Ing. Richard Bébr

VAKUS Praha

ČLOVĚK ROZMLOUVÁ SE STROJEM

1. Úvod.

Kdekdo dnes píše a mluví o databankách a oslňuje čtenáře či posluchače líčením romantických scén, ve kterých vedoucí pracovník pomocí displeje rozmlouvá s počítačem a hravě tak zvládá všechny problémy. Nesmělé dotazy (na příklad v jakém jazyku bude rozmluva probíhat) jsou odbyty poukazem na přílišný detailismus tazatele. Mezitím odborníci s hrůzou zjišťují, že řada profesionálních programátorů má potíže při domluvě se "svým" počítačem.

Pokusme se tedy uvést různé krajnosti na rozumnou míru.

2. Specialisté a laici.

2.1 Pojmy:

Abychom se dále domluvili, chápejme "rozmluvu neboli komunikaci s počítačem" jako činnost, kdy počítači sdělujeme své požadavky a on je (více či méně správně) plní.

Tuto komunikaci můžeme rozdělit na dvě třídy:
- komunikace na úrovni dat (zadání vstupů a přijetí výstupů)
- komunikace na úrovni algoritmů (zadávání postupu činnosti počítače).

"Specialista" říkejme tomu, jehož povolánim je rozmlouvat s počítači a který by měl mít k této rozmluvě jistý souhrn znalostí a dovedností (na př. programátor). "Laik" pak je ten, kdo příslušné znalosti a dovednosti nemá ("neprogramátor").

2.2 Důvody komunikace s počítačem:

Specialista potřebuje komunikovat s počítačem, protože je to jeho povolání. Programátor předkládá stroji programy a ladí je, operátor se domlouvá se strojem o postupu práce.

A laik?

Moderní počítačové systémy dávají možnost uložení, výběru a vyhodnocení velkého množství informací. Řada laiků něco takového při své práci potřebuje, avšak operativně; komunikace přes specialistu je zdlouhavá.

Výrobci nabízejí terminály a práci v reálném čase. Laici vidí možnost rozmluvy se strojem a tvrdě ji vyžadují, jsouce v tom podporováni reklamou a horlivci, zmíněnými v úvodu. V praxi se pak dožívají nemilých překvapení, o kterých se zmíním později.

2.3 Přístup laiků a specialistů:

Základním problémem, který si laik neuvědomí vůbec a specialista většinou jen v hrubých rysech je fakt, že s počítačem jako takovým nelze rozmlouvat. K uskutečnění komunikace je třeba prostředků - a to prostředků programových.

Pro laika je tato skutečnost zcela nepodstatná. Jeho zajímá výhradně obsah rozmluvy s počítačem z hle-

diska jeho vlastní profese.

Specialista komunikuje se strojem pomocí prostředků, které vytvořili jiní specialisté. Přitom ho zajímá převážně obsahová stránka. Sám tvoří prostředky pro komunikaci laiků nebo jiných specialistů; zde se věnuje hlavně formální stránce.

Obtíž je tedy trojího typu:

- abychom mohli se strojem rozmlouvat, musíme k tomu mít prostředky nejen hardwarové, ale i programové;
- ten, kdo prostředky komunikace tvoří, řeší především jejich formální stránku;
- pouze při tvorbě prostředků pro komunikaci specialistů je profese tvůrce shodná s profesí uživatele.

Z toho celkem logicky plyne:

Prostředky pro komunikaci specialistů jsou dovedeny na velmi slušnou úroveň. O prostředcích pro laiky se stále diskutuje; konkrétní návrhy se vyznačují formální vytříbeností, ale laik-uživatel jimi bývá zklamán.

Další problém spočívá v tom, že převážná většina občanů není žádným způsobem vedena k algoritmickému typu myšlení. Laik bývá schopen více či méně komunikovat se strojem pouze na úrovni dat. Pro rozmluvu na úrovni algoritmů mu chybí základní předpoklady, které musí získávat školením a praxí; pak se ale z laika stává specialista.

