# Amatéři, profesionálové a 90. léta

Petr Kubský
*Ústav teorie informace a automatizace ČSAV*
Pod vodárenskou věží 4, 182 08 Praha 8

29. března 1989

Programování dnes není jen povoláním, ale také lidovou zábavou. Miliony nadšenců po celém světě propadly kouzlu s firemní značkou Atari či Commodore, učebnice jazyka Basic se prodávají lépe než detektivky od McBaina a zasvěcenou diskusi o adresování obvodu 8255 můžete vyslechnout v kterékoli žižkovské hospodě. Nám profesionálním programátorům přinášejí nové podmínky novou úlohu: musíme se nějak identifikovat, vymezit něčím svou identitu. Ve svém příspěvku se proto chci zabývat právě rozdíly mezi amatéry a profesionály v programování, jakož i jejich vzájemnými vztahy a souvislostmi[1].

## 1 Who is who?

Vynález knihtisku umožnil svého času laickou vzdělání a vedl k prudkému vzrůstu kulturní a duchovní úrovně relativně širokých vrstev společnosti. Zdá se, že v současné době prožíváme něco podobného: levné mikropočítače způsobily totální demokratizaci oboru, který byl donedávna doménou hrstky vyvolených. To je samo o sobě vynikající a nesmírně cenná věc. Jestliže si dnes desetiletí kluci a děvčata hrají ve školním zájmovém kroužku alespoň s PMD-85, neznamená to samozřejmě, že jsou tím předurčeni pro povolání programátora. Ostatně i ti z nich, kteří se pro něj rozhodnou, se budou v produktivním věku muset úplně všemu učit znovu. Široké veřejnosti — a především dětem a mládeži — by měla práce s počítači dávat něco úplně jiného: měla by v nich uvolňovat a rozvíjet schopnost tvořivé práce v nejširším smyslu slova. V tomto směru má počítač obrovské, zatím zcela nedostatečně využívané možnosti.

Demokratizace přístupu k výpočetní technice staví do úplně nové situace i nás profesionální programátory. Neovolatelně jsme přišli o

pověst zasvěcenců, vstupujících denně ve styk s tajemstvím. Cobolskými programy na evidenci DKP těžko zaimponujeme nezletilým, kteří si přidávají životy ve hře Mad Miner a obratně umístěnými *pouky* vylepšují monitor svého Spectra. Jaký je tedy vztah mezi amatérskými a profesionálními programátory? Co nás spojuje a co odděluje?

Nejprve je třeba dát přesný obsah pojmům amatér a profesionál — a zde určitě nevystačíme s kritériem výplatní pásky. Velké množství cenné programátorské práce odvádějí lidé, kteří jsou de facto placeni za něco úplně jiného. Tento stav je víceméně celosvětový. Částečně je zřejmě způsoben tím, že společenská dělba práce dosud nestihla relativně mladý obor zcela přijmout a vstřebat; kromě toho se však lze domnívat, že k podstatě programování patří neostrá hranice s okolím. Vezměme si jako příklad chemického inženýra, který navrhne a implementuje simulační jazyk pro modelování výrobních linek chemické technologie. Čím je takový člověk ve vztahu k počítačům — amatérem, nebo profesionálem? Má taková otázka vůbec smysl? A pokud ano, na základě čeho ji zodpovíme — budeme si všímat toho, ze které školy má diplom? Nebo spíše, kolik procent své pracovní (popřípadě mimopracovní) doby věnuje práci s počítačem? Nebo rozhodneme podle kvality onoho simulačního jazyka?

Každé z uvedených potenciálních kritérií profesionality lze velmi snadno vyvrátit protipříkladem. Není dost dobře možné úřední cestou stanovit, že programátorem je člověk tehdy a jen tehdy, absolvoval-li informatiku na MFF nebo počítače na FEL (už jen proto, že pak by se ze stavu programátorského musela vyškrtnout celá starší generace včetně většiny pedagogů ze zmíněných kateder). Počet hodin strávených u počítače může vypovídat o čemkoliv od nadšení a zaujetí až po neschopnost vyrovnat se s danými úkoly v daném čase. Ještě nejnadějněji vypadá kritérium kvality. Ale pozor: pokud ho přijmeme, postavíme tím rovnítko mezi pojmy "profesionální — kvalitní, dokonalý, úspěšný" a tudíž také "amatérský — nekvalitní, nepodařený". To není věcně správné, a není to správné ani z hlediska formální logiky: nové pojmy nemá smysl zavádět jen proto, abychom získali synonyma pro pojmy staré.

