Lidský faktor v procesu tvorby programů

Jan M. Honzík

Předkládaný příspěvek je stručnou rekapitulací stejnojmenné kapitoly známé knihy Iana Sommervilla "Software Engineering", [SOM89], s některými dodatky čerpanými z dalších publikací uvedených literatuře. Jde o poznatky shrnuté z řady výzkumů i zkušeností vyspělých organizací v oblasti psychologie, organizace a řízení práce programátorského týmu.

1. Úvod

Ani ve vyspělých zemích nebyl dlouho člověk a jeho osobnost zahrnut do úvah o metodách vedoucích k co nejefektivnější tvorbě programů. V současné době, kdy u nás vzniká řada nově se formujících firem a podniků, orientujících se na tvorbu programových produktů, a kdy vztah mezi zaměstnavatelem a zaměstnancem, mezi vedoucím a podřízeným dostává nový obsah, kdy ekonomický tlak nepřipustí tak snadno jako dříve zapojení neužitečných pracovníků do procesu, začínají být i u nás znalosti o psychologických, ergonomických, a organizačních aspektech spolupráce v kolektivu mnohem významnější než dříve.

Jako nejvýznamnější důvody studia lidského faktoru v procesu tvorby programů uvádí Sommerville:

- Účinné řízení tvorby programového díla začíná být významné, když rozsah díla je velký a když je třeba aby projekt dodržel smluvené termíny, specifikované vlastnosti a rozpočet. V této situaci musí manažer znát svůj pracovní kolektiv, jak ze stránky jednotlivců, tak ze strany jejich vzájemných vztahů. Pochopení souvislostí psychologie a tvorby programů umožní manažerovi pochopit hranice tvůrčích možností jednotlivců a jejich nejlepší začlenění do tvůrčího týmu.
- Programy jsou používány lidmi, jejichž fyziologické i psychologické možnosti mají svou charakteristiku i omezení. Dobrá znalost a využití těchto možností jsou důležité pro dosažení vlastností umožňujících nejlepšího využití programů.
- Kritickým faktorem ceny software je produktivita programátorů. Pochopením vlivu lidského faktoru na produktivitu a aplikací některých doporučení a návodů se může dosáhnout zvýšení produktivity při nevýznamných nákladech.

Závěry, ke kterým dochází výzkum v této oblasti budí dojem obecně známých zkušeností, ke kterým lze dojít na základě "selského rozumu". Přesto, že tyto dojmy jsou většinou pravdivé, je nutné si uvědomit, že i zkušenosti získané praxí je vhodné

experimentálně prověřit a vytvořit model, který je vysvětluje a který může odvodit i nové užitečné závěry.

2. Některé pojmy psychologie z pohledu programování

2.1 Osobnostní aspekty programátora

V původně zdánlivě mužském povolání je v současné době na celém světě zaměstnáno možná více žen než mužů. Přesto, že se někteří psychologové snažili vytvořit prostřednictvím testování současných programátorů nějaký obecnější osobnostní profil, zatím se nezdá, že by existoval určitý osobnostní typ, který by představoval "typ programátora". Nedostatkem některých studií je skutečnost, že testovaní programátoři byli sledováni z pohledu osobnostních charakteristik bez ohledu na jejich schopnosti a úspěšnost v povolání. Ostatně inteligentní programátor umí většinou vyplnit testy s ohledem na očekávané parametry, což je přirozená lidská asertivní reakce jednotlivce, který se chce prosadit. Lze poznamenat, že taková vlastnost je dokonce velmi vhodná pro úspěšného softwarového inženýra. Nedá se tedy říci, že by pro programování byly předpokladem určité zvláštní osobnostní vlastnosti, snad s výjimkou dvou rysů, o kterých se zmiňují některé zdroje a to: schopnost adaptace a odolnost proti stresovým situacím. První vlastnost souvisí s expansivním rozvojem technologie programování. Neschopnost adaptace, přizpůsobení se stále novým změnám, může vést k morálnímu opotřebení programátorských schopností a celkovému poklesu produktivity jednotlivce. Druhá vlastnost souvisí se situací typickou pro oblast lidské činnosti, u které ještě není dokonale vyvinut systém plánování a měření vykonaného díla. Stres vyvolaný blízkým termínem dokončení projektu, může způsobit ještě větší zpoždění. Odolnost vůči stresu by měla nejen zabránit snižování výkonu v důsledku neurotické reakce psychiky, ale naopak mobilizovat pracovní výkon a soustředěnost na dosažení požadovaného výsledku.