2.4 Konflikt:

Pozorný čtenář jistě vytušil snahu o dramatickou gradaci: konflikt spočívá v tom, že laik vlastně nemůže rozmlouvat se strojem tak, jak by chtěl.

Pokusíme se o nějakou katarzi: laik může plně komunikovat s počítačem (i na úrovni algoritmů) při dodržení určitých podmínek, jejichž splnění mohou laici i specialisté více či méně ovlivnit.

Podmínky jsou tyto:

a) výchova laiků k algoritmickému typu myšlení (od mládí!)

b) pomoc specialistů, tvořících prostředky pro rozmluvu laiků s počítači.

Řešení ad "a" je ideální a velmi zajímavé; jeho rozbor však ponechme stranou (přesáhl by rozsah tohoto příspěvku).

Některé poznatky k řešení "b" tvoří jádro této úvahy a vytrvalý čtenář je nalezne v kapitole 5.

Aby však specialisté mohli zdárně pomáhat laikům a tvořit pro ně prostředky rozmluvy se strojem, musí si sami vyjasnit řadu problémů; pokusím se upozornit alespoň na některé z nich.

3. Problémy specialistů.

3.1 Analytik nebo programátor?

Práce s výpočetní technikou je módní povolání. Smůla je v tom, že vyžaduje dosti důkladné speciální vzdělání, znalosti i praxi. Chce-li někdo bez těchto předpokladů vyniknout, hledá skulinu. Zjišťuje: technik musí umět opravit počítač, programátor musí umět programovat. Tyto znalosti lze přezkoušet, praktické výsledky snadno zhodnotit. A co analytik? Zde to není tak jednoduché.

Proto se mezi analytiky dostávají lidé, kteří nemají valné znalosti, zato ale tím více hovoří a sepisují (pokud možno studie o systémové koncepce). Poznáte je okamžitě podle toho, že pohlížejí na programátora jako na nižší druh živočicha. Sami programovat neumějí ("důležité jsou myšlenky, programování je jen rutinní řemeslo"), i když prohlašují, že to nic není. Pokud jsou tvrdě domlouváni ke skutečné analytické práci, stávají se "systémovými inženýry" a

pohlížejí spatra i na analytiky. Existence "pracovníků" tohoto typu bohužel velmi zamlžuje situaci a prohlubuje spory o to, zda spojovat nebo rozdělovat funkce analytiků a programátorů.

Pro počítače III. generace platí: analytik, který neumí dobře programovat, není analytikem.

K tomu nutno ihned poznamenat, že "programováním" mám na mysli práci ve vyšším programovacím jazyku, výbornou znalost datových a paměťových struktur a důkladné zvládnutí příslušného operačního systému.

Navazuji zde na svůj loňský příspěvek /4/ a znovu odkazuji na podnětný článek /2/.

K podpoře svých tvrzení namátkou vybírám několik skutečností:

- analytik by se měl starat o modularitu programového řešení (má to vliv na analýzu i syntézu úlohy); k tomu musí znát možnosti příkazů CALL, COMMON a pod.
- analytik, navrhující postup zpracování potřebuje znát možnosti operačního systému při tvorbě jobů a job-streamů, namluvě už o úlohách s multiprogramovým režimem a úlohách pro real-time
- analytik musí rozumět datovým strukturám, jejich konkrétní interpretaci ve vnitřní i vnější paměti a programátorské práci s nimi, aby neztěžoval programátorům život a efektivně využíval možností moderních počítačů.

Uvedu poněkud obhroublý příklad z vlastní praxe:
Při práci s rozsáhlým souborem dat byl jsem nucen uložit soubor na disk, přebírat postupně jeho části do operační paměti a zpracovávat. Oragnizaci souboru jsem navrhl "analyticky" a při zpracování programu jsem ji musel zcela změnit, aby se souborem šlo v programu slušným způsobem pracovat. Pro zajímavost jsem nechal na počítači běžet obě alternativy; "analytická" měla čtyřnásobnou spotřebu strojového času proti "programátorské".

Raději se ani nešířím o tom, že by analytik měl znát i veškeré knihovní a servisní programy; bereme-li je při ananlýze v úvahu, šetříme práci i peníze.