Zkusme to jinak. Každý, kdo pracuje s počítačem, používá v podstatě dva druhy programů: jednak svoje vlastní, jichž je zpravidla jediným uživatelem, jednak programy převzaté z různých distribučních zdrojů, tedy koupené, vyměněné či kradené. Nejdůležitější programy (operační systém, překladače, nástroje, obecné aplikační programy) spadají vesměs do druhé kategorie. Uvedené rozdělení lze potom použít k poměrně přirozené definici pojmů, jež nás zajímají: amatér píše programy pro sebe, profesionál

pro uživatele. I tato definice je určitě napadnutelná, ale ve srovnání s dříve uvedenými možnostmi vypadá přece jen realističtěji. Zamysleme se nyní [2] nad některými jejími důsledky.

Především je zřejmé, že každé srovnávání amatérského a profesionálního programátora je velmi problematické, neboť ti dva nehrají na stejném hřišti. Profesionál je ozubeným kolečkem — nebo kolem — v lépe či hůře zařízeném mechanismu, produkujícím programy. Je řízen plánem, úkoly a termíny, a na svých prstech, roztančených na klávesnici, vždy cítí — nebo by měl cítit, je-li dobrý a svědomitý — upřené pohledy uživatelů.

Amatér je oproti profesionálovi daleko svobodnější. Především: může si vybrat, zda bude přemýšlet *před programováním*, nebo *až po něm*. Neošetřené chybové stavy, neprověřované vstupní údaje, nepřehledné tisky — nic z toho nemusí amatérovi dělat těžkou hlavu. Je přece jediným uživatelem svého programu. Programování se tím neuvěřitelně usnadní a zrychlí. Vzniklé programy pak ovšem skutečně nemůže používat nikdo kromě autora: za snazší programování se zaplatí pracnější obsluhou programu, což je však pro amatéra většinou dobrý obchod.

Dalším závažným rysem amatérského programování je libovolná redefinice úlohy během jejího řešení. Kdo si dělá zadání sám pro sebe, ten na něm nemusí lpět za každou cenu; vyskytnou-li se při programování potíže, lze zadání pozměnit nebo zúžit. I profesionál je občas zahnán do pasti, z níž není jiného východiska, ale nikdy si nemůže a nesmí z redefinice úlohy udělat základní princip práce.

Typické amatérské programy obvykle neberou žádný ohled na okolí, a to v nejširším smyslu slova. Není třeba, aby ho braly. Není třeba, aby dodržovaly kompatibilitu s čímkoli, není třeba, aby se řídily pravidly přehledného dialogu [3], není třeba, aby si udržovaly vlastní diagnostiku chyb a jako čert kříži se vyhýbaly chybovým hlášením systému. Tomu všemu dává smysl až uživatel programu. Není-li uživatele, není povinných pravidel.

## 2 Dobrý je ten, kdo je dobrý

Produktivita práce programátorů roste [2] rychlostí asi 2% za rok, tedy mnohem pomaleji než výkonnost hardware. Důsledkem je krize software jako setrvalý stav. Je přitom známo, že rozdíly ve výkonnosti programátorů jsou neobyčejně vysoké [4], minimálně 1:20; co jeden dělá měsíc, druhý stihne za den. Proto je neobyčejně důležité hledat a připravovat talentované jedince. Ale co to je talent k programování a jak se pozná? Nebo ještě obecněji:

které vlastnosti dělají dobrého programátora? Podle souhrnného názoru odborníků [2] to je především:

- schopnost přečkat stresové situace
- adaptivnost k rychlým změnám
- smysl pro pořádek
- pokora a skromnost
- sebevědomí a rozhodnost
- kooperativnost
- smysl pro humor.

Pokud jde o znalosti a dovednosti, jako nejpotřebnější se jeví:

- znalost cizích jazyků
- schopnost písemného i ústního vyjadřování
- znalost souvisejících oborů (elektronika, matematika, kybernetika, obor aplikace)
- znalost vlastní teorie programování.

K jednotlivým vlastnostem: o významu přizpůsobení se rychlým změnám a stresovým situacím není sporu. Pro navození stresové situace obvykle bohatě stačí stanovení termínu na úlohu. K tomu přičtěte nepolapitelné chyby, výpadky systému a následným zničením adresáře disku, krize interpersonálních vztahů v týmu, popřípadě i ve vlastní rodině, letní brigády na žních, zimní na úklid sněhu, nedostatek papíru do tiskárny, fixů do plotteru, náhradních dílů do čehokoli a manuálů na všechno — to je nějakých stresů a rychlých změn!

