U Ž I V A T E L A J E H O S V Ě T

Ing. Richard Bébr - Výzkumný ústav spojů

Příspěvek se zabývá vztahem programátora („řešitele") k uživateli. Snaží se vytypovat nejpalčivější problémy, se kterými se programátor ve vztahu k uživateli setkává. Hlavní myšlenkou je poznatek, že objektem programátorova snažení nemůže být jen fungující program sám o sobě, ale komplexní dílo, určené pro celkové vyřešení uživatelova problému.

1. Uživatel a jiné základní pojmy

Pojem „uživatel" nevznikl ze slova „užívati" (např. léky), ani z výrazu „užívati si" (např. světských radovánek). Uživatel je osoba (fyzická či právnická), která by měla užívat a využívat dobrodiní, poskytovaného výpočetní technikou. Z hlediska programátora je uživatel zavilý tvor, který nedbá ocenit nádheru elegantních programů a neustále narušuje tvůrčí zápal programátora ohavnými přízemními požadavky.

Pojem „ řešitel" byl důmyslnými metodiky vymyšlen proto, aby byla zamaskována nejasnost vztahu „ analytik - programátor ". S ohledem na určení tohoto příspěvku přidržuje se autor v dalším textu označení „ programátor ", které je líbozvučné a kterým lze i dnes uživateli připomínat tajemnost řemesla vyvolených.

Z hlediska uživatele je programátor obávaným nepřítelem, vyzbrojeným nebezpečným arsenálem odborné hantýrky (s jejíž pomocí dokáže zatemnit a přivést do slepé uličky jakoukoliv diskusi); programátor tajuplným způsobem ovládá ponuře vyhlížející skříně, produkující na jeho povel balíky potištěného papíru, předkládané uživateli jako řešení jeho problémů. Dále je programátor maniakem, který s chutí řeší problémy, zcela nezajímající uživatele a naopak odmítá se věnovat tomu, co by uživatel potřeboval.

Vidíme, že vzájemný vztah uživatele a programátora je spíše antagonický (nesmiřitelný, nepřátelský, protichůdný). Ač tím

budou obě strany popuzeny, je nutno prohlásit, že takový vztah působí největší škody a chyby při využívání počítačů.

Nicméně existují i výjimky, kdy uživatel se čirou náhodou (nebo jiným způsobem) shodne s programátorem. Tu pak vznikají díla, která jsou ozdobou dvacátého století a předmětem hrdosti všech, kteří stáli u jejich vzniku a zrodu. Dodejme prozaicky, že taková díla zákonitě vždy přinášejí úspory, pomáhají hospodářství a jsou pro nás všechny užitečná. Stojí tedy za to věnovat určité úsilí na sblížení uživatele s programátorem; následující řádky by měly přinést jisté podněty, směřující k tomuto cíli. Poznejme se navzájem a budeme si lépe rozumět; tato omšelá rada je klíčem ke zlepšení využití výpočetní techniky.

2. Uživatel a výpočetní technika

Abychom nalezli správný vztah k uživateli, musíme si ujasnit jeho poměr k výpočetní technice. Tento poměr hodnotíme za tří hledisek:

A. Emotivní hledisko

určuje „ citový " vztah uživatele k VT

a) uživatel - nepřítel: ze zásady prohlašuje, že výpočetní technika je k ničemu a jen přidělává práci a starosti; nad neúspěchy jásá, úspěchy považuje za dílo náhody.

b) uživatel - neutrál: nestimuluje ani nebrzdí; je mu celkem všechno jedno; nad neúspěchem krčí rameny, úspěchy bere bez komentáře;

c) uživatel - fanoušek: je nadšen výpočetní technikou, chce ji zavádět absolutně všude, jakékoliv výsledky hodnotí nekriticky kladně.

B. Odborné hledisko

zahrnuje znalosti a skušenosti uživatele

a) uživatel - laik: neví o výpočetní technice nic, nikdy s ničím podobným nepracoval;

b) uživatel - operátor: nerozumí podstatě, ale ovládá styk se systémem;

c) uživatel - programátor: někde se naučil vše, co by měl a potřeboval znát řešitel.

C. Provozní hledisko

je dáno zasazením uživatele do systému

a) uživatel - konzument: není zapojen do systému, pouze pasívně odebírá výsledky.

b) uživatel - aktivní prvek: je součástí systému, ovlivňuje a řídí jeho provoz.

Hledisek jsou na sobě nezávislá; čtenář si může ověřit, že kombinováním kategorií z jednotlivých hledisek obdržíme celkem 18 typů uživatelů. S každým typem nutno zacházet individuálně.