2.2 Mechanismy zpracování vědomostí

Jedním s významných faktorů, ovlivňujících možnosti zpracování nových vědomostí v procesu poznávání, je organizace paměti člověka. Podle výsledků výzkumu se zdá být pravděpodobné, že lidská paměť je organizována velmi podobně, jako některé počítačové systémy. Lidská paměť má hierarchickou strukturu, ve které nalézáme tři výrazné úrovně:

- **Malokapacitní krátkodobá paměť** s velmi rychlým přístupem, ve které se provádí prvotní zpracování informace získané lidskými smysly. Neslouží skutečnému uchovávání informace, ale jen jejímu zpracování. Tato paměť je v mnohém podobná registrům počítače.
- **Pracovní paměť** má větší kapacitu a delší přístupovou dobu. Slouží ke krátkodobému uchování informace a zejména k jejímu zpracování. Nepoužívá se pro dlouhodobé uchování informace. Podobá se operační paměti počítače.
- **Dlouhodobá paměť** má velkou kapacitu, dlouhou vybavovací dobu a nespolehlivý mechanismus vybavování, který může způsobovat zapomenutí uložené informace. Tato paměť se používá pouze pro dlouhodobé pamatování informace a pro účely jejího zpracování se přenáší do pracovní paměti.

Nastane-li expozice problému, přenese se odpovídající informace z krátkodobé paměti do pracovní paměti, kde se zpracuje současně s odpovídající informací přinesenou z dlouhodobé paměti. Výsledek zpracování formuje řešení, které se zanese do dlouhodobé paměti pro budoucí použití. Řešení může být nesprávné, což si vyžádá jeho revizi při některém dalším procesu zpracování. Zdá se, že uložení nového výsledku nedochází k likvidaci starého, chybného výsledku, který slouží jako mechanismus zabraňující provedení téže chyby znovu.

Omezená kapacita krátkodobé paměti má výrazný vliv na možnosti poznávacího procesu. podle některých výzkumů nepřesahuje její kapacita u průměrného jednotlivce asi 7 různých položek (zkuste si zapamatovat sekvenci náhodných čísel, které Vám někdo jiný diktuje). Pojem položka však nemá absolutní informační kapacitu, ale může představovat abstraktní pojem, shrnující charakteristickou informaci mnoha jiných položek. Pojmem "město", který tvoří jednu položku, si mohu zapamatovat mnoho položek typu "dům", představuje-li abstrakce město mnoho domů. Jinou lidskou schopností, která vedle abstrakce kompensuje omezenost lidské krátkodobé paměti je asociace. Existují-li mezi položkami určité vazby, které jsou schopny vzbudit lidskou asociaci, lze si zapamatovat někdy až překvapivě velké množství informace. Právě tato vlastnost umožňuje učení se básní nebo jiných literárních textů.

Přichází-li na vstup větší množství informace, může dojít k situaci, kdy zpracování nestačí vstupu a může dojít ke ztrátě informace.

Model našeho porozumění textu programu v souvislosti s paměťovou strukturou vědomostí může vysvětlit, proč lépe rozumíme programu zapsanému strukturovaným způsobem, proč jsou schopnosti programátora nezávislé na používaném programovacím jazyku a jakým způsobem se nejlépe učí novému programovacímu jazyku.

2.3 Modelování znalostí

Informace, která vstupuje do krátkodobé paměti se před uložením do dlouhodobé paměti zpracovává, a dostává podobu, které se říká znalost. Mezi informací a znalostí není přesně vymezena hranice, přesto lze znalosti z oblasti technologie programování rozdělit do dvou skupin:

- **Sémantické znalosti** jsou znalosti, které které se získávají zkušeností a učením a uchovávají se ve formě nezávislé na representaci. Způsob, kterým je sémantická znalost předkládána tedy neovlivňuje způsob jejího zapamatování.
- **Syntaktické znalosti** představují formální detaily určité representace, popis její formy nebo vyjádření jejího tvaru (např. způsob zápisu přiřazovacího příkazu). Uchování této informace má formu mnohem bližší původnímu tvaru vstupní informace.