Velice důležitý je smysl pro pořádek. Proti tomu bude asi málokdo protestovat nahlas, neboť navenek chceme všichni vypadat hezky. V duchu si však spousta lidí pomyslí svoje — o konfliktu fantazie a tvůrčího vzletu s úporným pořádkem byrokrata. Jnu, nejde jen o pořádek na psacím stole (i když o ten taky: kolik času týdně ztratíte hledáním založených lejster?), ale především o pořádek a řád ve vlastní hlavě. O kázeň v nejširším slova smyslu. O vůli dotahovat nápady do konce, nemít kolem sebe pořád deset

rozvrtaných a nedokončených projektů. Nezávisí to vždy jen na nás, ale závisí to na nás mnohem častěji, než jsme ochotni přiznat.

Pokud jde o skromnost a sebevědomí, dobrý programátor potřebuje — jako koneckonců každý člověk — dialektickou jednotu obého. Weinberg [2] k tomu podotýká: "Jestliže sebevědomí bez sebekritiky je jako parní kotel bez pojistného ventilu, pak sebekritika bez sebevědomí je jako pojistný ventil bez parního kotle." Což se dá chápat všelijak, mimo jiné i tak, že pojistný ventil bez parního kotle nemůže koneckonců způsobit žádné škody, zatímco věta obrácená neplatí.

Kooperativnost je nesmírně důležitá, ať už pracujeme v týmu nebo samostatně. Přinejmenším vždy píšeme programy pro někoho nebo pro něco: musíme být tedíž kooperativní vůči uživatelům i vůči aplikaci jako takové. Problém je v podstatě týž jako otázka rovnoprávného dialogu v manželství [5]. Nesmíme si neoprávněně přisvojit víc moci, než nám náleží, ale nesmíme se také nechat využívat. Účinná spolupráce je neustálým hledáním a obnovováním dynamické rovnováhy.

A co smysl pro humor? Ten jednak velmi dobře podporuje většinu ostatních vlastností, jednak je přímo nutnou podmínkou každodenní práce s počítačem. Počítače si z nás přece neustále dělají legraci. Vysmívají se našim dobře míněným programům a předkládají nám nejapné hádanky, jejichž uhodnutím podmiňují další spolupráci. Nejsou na to žádné statistiky, ale myslím si, že programátorů bez smyslu pro humor je málo. Buď ho u počítače získají, nebo od takové práce utečou.

U odborných znalostí naprosto nezáleží na jejich rozsahu a hloubce, ale pouze a jedině na schopnosti uplatnit je v praxi — tedy převážně na zkušenosti. Angličtina je programátorovi většinou užitečnější než matematika, schopnost srozumitelného verbálního vyjádření bývá cennější než perfektní znalost šesti programovacích jazyků. Nejrůznější testy, ať už inteligenčního kvocientu, nebo speciální testy pro programátory, mají proto dosti nízkou vypovídací schopnost; dobří jsou prostě ti, kdo jsou dobří.

# 3 Profesionál a profesionalita

Profesionální není automaticky totéž co kvalitní. Mezi oběma pojmy je však přece jen těsný vztah. Ne každý program, napsaný pro uživatele, se skutečně používá; rozhodující je prověrka praxí. Kvalitní profesionální programy jsou ty, které se prosadí na trhu (nebo v jinak organizované nabídce) software. Zkusme se nyní zamyslet nad tím, co charakterizuje profesionalitu

oftware; co by měl umět, co by měl a neměl dělat programátor-profesionál hodný toho označení[1] .

1. Přijmout každý rozumný požadavek. Nehrát si na primadonu, nezneužívat toho, že po každý je schopen naši práci sledovat a kontrolovat. Být dělníkem programování, být solidním řemeslníkem v tom nejlepším smyslu slova.

2. Odmítnout každý nesmyslný požadavek. Jistě, to se snadno řekne. Ale často to jde, často je v našich silách vysvětlit zadavateli nekorektnost jeho požadavků. Problém je v tom, že leckdy dá méně práce naprogramovat a odevzdat hloupost, než objasnit, že to vůbec nemá smysl. Je ... tím, že svému oboru rozumíme, se stáváme spoluodpovědnými za jeho úroveň. Počítače tak jako tak denně chrlí tuny zbytečností; je trestuhodné toto množství vědomě zvyšovat.

3. Pečovat o detaily, neodbývat dokončovací práce. Dobrého řemeslníka poznáte podle toho, že je precizní o poznání víc, než je nutné. To není puntičkářství, to je úcta k materiálu, k dílu i k zákazníkovi; úcta k práci.