Zařazení uživatele do rámce určitého typu je důležité například v různých fásích řešení:

i) Přípravná fáze (formulace úlohy, projednávání koncepce): pozor hlavně na extrémní typy z emotivního hlediska; jak nepřítel, tak i fanoušek bývají nebezpeční. Programátor musí být vždy uměřený, decentní a jistý si svou věcí.

ii) Projektování: zde pozor na odborné hledisko; s laiky musíme hovořit na jejich úrovni, snažit se o vzájemné pochopení. Mnoho škody natropí uživatelé - programátoři, kteří neustále radí a jsou v zajetí představy, že oni by to vyřešili lépe; zde je na místě být skromný, uživateli přikývnout a řešit si úlohu po svém. Někdy však uživatel - odborník přijde s vynikajícím nápadem a ten nesmíme propást!

iii) Zavádění do provozu: směrodatné je provozní hledisko. Pozor

však na uživatele - konzumenty: jsou poměrně pasivní a ovladatelní. Dobrý programátor v tomto případě musí myslet za uživatele a nikdy nesmí konzumentského přístupu zneužít k bezuzdnému řešitelskému ladění.

Uživatel ovšem není většinou jediná fyzická osoba. Přizpůsobíme tedy metody vzájemné komunikace vždy typu jedince, se kterým jednáme.

Typové rozčlenění uživatelů je důležitým prvkem i při výchově uživatele (která je sice nedeklarovanou, ale přesto svatou povinností programátora). Výchovu uživatele provádíme jak při školeních, tak i při vzájemném styku (zde ovšem uživatele vychováváme tak, aby si toho nebyl vědom). Avšak pozor: chcí-li někoho vychovávat, musím být sám dobře vychován!

Typ uživatele hraje velkou roli i při programování úlohy; musíme si přesně ujasnit zvláště otázku, komu jsou určeny výstupy. Markantně se to projevuje při terminálovém provozu, kdy je rozhodující odborné a provozní hledisko. To je vcelku známé a dodržované pravidlo; méně však již respektujeme typ uživatele při návrhu výstupních sestav a to je chyba. O tom se zmíním v kapitole 6. tohoto příspěvku.

Závěrem ještě jedna rada: nesnažme se nikdy uživatele zlomit a násilím z něj udělat jiný typ; důsledky takového počínání jsou nedozírné a způsobí mnoho škod. Uživatele můžeme ovlivnit a pozměnit jen a jen kladnými výsledky programátorské práce!

3. Uživatel a řešitel

Vždy než zasedneme s uživatelem ke kancelářské kávě, znovu si ujasníme jeho typové zařazení. Při jednání pak uživatele respektujeme, nepropadáme zoufalství ani agresivitě.

Jsem si vědom, že nyní každý programátor namítne: proč zrovna řešitelé musí respektovat uživatele a proč by to taky někdy nemohlo být naopak?

K tomu si dovolím uvést tři své (vlastní) názory:

a) Tento příspěvek je určen pro seminář programátorů. Je tudíž nemístné a nelogické radit v něm uživatelům. Možná, že uživatelé mají svá tajná sezení, kde se učí rozlišovat typy programátorů a hledají způsoby, jak s nimi jednat. Na jedné straně jest nám si přáti, aby tomu tak bylo; na druhé straně se pak vžijte do role uživatelů a pokuste se vymyslet, jak byste hodnotili programátory! Tato představa není příliš veselá.

b) Zatím se to veřejně neříká, lec jde o drsnou realitu: jediným smyslem programátorovy práce jsou programy, sloužící přímo či nepřímo uživateli! To zní hezky a jednoduše. Dovedeno do důsledků to však znamená, že my, programátoři, jsme povinni respektovat všechna přání uživatele a vyvinout veškeré úsilí, abychom našli řešení uživatelových problémů. Řekněme si konečně na rovinu, že programátor je v uživatelských službách. Je už konec tajemných sání, kde se v přítmí čs. zářivek pohybovali lehce zarostlí kouzelníci, řešící jakési problémy, prostému člověku zcela nejasné. Je už konec umělců, pro které byl vrcholem program, dělající cokoliv elegantním způsobem. Zbytky těchto černokněžníků živoří mezi systémovými programátory nebo se dali na vědecko-technické výpočty. Ale i tam na ně brzy dojde! (Autor skrsle přiznává, že sám do takového magického spolku patřil - a bylo to velice krásné; byl však zlomen realitou tohoto světa).

c) Jestliže poetické vývody minulých odstavců nedokázaly některé jedince přesvědčit, pak platí ekonomický imperativ: Vážení programátoři - všechny vás živí uživatelé! Nikde na světě neexistuje výpočetní středisko, které by se zabývalo pouze tvorbou krásných programů, nesloužících ničemu. Realita je taková, že každý programátor je živ z peněz uživatele, a je tudíž povinen za tyto peníze poskytnout kvalitní protihodnotu.