Rozdíl mezi sémantickými a syntaktickými znalostmi je velmi patrný, když se již zkušený programátor učí novému programovacímu jazyku. Velmi snadno pochopí principy nového jazyka jako je přiřazení, cyklus, větvení, atd. Tyto principy náleží do jím pamatovaného systému sémantických znalostí. Horší je to se syntaxí, když se konstrukce z Fortranu míchají do prvních programů v Pascalu (např. symbol přiřazení) a konstrukce Pascalu se vnucují při psaní prvních programů Moduly nebo ADY. Tyto problémy sice nemá začátečník. Má ale jiné problémy. Všichni si snad pamatují na počáteční chaos, který zpočátku ovládne snad každého programátorského novice a u slabších povah bývá dokonce příčinou skepse, vedoucí k závěru, že daný jedinec se na dráhu programování nehodí. Příčinou je nejčastěji skutečnost, že adept a možná ani jeho učitel, nedovedl dostatečně oddělit sémantické znalosti od syntaktických. Pak jednoho dne se dostaví okamžik prozření, od kterého je další zvládání jazyka na řádově vyšší úrovni. Je to právě ten okamžik, kdy se podařilo vytvořit hranici mezi sémantickými a syntaktickými znalostmi. Snadnost zvládnutí sémantiky nového jazyka zkušeným programátorem však platí jen tehdy, jde-li o příbuznou sémantiku, např. sémantiku odvozenou od Von Neumannovy koncepce počítače, jako je tomu u jazyků jako je Fortran, Pascal, C, Modula. Učí-li se ale zkušený programátor jazyk s odlišnou koncepcí sémantiky, jako je např. Prolog, může mít díky zatíženosti pamatováním starých znalostí více potíží než začátečník. I při učení se takového jazyka jako je ADA programátorem znalým Pascalu, může dojít k velmi rychlému počátečnímu po-kroku při ovládání principiálně shodných sémantických konstrukcí. Ty ale tvoří menší část jazyka. Ovládnutí ostatních vlastností, jako je využití výjimečných stavů, úloh (tasks) nebo datových abstrakcí (packages) může tvořit podstatnou část výloh spojených s rekvalifikací programátora.

Zdá se, že i sémantické znalosti lze rozdělit do dvou skupin podle způsobů svého uchovávání. Jednu skupinu tvoří znalosti "výpočetních" principů (sémantika nástroje), jako např. princip cyklu nebo mnohačetné alternativy (case) a druhou skupinu tvoří znalosti principu "úkolu" (sémantika problému a jeho algoritmu) např. princip binárního vyhledávání nebo princip průchodu stromem.

Řešení složitých a rozsáhlých programových projektů zahrnuje potřebu integrace "nástrojových" i "problémových" a ostatních složek zpracování znalostí. Jednotliví programátoři však nevynikají vždy v všech těchto složkách. Proto je třeba zvážit předpoklady jednotlivců pro práci v kolektivu. "Uživatelský" typ může ovládat "problémové" aspekty projektu, programátor-návrhář by měl být odborníkem v "počítačových" (nástrojových) znalostech a řešitelský kolektiv musí mít manažera, který dovede myslet v pojmech organizačního charakteru.

3. Řízení kolektivu

Nejtypičtější představou programátora je samotář obklopený hromadami výpisů, zřající vzdálen životu na obrazovku terminálu, nebo je to do počítačů pobláznění mladík, jehož jedinou touhou je napojit se na chráněná data cizího počítače. Je skutečností, že psychologické průzkumy v USA (1978) došly k závěrům, že většina programátorů z oblasti zpracování dat má zanedbatelné potřeby spolupracovat s jinými programátory. Přesto, že řada programátorů pracuje bez spolupracovníků, většina pracuje v tvůrčích kolektivech. Výzkum, který provedla firma IBM v r 1978 ukázal, že 50% pracovního času stráví pracovníci vzájemnou komunikací, 30% samostatnou prací a 20% neproduktivní činností. Porozumění dynamice vztahů v pracovní skupině pomáhá jak manažerovi, tak jednotlivým členům týmu dosáhnout co nejvyšší produktivity.