Závěrem si uvědomíme, že analytik, který neprogramuje, vzdává se dobrovolně komunikace s počítačem. Jak potom může sám prostředky komunikace navrhovat?

3.2 Komunikace programátora s počítačem:

Programátor rozmlouvá s počítačem pomocí programovacího jazyka. Věřím, že jsme se už všichni shodli na tom, že existují dva typy jazyků

- strojově orientované - assemblery
- problémově orientované - vyšší jazyky.

Jaká jsou hlediska volby programovacího jazyka? Zatíženi II. generací hlásají různí teoretici: vyšší jazyk jen tam, kde nezáleží na rychlosti vlastního výpočtu; assembler použijte pro časově náročné opakované partie úlohy a také tam, kde vyšší jazyk svými možnostmi nestačí.

Praxe III. generace však ukázala:

- moderní kompilátory pracují velice efektivně a výsledek překladu je co do rychlosti výkonu více než uspokojivý;
- pokud se někdy vyskytne případ, že vyšší jazyk svými možnostmi nestačí, jde o dva typy problémů:
  - mimořádný případ, kdy se vyplatí upravit analýzu problému tak, aby vyšší jazyk vyhověl (zná-li analytik programování, vůbec k tomuto případu nedochází, neboť již při analýze "myslí v jazyku")
  - opakovaný případ, vyskytující se často se předává k řešení systémovým programátorům (viz též /4/).

Celková rozvaha ekonomiky analýzy, programování a provozu ukazuje, že u počítačů III. generace jsou kriteria volby programovacího jazyka zcela odlišná.

Řídíme se hlavně těmito hledisky:
- rychlost tvorby programu
- rychlost odladění
- přehlednost a čitelnost programu
- snadné pozdější zásahy do programu.

Tato hlediska mluví zcela jasně ve prospěch vyšších jazyků. V této souvislosti odkazuji na /3/,/5/ a případně i na velice zajímavý článek /1/.

Jistě se všichni shodneme na tom, že je potenciální šílenství programovat složitý vědeckotechnický výpočet pro počítač III. generace v assembleru. U hromadných dat to tak zřejmé není.

I když pominemo výše uvedená hlediska, uvědomíme si, že programátor (a zvláště pak dobrý programátor) má v assembleru snahu pomocí různých triků šetřit strojový čas - většinou šetřením instrukcí. Požadavek přehlednosti a čitelnosti sice různé triky přímo zakazuje, přesto se v praxi šetření instrukcí trpí a dokonce opěvuje. Proveďme jednoduchý výpočet pro konkrétní agendu telefonních účtů pro Prahu, která má 250000 položek měsíčně:

Jestliže v programu ušetříme 100 instrukcí na 1 položku, pak roční úspora strojového času činí 10 minut, tedy cca 500,- Kčs (slovy pětset)!

Uvážíme-li, že čas na programování a ladění v assembleru je 8 až 10krát delší než pro týž program v COBOLU, pak "úsporný" program stál jen na základních platech programátorů o 20000,- Kčs více a "návratnost" tohoto vylepšení je 40 let.

Budeme-li programovat ve vyšším jazyku, zbyde nám spousta času na důkladnější analyticko-programátorské řešení na příklad datových struktur. Ušetříme-li ve zmíněné agendě telefonních účtů jeden diskový přístup na jednu položku, znamená to při průměrných cenách roční úsporu strojového času v hodnotě 100000,- Kčs !

3.3 Závěr k problémům specialistů:

Počítač III. generace je pro komunikaci se specialistou vybaven nejen lépe, ale především jinak než jsme bývali zvyklí. Tento poznatek se musí projevit v celém pojetí přístupu specialisty ke stroji. Jinak by jeho rozmluva se strojem připomínala vedení pivních řečí k univerzitnímu profesorovi. Uvědomme si, že stroj je sice shovívavý, klidně počká, než se správně vyjádříme, ale při tomto čekání polyká peníze!