Poznámka: s uživatelem agresivním, vyžadujícím evidentní nesmysly,

přerušíme raději veškeré styky (pozor však, aby šlo opravdu o nesmysly). Jsme-li donuceni s ním jednat, pak užíváme pravidla „ všechno písemně a podepsat ". V nejzoufalejších případech začneme důsledně dodržovat platnou metodiku ASŘ; tímto způsobem byli v praxi pacifikováni i nejzatvrzelejší uživatelé.

4. Uživatel a dokumentace

Stavař nejdříve vypracuje plány a potom postaví dům; strojař zkonstruuje automobil, který se pak podle výkresů vyrábí.

Programátor nejprve spáchá systém a pak k němu (zcela otráven) udělá jakousi dokumentaci. To je jeden extrém. Druhou krajnost se pokusili zavést ponuří metodici ASŘ:
nejprve zpracovat co nejpodrobnější dokumentaci a pak podle ní udělat systém; aby byli programátoři úplně zkrocení, určuje se jim obsah dokumentace do nejmenších podrobností. V praxi tak bylo dosaženo rozkošného stavu: projekt, který by měl metodiku přesně respektovat nebude nikdy realizován, neboť veškerá dostupná energie (i čas) řešitelů se spotřebuje na tvorbu dokumentace. O řadě praktických příkladů je autor ochoten podat věrohodné svědectví.

Má-li uživatel a řešitelem společný zájem na realizaci díla, většinou se dohodnou na dokumentaci střídmé, avšak účelné. Vycházejí přitom ze dvou základních účelů dokumentace:

- zachytit co nejstručněji klíčové body řešení (popis datových souborů, popis struktury programového systému atd.), sloužící hlavně pro orientaci řešitele při pozdějších zásazích do systému
- vytvořit pracovní příručku pro uživatele (popis vstupů, výstupů, organizace zpracování atd.), aby byla uživateli umožněna práce se systémem bez neustálých dotazů na programátora.

Při vytváření dokumentace dodržujeme asketickou stručnost. Používáme tabulky, grafy, nákresy a přehledy všude, kde je to možné. Nesmírně se osvědčilo poměrně nenápadné opatření:

programátor musí v úvodu dokumentace zpracovat popis systému v rozsahu nejvýše dvou stránek A4, ve kterém se nesmí vyskytnout žádné odborné pojmy. Vycházíme ze známé skutečnosti, že stručné a jasné vyjadřování nutí člověka k přemýšlení. Navíc má tento popis systému neocenitelný význam např. pro vedoucí pracovníky (kteří nemají čas na luštění zakuklených obsahu tlustopisů), pro různé schvalovací a posuzovací orgány, pro každého, kdo chce rychle vědět, oč vlastně v projektu jde atd. Zákaz odborných pojmů umožňuje skutečně široké využití stručného popisu. Zkuste a budete nadšeni! (Pozor však: zpracovatelé stručného popisu čeká dřina a skřípění zubů - je to daleko obtížnější, než se na první pohled zdá!).

Velmi náročná je dokumentace, určená uživateli; podrobně zvažujeme, kdo s kterou částí bude pracovat a jak. Provozní pracovnice, která musí neustále listovat v 500-stránkové bibli, nám může jednou tuto „ dikumentaci " hodit na hlavu!

Také si uvědomíme rozdíl mezi učebnicí a manuálem (neboť kombinovat tyto dvě formy nelze):

Učebnice je určena pro toho, kdo o systému nic neví; může být pomůckou ke školení nebo sloužit k samostatnému studiu.

Manuál (česky příručka) je používána přímo při každodenní práci pracovníkem, který systém zná (byl např. vyškolen).

V učebnici můžeme tutéž věc probírat na několika místech (např. s různou podrobností), v manuálu musí být vše o jednom pojmu na jednom místě.