4. Nedělat samoúčelnou, byť třeba radostnou práci. To je doplněk předchozího pravidla, jeho protiváhaří, které brání sestupu k nekonečnému pilování a broušení. Profesionál si musí řešení konkrétní úlohy vybírat podle účelu, nikoli podle vlastní záliby. I kdyby úlohy numerické matematiky byly největší vášní našeho programátorského života, musíme podobné požadavky vyřizovat hledáním ve stávajících knihovnách. Psát vlastní programy pro tisíckrát vyřešené úlohy je mírně řečeno nerozumné. Zajdete-li do laboratoře požádat o trochu kuchyňské soli, pak přítomný chemik také pravděpodobně sáhne do skříně s reagenciemi, místo aby smísil ve zkumavce HCl s NaOH — přestože druhý postup by byl určitě působivější.

5. Umět se vcítit do potřeb partnera, ať už je jím uživatel, řešená úloha, počítač nebo kolegové v týmu. R. Bébr [6] v této souvislosti postuluje tzv. *Bébrův zákon:* erotický zdatný programátor je dobrým programátorem, neboť obě činnosti vyžadují přesně vyváženou kombinaci něhy a brutality. Vtip je to dobrý, a dobré vtipy často mívají racionální jádro.

[1] Následující tezi nechtějí být ničím jiným než seriózním příspěvkem do potenciální diskuse.

6. Plánovat neúspěch, počítat s komplikacemi. Optimismus je výborná věc jako životní názor, jako hnací motor, jako zdroj vůle a nadšení; ale v technickém plánování nemá co dělat.

7. Nebýt fanda. Tohle chce pečlivé a opatrné vysvětlení, protože víceméně každý programátor je nadšencem svého oboru. Jde však o míru fandovství a hlavně o jeho zaměření. Nekritické zaujetí velmi rychle přejde v klapky na očích. Připomeňte si jen, jak to vypadá na typickém terminálovém pracovišti pět minut před koncem provozu. Prsty na klávesnicích zrychlují tempo až někam k normě výkonných písařek, oči sledují napjatě obrazovky; tak asi hledí hráči v Monte Carlu na kotouč rulety těsně před zavřením kasina. Kolik chyb se zavleče do programů těmi uspěchanými opravami? To si nikdo nepřipustí; důležité je pouze stihnout ještě jeden běh překladače ... Profesionální programátor by proto měl vědět, že kromě počítačů je na světě ještě mnoho jiných zajímavých a užitečných věcí. Že člověk není periferním zařízením, a pokud to tak někdy vypadá, pak že je cosi v nepořádku. Samozřejmě, kdo se živí tvůrčí prací, ten nemá doopravdy *podle* nikdy — a o to zde také nejde. Jde o to, vidět věci ve správných proporcích, nepodlehnout lokálně omezenému zornému úhlu, nenechat své fandovství přerůst do patologické podoby. Přitažlivost počítačů je pro přemýšlivé lidi obrovská a k podobným extrémům svádí. Pozor na to: počítač je sluha, páni jsme my. Ne opačně.

# 4 Zítřejší starosti

Jak známo, vývoj programového vybavení se neustále opožďuje za vysokým tempem rozvoje hardware. Zatímco se konstruktéři snaží o fyzické ztvárnění svých představ páté generace, programátoři stále ještě řeší problémy povýtce dnešní, občas i včerejší. Od masového nástupu osobních počítačů se však vývoj software přece jen zrychlil. Přibyla totiž přímá komerční motivace, což je odedávna a všude páka z nejsilnějších. Současné období charakterizuje několik významných vývojových trendů, jež se zdají být předzvěstí budoucna. Navzájem se prostupují, ovlivňují, mísí, občas si překážejí — a tak vzniká obrázek aktuálního stavu našeho oboru, obrázek možná trochu chaotický, ale s jasně vystupujícími hlavními konturami.

Prvním z důležitých trendů je směřování k integrovanému prostředí: jednotná podoba rozdílných aplikačních programů, vyřešené vazby mezi nimi, vše v úhledném balíčku s růžovou stužkou a heslem *user friendly*.

Dokud se z produkce software ještě nestal průmysl, tolerovala se značná nejednotnost a roztříštěnost. Průmysl však potřebuje unifikaci, normy a standardy. Programy se proto začínají podřizovat stejným zákonitostem jako šrouby, cihly, panely a konektory. Zavádí se jednotnost vnějších podob programu, unifikace formátů dat. Proto jsou tak oblíbené různé preprocesory a nadstavby; plní v podstatě stejnou funkci jako redukce spojující potrubí o různých průřezech.