4. Problémy laiků.

4.1 Databanka - móda dneška:

Jak jsem se již zmínil, je všude velký zájem o databanky. Abych se vyhnul různým invektivám, upozorňuji předem, že proti databankám osobně nic nemám; snažím se pouze být realistou.

Úvodem příklad: v ČSSR existuje cca 5 milionů plátců soustředěného inkasa. Bylo by jistě zajímavé, mít možnost libovolného dotazu na nejrůznější údaje a statistiky z tohoto souboru.
Jaké by byla cena této zajímavosti?
Počítač s diskovou pamětí 1500 MBytů (200 diskových jednotek EC) by běžel nejméně 1 směnu denně neustále jenom proto, že někdo někde někdy může přijít s nějakým dotazem. Námitku, že počítač v multiprogramovém režimu může současně dělat i jiné práce připouštím, ale ty jiné práce nesmí mít žádné nároky na disky (a to je u III. generace škoda).

Možná, že je to extrémní příklad; avšak průměrný resort spotřebuje jenom na personální databanku 100 MByte paměti (což je bratru 14 dostupných diskových jednotek!). Nemluvím již o tom, že je v módě též "integrace úloh", takže nestačí přemýšlet na příklad jen o personalistice.

Je nutno vážit cenu dotazu:
jaké budou provozní náklady databankového systému a jaký bude průměrný interval dotazů? Musíme vždy zkoumat, zda se nevyplatí raději zajistit pravidelné zpracování přehledových sestav, kde budou zachyceny odpovědi na různé možné dotazy.

4.2 Co laik chce a nechce:
Snem mnohých laiků je mít na psacím stole terminál a hovořit s počítačem.
Problém je dvojí:
a) o čem vlastně chce laik s počítačem hovořit
b) jak to chce provádět.
Doporučuji všem analytikům položit horlivému zájemci o databanku několik dotazů tohoto typu:
- co potřebuješ od počítače vědět?
- jak často to potřebuješ vědět?
- musíš to vědět okamžitě?
- jak si představuješ formulaci svého dotazu počítači?
Jestliže laik (budoucí uživatel) i specialista (tvůrce) mohou na uvedené otázky odpovědět jednoznačně a jednoduše, je zřízení databanky na místě (vyhoví-li ekonomická kriteria).

Pro potřeby dalších úvah budeme mluvit o dvou typech databank:
- databanky specializované (rezervace letenek, řízení skladů atd.)
- databanky obecné.
V případě specializovaných databank budou většinou odpovědi na uvedené dotazy jednoznačné; proto se doporučuje budováním těchto databank začínat.

4.3 Komunikace laika:

Vrátíme se k jádru problému komunikace s počítačem: jak může laik formulovat dotaz počítači?

Řešení je různé:

A) Laik komunikuje na úrovni dat:
uvádí skutečnost, kterou chce znát, počítač ji vyhledá a předloží. Dotaz je jednoduchý, práce databanky složitá. Není-li v databance příslušná skutečnost zabudována (ať již údajem nebo rutinou, která údaj získá), nelze odpověď dostat.

Proto dobře fungují specializované databanky, kde není mnoho v úvahu připadajících požadavků a nepředpokládá se výskyt dotazů neobvyklých.

B) Laik komunikuje na úrovni algoritmů:
uvede návod, jak má počítač z údajů databáze tísekat výsledek, který chce laik znát. Jsou to známé příklady typu

```
FIND VZDEL IS ING AND VEK GE 30 AND VEK LE 40
```

Ukažte laikovi, že toto je příklad velmi jednoduché formulace pro databanku, začněte mu vysvětlovat pravidla priorit, formátování výstupů atd. a pozorujte jeho reakce. Je to zajímavé a poučné. Většina laiků - mimochodem zcela právem - nechce být programátory a odmítá se učit i jednoduchý jazyk pro styk s počítačem.

Poznámky k řešení tohoto problému jsou v odstavci 2.4, podrobněji se k němu vrátíme v kap. 5. Některé poznatky jsou i v lit. /6/.