Závěrem se ještě zmíním o tom, jak dokumentaci pro uživatele řeší velké firmy, kterým na zákaznícké spokojenosti záleží. Učebnice a manuály zpracovává tým, složený z těchto profesí:

- specialista (odborný obsah)
- linguista (čistota a jednoduchost jazyka)
- pedagog (řazení látky, použití příkladů, ...)
- redaktor (celková úprava, přiblížení čtenáři, ...)
- grafik (obrázky, grafická úprava, zvýrazňování, ...)

s případnou účastí dalších odborníků, např. psychologů. Je to sice dražší, ale v praxi se tento přístup vyplácí.

Poznámka: autor odmítá poslouchat zdůvodnění, proč je obdobný přístup u nás nemožný. Vědecké rozbory „ proč cokoliv nejde " bývají až příliš běžnou náplní debat uživatelů s programátory, všichni je známe a máme jich plné zuby.

5. Uživatel a organizace

Jistá množina šílenců z řad programátorů prosazuje „ přizpůsobení organizačních struktur uživatele výpočetní technice " (tento přístup lze částečně omluvit nekritickým hltáním zahraniční literatury). Nebohý uživatel bývá tímto požadavkem často přiveden do stavu hlubokých depresí. Kdybychom však řešitelského teoretika požádali, aby zodpovědně (toto slovo třikrát podtrhuji) navrhl správnou a výpočetní technice vyhovující organizaci uživatelova podniku (koncernu, resortu), nezbylo by mu než odejít na poušť a stydět se.

V praxi upřednostňujeme raději dvě zásady:

a) Zásada mírných zlepšení - snažíme se s uživatelem dohodnout takové úpravy organizačních struktur, které pomohou oběma stranám, aniž by jim způsobily nepřekonatelné problémy.

b) Zásada zlatých českých rukou - v daném rámci použijeme veškerého důmyslu ke schůdnému řešení úlohy. Skutečný odborník se totiž pozná podle toho, jak umí zadaný problém vyřešit při omezeních, daných existujícími reálnými podmínkami (tuto myšlenku jsem kupodivu zřídka nalézal v zahraniční literatuře; vynikající anglický odborník tam nešetřil chválou na české programátory, kteří právě tyto metody mistrovsky ovládají).

Jsou však i organizační problémy, se kterými by se mělo opravdu něco dělat (v celostátním měřítku). Jedním z nich je známá kumulace termínů: plánovači znají pouze poslední nebo

první den v měsíci pro termíny čehokoliv. Většina výpočetních středisek tudíž v okolí konce měsíce šílí, aby pak po zbytek času takřka neměla co dělat. Cestu z tohoto zmatku neznám; budu vděčen za jakékoliv informace o tom, zda a jak se tento děsivý stav podařilo někde vyřešit.

6. Uživatel a výsledky

Účelnost a efektivnost nasazení výpočetní techniky se nejvýrazněji projevuje na výstupech zpracování a jejich praktickém využití. Znám agendu, jejímž výstupem je každý měsíc menší valník sestav (přesně dle přání zadavatele úlohy). Tyto sestavy jsou dováženy uživateli, který je ukládá do předem určeného skladu. Jiný účel agenda nemá.

Jestliže se nám podaří vytvořit systém, který má opravdu nějaký účel, pak je důležitá úroveň a vzhled výstupů. Výstupní sestavy jsou nejen vizitkou programátora, ale i dokladem jeho vztahu k uživateli. Klasický čaroděj koncipoval sestavu tak, aby každému připomínala záhadnost „ automatických mozků ". Popisy bývaly přebírány z angličtiny a navíc umně zkracovány. Nutno zajisté uznat, že označení TOTL SUM je mnohem efektivnější než prosté CELKEM. Později vznikla snaha o přiblížení sestavy uživateli počešťováním. Zkratky se však tvořily anglickou metodou - vypouštěním samohlásek. Vznikaly tedy rozkošné pojmy jako SCT, PRMRN, PLZK, PRNS, RCNVVJ, PCTPRC, PRDJ, ZVTL. Málokdo z nich ovšem vyluštil významy: součet, průměrně, položka, přenos, roční vývoj, počet pracovníků, prodej, uživatel. O grafické úpravě a přehlednosti sestav raději pomlčme.

Naproti tomu uživatel takřka vždy žádá co největší balíky sestav; programátor kupodivu rád vyhoví a s lehkým srdcem nechá ničit tiskárny (ať technici taky něco dělají).

Lze tedy snadno diagnostikovat: agenda s hrůzným množstvím nepřehledných výstupů vznikla na základě zcela falešného pojetí vztahu uživatel - programátor, kdy oběma stranám bylo umožněno realizovat ty nejhorší nápady.