3.1 Typy osobností v kolektivu

Zkušenosti četných společností ukazují, že vliv složení pracovních kolektivů z pohledu osobnostních typů může mít dramatický vliv na úspěšnost kolektivu. Z pohledu produkce programů lze jednotlivce tvůrčích kolektivů klasifikovat celkově do tří skupin:

- **Typ orientovaný na úkol.** Tento typ je motivován vlastní prací. Zadaný úkol je pro něj intelektuální výzvou, vrcholem, který musí zdolat.
- **Typ orientovaný na sebe.** Motivem k práci takového jednotlivce je osobní úspěch. Vytvoření dobrého programu je jen prostředkem k dosažení osobního cíle. Tito jednotlivci často opouštějí technicko-programátorské problémy, jakmile jim to úspěch dovolí, a snaží se o pozici manažerů a vedoucích.

- **Interakčně orientovaný typ.** Tento typ je motivován zapojením do společnosti spolupracovníků. Až donedávna byl tento typ mezi programátory poměrně řídkým jevem, ale v souvislosti s nezadržitelnou tendencí k tvorbě programů úzce zaměřeného na uživatele je tento typ stále častější.

I když většina jednotlivců nepředstavuje zcela vyhraněně jeden z uvedených typů a i když typovost se může u jednotlivce měnit v čase, lze říci, že v daném čase projevy jednoho typu u daného jednotlivce dominují.

Byla-li pracovní skupina vytvořena výhradně ze členů téhož typu, pak byla skupina úspěšná pouze v případě interakčně orientovaného typu. Jednotlivci jiných typů se ve svých skupinách necítili dobře. Snad pro nadbytek vůdčích typů. U týmů v nichž výrazně převažoval typ orientovaný na problém se pozorovaly častý jev nedodržování standardních rozhraní a jiných smluvených konvencí, nutnost přepracování návrhu poté, co byl dokončen zápis zdrojového tvaru programu, častý výskyt zbytečných nefunkčních ozdob ap. Zkušenosti ukazují, že dobře pracující týmy jsou složeny z jednotlivců všech typů s manažerem, který je typu orientovaného na úkol. Nepodaří-li se vytvořit komplementární strukturu pracovního týmu, je nutné autoritativní vedení, které potlačí tendence k prosazování vlastních řešení.

Zajímavým úkazem je skutečnost, že struktura vytvářeného programového díla má tendenci odrážet strukturu tvůrčího kolektivu. Jestliže tříčlenný kolektiv vytváří kompilátor, lze očekávat, že bude mít tři průchody, z nichž každý je vytvořen jedním členem kolektivu. Je-li tvůrčí kolektiv organizován hierarchicky se silným manažerem, lze očekávat, že vytvářený produkt bude podobně hierarchický, v němž hlavní program vytvořený manažerem bude volat komponenty vytvářené podřízenými. Tento jev se zdá být nevyhnutelný a snaha o vnucení jiné struktury kolektivu je zřídkakdy úspěšná. Aby se minimalizovaly nežádané úkazy, je vhodné zapojit do procesu návrhu projektu všechny členy týmu. Pochopí-li každý z řešitelů koncepci celého projektu i jeho částí, snadněji přijme řešení, které je v zájmu celého projektu. Je pochopitelně obtížné zapojovat všechny členy kolektivu do každé fáze tvorby programového díla. Z toho vyplývá, že velké kolektivy jsou při spolupráci na jednom projektu častěji méně úspěšné, než malé kolektivy, což je dokumentováno řadou skutečných případů, kdy velké projekty nedosáhly ani předpokládaného termínu, ani ceny a nakonec ani všech specifikovaných funkcí. Jedním z řešení je rozdělení projektu na samostatné části, jejichž návrh i vývoj je řešen menšími, samostatnými týmy.