4.4 Shrnutí:

Problémy databank:

- ekonomičnost (cena dotazu)
- komunikace laika na úrovni algoritmů.

Řešení může být:

- budování vrcholových databank, evidujících pouze souhrnné a přehledové údaje (viz /6/)
- výchova laiků k algoritmickému myšlení
- převod komunikace na úrovni algoritmů na komunikaci na úrovni dat.

Převod na "datovou komunikaci" (bude popsán v kap. 5) má tento důsledek:

zjednodušení dotazu = zkomplikování databanky.

Buď totiž musíme mít v databance kromě základních dat i data "předzpracovaná" (zvýšené nároky na paměť), nebo musíme mít programově podchyceny různé možnosti vyhledávání a vyhodnocování údajů (zvýšené nároky na programy).

5. Specialista pomáhá laikovi.

5.1 Databankové systémy:

Řada výrobců dodává již hotové databankové systémy. Ve smyslu odst. 4.2 jde o databanky obecné se všemi problémy, které z toho vyplývají. Především je to opět dotazovací jazyk, který bývá velmi dobře vymyšlen, ale vzhledem ke své obecnosti je obtížný ke zvládnutí laikem. Od laika se vyžaduje algoritmické myšlení. Dotazovací jazyk je většinou postaven na bázi angličtiny, takže v našich podmínkách zní dotazy i odpovědi cize a hlavní myšlenka přirozené rozmluvy se strojem je narušena.

Podniká se řada pokusů o tvorbu českého dotazovacího jazyka. Vadí tu háčky a čárky a malá schopnost češtiny zkracovat slova. Někdy se též přehlíží, že databankový programový systém není jen jazyk a jeho in-

terpreter, ale i programy pro tvorbu a údržbu datových struktur a řada dalších softwarových celků (viz /7/). Nelze zakoupit od cizího výrobce databankový systém a předělávat v něm jen jazyk. Většinou se nevyhneme tvorbě celého komplexu programového zabezpečení a to je velmi náročná (a nákladná) práce.

5.2 Komunikace dialogem:

Specializované databanky s úspěchem používají komunikace dialogem. Pracuje se tímto způsobem:

a) Laik vybírá jednu nebo několik možností, nabídnutých počítačem:
počítač vypíše možnosti a uživatel určí číslo té možnosti, kterou si přeje; pokračuje se rozvíjením stromu dotazů.
Příklad (odpovědi uživatele podtrženy) :

```
1. EVIDENCE UKOLU V BEHU
2. EVIDENCE UKOLU SKONCENYCH
3. EVIDENCE PRACOVNIKU
?? 1

UKOLY V BEHU
1. TERMINY
2. FAKTURACE
3. PLNENI
?? 1
```

b) Laik vkládá údaje dle požadavků počítače:
počítač specifikuje jeden nebo několik údajů, které od uživatele potřebuje.
Příklad:

```
PREHLED PLNENI UKOLU

UKOLY OD CISLA 20  DO CISLA 35
```

Zde se na alfanumerických displejích výhodně používá možnost "vyplnění formuláře" : na displeji uvedeme nejprve názvy všech potřebných údajů a cursor programově nastavujeme na místa, kde je třeba doplnit údaj.

c) Laik zadává údaje dle návodu počítače:
počítač poskytne návod a uživatel pak vkládá různé údaje. Příklad:

```
X    - FAKTURACE UKOLU CISLO 'X'
X,Y  - FAKTURACE UKOLU CISLO 'X' S DOBROPISEM 'Y'(KCS)
-X   - PREHLED DOSAVADNI FAKTURACE PRO UKOL 'X'
0    - KONEC PRACE
30
35,5600
-15
0
```

a počítač fakturuje úkol 30, dále úkol 35 s dobropisem 5600,- Kčs a vypíše přehled fakturace úkolu číslo 15.

Dialogový způsob je velmi jednoduchý pro uživatele. Jeho nevýhoda je v poměrné zdlouhavosti dotazu a také v tom, že nelze pokrýt všechny případné požadavky uživatele. Přes tyto nevýhody je dialog využitelný i v obecných databankách.

