a čase nemožná nám slouží oponentní řízení u každého stupně projektové dokumentace. Zde se zkoumá, zda jsou dodržovány platné normy pro tuto dokumentaci i interní normy pro zpracování dat. Probíhá jednání s uživateli, jsou vyčíslovány úspory. Samozřejmě, že stále existují práce, které někdo udělá uživateli v zájmu dobrých vztahů. To někdy může udělat dojem, že nemáme dělat, jindy dojem, že po nás mohou chtít cokoliv bez ohledu na efektivnost. Toto však v celém rozsahu uhlídat lze jen obtížně a souvisí to s disciplínou každého pracovníka.

Co do budoucna? Přát si řadu dobrých projektů, rychlé a spolehlivé výpočetní systémy, kvalitní výuku programování na školách a hlavně příjemné programování.

## 4. Zobecněné závěry (pro a proti)

V mnoha podnicích osciluji organizační formy útvarů realizujících ASŘ mezi dvěma krajními polohami :

a) spojené profese systémového analytika a programátora,
b) rozdělené profese systémové analýzy a programování.

První forma organizace je velmi lákavá, argumenty pro její uplatnění jsou líbivé, ale zrádné. Probereme si postupně tyto argumenty :

1. "Lze eliminovat přesnou definiční fázi programů".
   Toto platí, pokud analytik bude psát všechny programy daného projektu sám. Projekty obvykle obsahují více programů než je schopen zvládnout jeden analytik - programátor.

2. "Lze eliminovat analyticko-programátorský interface".
   Toto tvrzení opět neplatí pro větší projekty.

3. "Eliminace definiční fáze programů účelně redukuje dokumentaci".
   Je to pochybný přínos pro budoucnost, kdy do programu musí často zasahovat někdo jiný než původní autor.

4. "Lze podstatně zkrátit počet člověkoměsíců od úvodní koncepce do implementace".
   Platí opět pro jednoduché programy nebo několik programů.

5. "Zkrácením doby trvání projektu jsou sníženy celkové náklady".
   Vezmeme-li v úvahu skutečnost, že systémový analytik je obvykle lépe placen než průměrný programátor, pak jeho využití (u velkých projektů) jako pouhého programátora je velmi neekonomické.

Z výše uvedené argumentace snad dostatečně vyplývá nutnost specializace systémových analytiků a programátorů. Je třeba ještě dodat, že obě profese mají své odlišné potřeby a zájmy. Je-li systémový analytik zaměřen extrovertně na mezilidské vztahy, řízení a optimální využití všech zdrojů, pak programátor musí být zaměřen introvertně na optimální využití všech zdrojů počítače. U obou profesí je vývoj poznatků a nových metod velmi rychlý, tzn. obrazně řečeno, obě profese musí běžet co nejrychleji ve svém sebevzdělávání, aby se udržely na stejném místě. Je proto zcela na místě rovnoprávnost a rozdělení obou profesí v podnikové praxi. Lze si jen přát, aby rovnoprávnost obou profesí byla vyjádřena i rovnoprávnou úrovní tarifních tříd, což je údajně v některých státech zcela běžná záležitost.