3.2 Programování s potlačením osobnosti jednotlivce

Pojem "programování s potlačením osobnosti jednotlivce" (egoless programming) zavedl Weinberg v [WEI71]. Přesto, že tento fenomén byl zaveden v prostředí

vyspělých západních zemí, kde svoboda jednotlivce, jeho osobní ambice a právo na individuální projevy nemají potlačenou podobu jako v našem sociálním prostředí a přesto, že charakteristika tohoto stylu nám připomíná nepřesvědčivé rysy kolektivizační éry života v socialistické výrobě, je účelné se o něm zmínit. V mikrokolektivu řešitelských týmů může fungovat i to, se se celospolečensky neosvědčilo.

Programování ve stylu potlačení osobnosti jednotlivce je charakteristické přístupem kolektivu k celému programovému projektu jako ke společnému majetku. Zodpovědnost za výsledek sdílí všichni bez ohledu na to, který jednotlivec zodpovídá za celkovou produkci. Tento postoj se nevztahuje jen k programu jako takovému, ale k jeho všem náležitostem, od návrhu, specifikací, programování až po dokumentaci, distribuci a údržbu.

Významným rysem tohoto stylu je, že výskyt chyb v programu se považuje za přirozený jev, a jejich existence se nespojuje s konkrétní osobou a její vinou. Při tomto přístupu je se uplatňuje kolektivní snaha a zodpovědnost, ustavená na neformální bázi, při které programátor, který vytváří program má stejnou zodpovědnost jako ostatní členové týmu zúčastněni na projektu.

Tento styl práce má dobrý vliv nejen na vnější produktivitu týmu, ale i na vzájemnou komunikaci mezi jednotlivými členy kolektivu bez ohledu na jejich postavení, zkušenosti nebo pohlaví. Má však i své negativní rysy, zejména, stává-li se kolektiv příliš uzavřený a do sebe zahleděný. Přesto, že se tomuto stylu programování dostalo značné propagace a dosti doporučení, zdá se, že nejlepší cestou je kombinace tohoto přístupu s více formálními kontrolami a a revizemi, které zajistí, aby se cíle kolektivu nezačaly výrazně odchylovat od cílů organizace, do které skupina náleží.

3.3 Vedení týmu

Vedoucí týmu významným způsobem ovlivňuje úspěšnost produkce. Většina týmů má oficiálního vedoucího, ustaveného vyššími orgány instituce. Ne vždy musí být tato osoba skutečnou autoritou z ohledem na technický charakter řešených problémů. Tato situace však nemusí být kontroverzní. Není nezbytné, aby se administrativní i technické řízení udržovalo v jedněch rukou a vzájemné doplňování "titulárního" - administrativního a přirozeného - technického vedoucího může být často prospěšné.

Skutečným vedoucím kolektivu je osoba, která má největší vliv na ostatní členy. Tento vliv může být výsledkem technických schopností, postavení nebo dominance osobnosti. Vedení se může v různých fázích projektu přenášet na tu osobu, která je pro tuto fázi nejkompetentnější.

Autoritativní vedoucí je vhodný za podmínek, které se vyskytují ve vojenských zařízeních, kde je rychlé a důrazné rozhodování a velení obvyklým řídicím pro-

středkem. V ostatních případech vytváří demokratické způsoby řízení větší pracovní pohodu a větší spontánní účast členů kolektivu na celkovém výsledku. Je žádoucí, aby nadřízené orgány v podniku či instituci dovedly odhalit přirozenou vůdčí osobnost. V technicky rychle se měnícím prostředí, jako je prostředí výpočetní techniky mohou nastat mnohé potíže plynoucí ze skutečnosti, že mnozí mladí a lépe vzdělaní jednotlivci mohou být odborně kompetentnější, než starší, zkušení vedoucí. Technicky zdatný jedinec, který se dostane do vedoucího postavení, by neměl být osvobozen od tvůrčí programátorské práce. Celkovou organi- zaci je třeba uspořádat tak, aby vedle manažerské práce byl vedoucí stále zapojen do konkrétního řešení. Takovou organizaci zavedli ve firmě IBM i v jiných firmách pod názvem tým hlavního programátora (chief programmer team).