Další trend lze označit jako skrývání implementačních detailů. Velmi významná věc, neboť nás zbavuje povinnosti myslet na všechno sami. Ideálem dnešního programovacího stylu je implementace skrytá před uživatelem stejně dobře jako genetický kód před novomanželi o svatební noci. Typickým uživatelem počítače dnes koneckonců není programátor, ale člověk úplně odjinud: konstruktér, písařka, pokladní v bance. Proto vzniká snaha o transparentnost počítače. Koncový uživatel by vůbec neměl počítač vnímat jako počítač, ale jako nástroj aplikace: jako automatizovanou kartotéku, jako šikovný psací stroj, jako dokonalé rýsovací prkno.

Každá opravdu náročná aplikace však dříve či později donutí uživatele programovat; předem připravené prostředky přestanou vyhovovat. Doby hrdinů, ochotných při své profesi ještě studovat strojový kód, však již minuly. Dnešní uživatel ohrnuje nos i nad TURBO PASCALEM 5.0. Má na to ostatně plné právo. Čas člověka je dnes dražší než strojový čas počítače. Když už je tedy uživatel nucen programovat — a to je poslední z trendů — pak rozhodně ne ve FORTRANu či Pascalu, ale v některém z programovacích prostředků velmi vysoké úrovně[2], pro něž je charakteristická velká vyjadřovací síla a možnost formulace způsobem blízkým buď přirozené řeči, nebo nějakému zaužívanému formalismu, s nímž je uživatel dobře obeznámen. Do této skupiny patří třeba SMALLTALK, dotazovací jazyky databází, ale také programy jako MATLAB. Volněji sem lze přiřadit i mnoho jiných prostředků a nástrojů včetně běžných spreadsheetů i třeba DBASE.

Není to tak dávno, co se objevily první zprávy o japonském projektu počítačů páté generace. Tehdy vznikla obava, zda tyto nové stroje nepřipraví rázem všechny programátory o práci. Nyní lze předložit k úvaze jinou verzi: zda nás o chleba nepřipraví — nikoli rázem, ale postupně — právě programovací prostředky velmi vysoké úrovně. Anebo spíš, jak se s jejich rozvojem bude měnit role a zařazení programátora, včetně dalšího vývoje obsahu pojmů amatér a profesionál.

[2]Říkat se jim také jazyky čtvrté generace.

## 5 Pár slov závěrem

Jsme jedné krve, vy i my. Vy, kterým je programování zálibou a vášní, i my, kteří si jím vyděláváme na živobytí. Jsme jedné krve, my, kteří skloňujeme slova COBOL, OS/VS1, agenda a termín, i vy, kteří odebíráte Mikrobázi a stavíte si doma myš. Pokud vám table slova připomenou Knihu džunglí, není to náhodou. Všichni jsme členy určité komunity, programátorské obce. A bylo by si přát více porozumění navzájem. Nejen mezi amatéry a profesionály, ale také třeba mezi teoretiky a praktiky. Mezi lidmi z výpočetních středisek a od mikropočítačů. Mezi systémáky a uživateli. Mezi uživateli TNS a Robotron 1715. Mezi programátory z Prahy, Slušovic i Žiliny. Už v minulém století nás přece učil klásik: cesty mohou být rozličné, jenom vůli mějme všichni rovnou. Jistěže, nemáme to nijak jednoduché. Spoustu času ztratíme řešením různých malicherných problémů, podobni ulítaným hospodyňkám při pátečním velkém úkupu. Nemáme softwarový časopis, máme neskutečně málo diskusních fór jako je toto. Přesto i proto bychom měli držet pohromadě. Máme spoustu problémů ke společnému posouzení, od alespoň minimální unifikace programové základny až k otázkám etiky povolání. Ale to je vesměs už jiná pohádka.

## Literatura


[1] Koubský P.: Základy praktického programování. JZD Agrokombinát Slušovice, v tisku.

[2] Weinberg G.M.: The Psychology of Computer Programming. Van Nostrand Reinhold Comp. New York 1971

[3] Tvrdík J.: Interaktivní programy a pravidla dialogu. Sborník PROGRAMOVÁNÍ'88, Dům techniky ČSVTS Ostrava 1988, str. 106–113

[4] Král J.: Programování v reálném čase. Skripta postgraduálního studia FEL ČVUT, Ediční středisko ČVUT Praha 1988

[5] Plzák M.: Gordické uzly rovnoprávného párového dorozumívání. Mladá fronta Praha 1986

[6] Bébr R.: Na úrovni. Sborník PROGRAMOVÁNÍ'88, Dům techniky ČSVTS Ostrava 1988, str. 53–68