Správně by mělo dojít ke vzájemné korekci:

programátor navrhuje (a příklady dokládá) efektivní stručné výstupy a uživatel přivádí programátora k lidskému vyjadřování. Chce to ovšem trochu námahy a důmyslu.

Další HRZ (rozuměj „ hrůzy ") přineslo používání terminálů. Uživateli jsou vnucovány důmyslné jazyky pro styk se systémem nebo je nucen k zvládání ďábelských parametrických příkazů (je-li na čtvrtém místě písmeno "A" pak na pátém místě musí být číslice, jejíž význam je dán hodnotou třetího parametru). Programátor v tvůrčím zápalu zapomíná, že ZADANA HODNOTA může znamenat "zadaná", ale i "žádaná", že vzkazy typu
PROSIM ZADEJ JEDNU NEBO NEKOLIK NEBO ZADNOU Z VYSE UVEDENYCH MOZNOSTI KTERE ODDELUJ CARKAMI NEBO STREDNIKY NEBO BLANKY A NAKONEC STISKNI KLAVESU ENTER NEBO JINOU S TYMZ VYZNAMEM

jsou pro uživatele poněkud deprimující, že prokousat se obsáhlým „stromečkem" dialogu je pro uživatele sice jednoduché, ale strašně pracné a zdlouhavé a podobně.

Není smyslem tohoto příspěvku podávat jednoznačné návody. Ty jsou částečně popsány v literatuře a částečně jsou předmětem diskusí (např. na seminářích typu Havířov). Chtěl bych však zdůraznit prostou, leč účinnou zásadu: vzájemné pochopení myšlenek a cílů mezi programátorem a uživatelem vede takřka vždy k dobrému řešení.

7. Uživatel a budoucnost

Ač to není příliš potěšující, budou uživatelé existovat i v dalších letech. Nám programátorům jest se tudíž s touto skutečností vyrovnat. Můžeme zvolit boj nebo vzájemné porozumění.

Historie nás učí, že řešení vzájemných vztahů střetnutím je sice jednoduché, zákonitě však vede k likvidaci jedné ze střetnuvších se stran. Zničení programátorů je patrně nežádoucí; úplné vyhubení uživatelů je tajným přáním takřka všech řešitelů. Stranově obrácený (osově souměrný) přístup můžeme zjistit i u většiny uživatelů.

Ale: programátor nemůže existovat bez uživatele právě tak jako uživatel bez programátora. Nebude tedy řešení sporu bojem příliš efektivní. Jedinou možností je vzájemné porozumění, dohoda a spolupráce.

Praktická potíž bývá v tom, jak konkrétní osoby přimět (dokopat) k vzájemné komunikaci. Kromě námětů, které jsem uvedl v předchozích kapitolách existují ještě obecně platné, byť i poněkud extrémní metody, např.:

a) Narušování vztahů je postihováno stanovenými přísnými sankcemi.

b) Za vzájemné porozumění jsou oběma stranám přislíbeny atraktivní prémie neb jiné požitky.

c) Na programátora budiž vysílána sličná zástupkyně uživatele, případně ješitný uživatel nechť je dán na pospas něžné programátorce.

a podobně.

V praxi se tiše a nenápadně šíří jednoduchá, leč nesmírně účinná metoda:

Ze uživatele i programátora smí spolu jednat pouze lidé (nikoliv věševědi, úředníci, vědátoři, řečníci a jiní nekomunikativní tvorové). Pak lze vždy uplatnit přístup typu
„Jsme přece lidé, kterým o něco jde - tak se snad dohodneme"!

8. Literatura

Monografie o zkoumaném tématu nebyly autorovi dostupné; spousta prací o počítačích se však uvedenými problémy více či méně podrobněji zabývá. Autor tudíž doporučuje (vřele) studium dosavadních sborníků ze semináře "Programování"-Nevířov 1975 až 1981 (i sborníku 1982), zvláště pak těchto vybraných příspěvků:

Ing. Jaroslav Klečka: Vztah dvojice programátor - operátor. (1977)

Ing. Ivan Dzida: Problematika vztahů analýza - programování. (1977)

Ing. Vlastimil Čevela: Problematika komunikace mezi účastníky zpracování hromadných dat. (1978)

Vilém Holáň, p.m., Ing. Petr Miklica: Programátor z pohledu uživatele. (1979)

Dále doporučuje autor své vlastní příspěvky semináře Havířov z let 1976, 1977, 1980 a 1981.

K prohloubení znalostí ve zkoumané oblasti není dále na škodu studium odborných časopisů, denního tisku, děl Dr. Plzáka a Dikobrazu.