3.4 Podřízenost týmu

Přirozená podřízenost týmu, ze které vyplývá spontánní souhlas s pokyny vedoucího a potlačení individuálních představ ve prospěch vedeného týmu, je obecně dobrý jev. Má však i své nevýhody. Je to především odpor týmu ke změně vedoucí osoby a ztráta celkového kritického náhledu na práci týmu, protože kolektivní podřízenost může převládat nad ostatními názory na situaci. Přímým důsledkem může být i "kolektivní názor" (groupthink), při kterém může být žádanou alternativou ta, která je podpořena většinou názorů kolektivu, aniž byla podrobena důkladné analýze.

Vedení by se mělo snažit čelit těmto jevům. Jednou z možností je pořádání organizovaných diskusí, ve kterých jsou jednotliví členové vyzýváni ke kritice a rozboru nejrůznějších rozhodnutí. Těchto diskusí by se měla zúčastnit i přizvaná kompetentní externí osoba-expert, která nezaujatě komentuje vzniklou diskusi. Je účelné dát prostor i osobám, které se jako permanentní šťouralové, nespokojenci a dotazovači jeví někdy i jako osoby nepohodlné. V organizované diskusi mohou působit velmi pozitivně, protože nutí ostatní hledat odpovědi a argumenty vyvracející pochybnosti těchto "čertových advokátů".

3.5 Interakce uvnitř týmu

Vzájemná komunikace mezi jednotlivými členy kolektivu tvoří podstatnou část pracovní aktivity. Může mít podobu pracovních porad, průběžných kontrolních schůzek ap. Komunikace však může být vynucena také špatnou organizací nebo nedostačující dokumentací. Je žádoucí redukovat neproduktivní komunikace na minimum. Vnitrotýmovou komunikaci ovlivňuje několik faktorů:

- velikost týmu,
- struktura týmu,
- postavení a dominance jednotlivých osobností v týmu,
- úroveň technického pracovního prostředí týmu.

Komunikace je obecně dvousměrná. Oba směry komunikace však nemusí být vždy v rovnováze. Tuto skutečnost ovlivňuje postavení, osobnost i pohlaví osob. Je žádoucí, aby silnější partner v komunikační vazbě povzbuzoval slabšího partnera ke komunikaci všude tam, kde jde o produktivní předávání informace.

Z pohledu složení skupiny, dávají ženy i muži přednost smíšenému kolektivu. Z typového pohledu mají ženy větší tendenci k informačně-orientované osobnosti než muži a z toho důvodu mohou často výrazněji ovládat tok informací v týmu. Ve smíšených kolektivech však mohou vzniknout také problémy s ohledem na tradicionalistické hodnocení postavení ženy. Zatím co muži snášejí situaci, kdy se žena transformuje do výrazně informačně-orientované osobnosti, špatně snášejí, když se transformuje do agresivně vůdčí osobnosti. Tento problém, který má obecně kulturně-historické pozadí, musí být citlivě detekován nadřízenými, kteří musí ženě v takové posici poskytnout potřebnou podporu.

Formálně organizovaná komunikace ve skupině může mít hvězdicovou nebo obecně grafovou (síťovou) mnohočetně propojenou strukturu. V hvězdicové struktuře je jeden člen týmu v centrální pozici a rozhoduje o tom, komu všemu předá informaci, kterou obdržel od daného člena týmu. V síťové organizaci cirkulují dokumenty mezi všemi členy kolektivu. Podle některých výzkumů se zdá být druhý způsob účinnější. Pracovníci dávají přednost volněji organizovaným strukturám. Z pohledu sběru a šíření informace je však první systém výhodnější. Nejdůležitějším faktorem je však opět velikost týmu, která ovlivňuje účinnost obou uvedených struktur. Výzkumy jednoznačně prokázaly, že spokojenost a produktivita pracovníků je nepřímo úměrná velikosti týmu. Velikost týmu koreluje pozitivně s výskytem absencí a s fluktuací pracovníků.

3.6 Radosti a starosti práce

Významným faktorem je vztah pracovníka ke své práci. V čem spočívá přitažlivost a v čem jsou nepříznivé stránky tvorbu programování?

- Programování je tvůrčí činnost. Tvoření je božské, a dává pocit velikosti člověku od jeho dětství.
- Programování vytváří užitečné produkty. Pocit vytváření něčeho užitečného, co bude sloužit jiným lidem, je povznášející.
- Složitost rozsáhlého programu dovede upoutat toho, kdo jej ovládá stejně, jako složitý hlavolam.