5.3 Programové řešení dialogu:

Tvůrce programů pro dialog se musí vyrovnat s těmito hlavními problémy:

- pokrytí všech myslitelných dotazů uživatele; chce to hodně přemýšlet a jediná praktická rada zní: nechávejte si rezervy ve stromech dialogu pro budoucí rozšiřování!
- formulace vzkazů počítače; je to jeden z klíčových problémů dialogu. Vzkazy musí být přesné, výstižné,

srozumitelné a stručné. Praktická rada: nestyďte se konzultovat s nejrůznějšími odborníky (třeba i jazykovědci), předkládejte návrhy laikům k posouzení a nechte si poradit!

Další obecné zásady:

Doporučuje se využívat možnosti "HELP", kdy vzkazy počítače jsou velmi stručné. Uživatel má možnost při dialogu požádat o pomoc; na tuto žádost počítač svůj vzkaz rozvede a podá podrobné vysvětlení. Zběhlý uživatel detailní vzkazy nepotřebuje a dialog je urychlen.

Obratným řešením vzkazů můžeme dát uživateli možnost zadávání algoritmů, aniž bychom vyžadovali algoritmický způsob myšlení. Dialogové řešení příkazu

FIND VZDEL IS ING AND VEK GE 30 AND VEK LE 40

(odst. 4.3) může být na příklad toto:

```
KRITERIA VYBERU
1. VZDELANI
2. VEK
3. PLAT
?? 1,2
VZDELANI
1. ING
2. DR
3. -
?? 1
VEK OD  30  DO  40
```

a podobně.

Programově se systém řeší samozřejmě modulárně. Při vzkazech a odpovědích se buduje tabulka postupu práce. Podle této tabulky se pak řídí přivolávání příslušných podprogramů a předávání parametrů.

V závěru malá poznámka: mluví se zde neustále o bankách dat; u čtenáře to může vzbudit představu, že popsané principy rozmluvy se strojem se týkají jen

velkých databankových systémů. Ve skutečnosti můžeme s výhodou aplikovat tyto poznatky i na menší úlohy (zvláště vědeckotechnické výpočty), řešené na minipočítačích nebo terminálech v režimu real-time. Je-li program vědeckotechnického výpočtu dobře navržen, umožňuje aplikace dialogu

- snadnou změnu jakýchkoliv základních údajů
- provádění různých výpočtů v širokém oboru
- zkoumání reakce matematického systému na změny vstupních parametrů.

Příkladem je systém pro výpočty stacionárních spojových družic, navržený a realizovaný ve VAKUS Praha/8/.

6. Závěr - shrnutí.

Počítače III. generace dávají velké možnosti komunikace člověka se strojem. Specialisté generační skok podceňují a návyky,převzatými z II. generace často způsobují neefektivní využívání strojů i svého pracovního času. Laicí možnosti rozmluvy se strojem přeceňují a vyžadují komunikaci, kterou nejsou schopni využívat.

Zlepšením přístupu specialistů a tvorbou prostředků, usnadňujících komunikaci laiků můžeme podstatně zvýšit využití moderní výpočetní techniky.

Literatura:

/1/ Ledgard,Cave: Cobol under control. Communications of ACM,Vol.19,Nmb.11,1976

/2/ Grégrová: Generační přelom ve výpočetní technice. Výběr informací KSnp,1975,č.5

/3/ Čevela: Systémový přístup a logická stavba programu. Sborník-Havířov,DTČVTS Ostrava,1976

/4/ Bébr: Programové řešení typových projektů pro různorodé uživatele. Sborník-Havířov,DTČVTS Ostrava 76

/5/ Volák: Úsporné programování. dtto

/6/ Bébr: Použití minipočítačů pro řízení a vědeckotechnické výpočty. Sborník SIM 76,DTČVTS Praha(v tisku)

/7/ Chvalovský: Banky dat. SNTL-ALFA,1976

/8/ Bébr: Využití počítačů při návrhu družicových spojů. Sborník IV.věd.konf. ČVUT Praha,1977 (v tisku)