- Neustále nové problémy a nová technologie nutí programátora k neustálému učení a poznávání něčeho nového.
- Vytváření programů je tvorba velmi s velmi tvárným materiálem, umožňujícím plnit i fantastické představy programových vzdušných zámků. Ztěží se najde tvůrčí činnost s tak málo omezenými prostředky k vyjádření tvůrčí imaginace.

Programování má však i své méně příjemné stránky, které zasahují do čiré radosti z tvůrčí práce:

- Práce na programu musí být dokonalá a bez jediné chyby. Člověk není zvyklý na takovou perfektnost. Tato vlastnost bývá nejobtížnější pro starší pracovníky, kteří si na programování zvykají v rámci rekvalifikace.
- Jiní lidé zadávají úlohy a stanovují požadavky a podmínky. Programátor se snadno dostane do pocitu jen podřízeného postavení, ve kterém má mnohem více zodpovědnosti, než pravomoci a práva rozhodovat.
- Navrhovat velký programový systém je dobrodružství a napínavá činnost. Hledat chyby je otravná práce, která v drtivém počtu případů trvá déle, než se zdá být tvůrci únosné. Hledání chyb, které je samozřejmou součástí programátorské práce postrádá tvůrčí charakter. Doba ladění programu konverguje lineárně tam, kde je očekávaná kvadratická konvergence a poslední chyby se nejčastěji odhalují mnohem obtížněji a déle, než první.
- Je deprimující, když tvůrce zjistí, že program který právě dokončil, nebo který je před dokončením, je již zastaralý, nebo že na trhu se objevil elegantnější a efektivnější program téhož účelu.

4. Pracovní prostředí

Pracovní prostředí je neměřitelný ale velmi důležitý faktor ovlivňující pracovní výkon programátorů. Psychologické výzkumy ukazují že chování jednotlivce je ovlivňováno velikostí pracovní místnosti, nábytkem, vybavením, teplotou, vlhkostí, osvětlením, hlukem a mírou soukromí. Pracovní prostředí ovlivňuje i obecnou náladu kolektivu. Nespokojenost vede k častější fluktuaci a k nákladům spojeným se zaškolením nových pracovníků. Poměrně málo pozornosti se věnuje návrhu budov a pracoven, určených pro programátory. Velmi často jsou pracovníci umístěni ve velkých místnostech a jen vedoucí pracovníci mají vlastní kanceláře. Studie, které provedla firma IBM ukazují, že takové uspořádání se pracovníkům nelíbí a nepřináší ani vyšší produktivitu. Podle těchto studií patří mezi nejdůležitější faktory prostředí:

- soukromí,
- osvětlení vnějším denním světlem a možnost výhledu do okolí,

- možnost přizpůsobit pracoviště osobnímu vkusu pracovníků.

Výzkum ukázal, že výkon pracovníků, kteří původně pracovali ve velké společné místnosti, se výrazně diferencoval a obecně zvýšil. Kromě soukromí pracovišť se požaduje prostor pro formální i neformální komunikaci. Jeho vhodné vybavení, podporující neformální sdružování a komunikaci, odstraní nevýhodu snížení komunikace v důsledku individuálních kanceláří, do kterých ostýchavý pracovník jen ztěží přijde za zkušeným kolegou. Uvádí se anekdotický příběh, kdy odstranění kávovaru za účelem snížení ztrát času vzájemným vykládáním vedl k extrémnímu zvýšení požadavků na formální porady a konzultace. V řadě pracovišť se organizují pravidelné "kávové" přestávky, které se považují za povinné, a které spojují psychickou relaxaci a osvěžení s neformální výměnou informací.

5. Plánování a kontrola tvorby programu

Pro naprostou většinu rozsáhlých programových projektů je typické, že se nedaří dodržet ani termín, ani rozpočet a často ani plný rozsah projektovaných funkcí. Tyto jevy nejsou v jiné materiální činnosti ve vyspělých zemích běžné. Proč se nedaří programovací činnost plánovat tak dobře jako jinou konstrukční činnost v oblasti materiální výroby i složitých produktů? Zajímavé postřehy v tomto směru uvádí [BRO52].

Jako motto jedné kapitoly uvedl Brooks citát jednoho jídelního lístku francouzské restaurace: "Příprava dobrého jídla vyžaduje čas. Jste-li nuceni čekat, je to jen proto, že Vás chceme obsloužit co nejlépe".

Metody odhadování potřebného času a nákladů, které jsou nedílnou součástí technologie programování, nejsou zatím dostatečně vyvinuté. Velmi často jsou postaveny na zcela mylném předpokladu, že vše půjde hladce.

Řada metod odhadu chybuje v tom, že zaměňuje úsilí se skutečným pokrokem v práci, čímž se zatemňuje skutečnost, že lidé a roky (či původně lidé a měsíce, protože ve vyspělých zemích se nepočítá na člověko-roky, ale člověko-měsíce, angl. man-months) se nedají zaměňovat.

Protože panuje nejistota o odhadech doby potřebné k vytvoření programu, vedoucí manažeři nemají trpělivost, o kterou prosil šéf kuchyně výše zmíněné francouzské restaurace.

Měření a kontrola množství provedené práce na vytvářeném programu je velmi nespolehlivé, protože měřicí metody nejsou stále dostatečně vyvinuté a i ty co existují se používají velmi málo a to přesto, že v jiných oblastech činnosti jsou samozřejmostí.

Zjistí-li se skluz termínu, pak nejčastější reakcí manažerů je sebevražedný akt přidání dalších pracovníků, který špatnou situaci ještě výrazně zhorší. Tento zásah vychází z podvědomého, ale zcela mylného přesvědčení, že práce, která je odhado-

vána na 12 člověko-měsíců proto, že by ji měl jeden programátor udělat za 1 rok, je možné udělat s 12 programátory za měsíc. Tato trojčlenka v plánování potřebného času je prokazatelně nebezpečným mýthem.

Plánování jednotlivých fází tvorby programového díla je v důsledku notorického optimismu většinou nepřesné a potřebná doba je podhodnocená. Nejhůře naplánovanou fází bývá ale testování a ověřování programu. Podle zkušeností z velkých projektů ve firmě IBM dochází Brooks k tomuto pravidlu:

1/3 plánování a návrh
1/6 implementace (kódování)
1/4 testování komponent a předběžný systémový test
1/4 úplný systémový test

Toto pravidlo se liší od běžných přístupů především v tom, že:

- část pro plánování a projekci je delší; stejně většinou nestačí pro vytvoření dokonalých specifikací, nemluvě o důkladném studiu a výzkumu zcela nových metod a postupů,
- polovina času je věnována testování, což je více, než většina manažerů připouští přesto, že skutečnost nakonec vždy takovou potřebu potvrdí,
- části tvorby programu, kterou lze odhadnou snad nejpřesněji se věnuje pouhá 1/6 celkové doby.

Špatný odhad doby pro testování mívá závažné následky. Jak narůstá zkluz v dokončování programu a blíží se očekávaný termín jeho odevzdávání, obavy z nedodrženého termínu zatlačují obavy z nedokonalosti odevzdávaného produktu. Většinou si málokdo uvědomuje, že náklady spojené s nápravou chyb a důsledků předčasně předaného produktu jsou větší, než náklady, které by se musely vydat na dokončení úplného testování i za cenu nedodržení smluveného termínu.

6. Závěr

Tento krátký příspěvek nemůže než naznačit šíři problematiky spojené s plánováním a řízením tvorby rozsáhlých programů. Podrobnější rozbor ukazuje, že problémy s velkými programy mají jiný charakter než problémy se středními a malými programy. Všechny však vzbuzují nutkavé přesvědčení, že v oblasti tvorby programů vznikají nejsložitější produkty lidské činnosti vůbec. To už je dostatečně závažný důvod k tomu, aby se této části disciplíny programování a hledání a vytváření odpovídajících nástrojů a metod věnovala větší odpovídající pozornost, protože podle starého přísloví "dobrý řemeslník se pozná podle nástrojů, se kterými pracuje".

7. Literatura

[BRO82] Brooks,F.P.Jr.:The Mythical Man-Month, Essays on Software Engineering, Addison-Wesley, Reading, Massachusetts, 1982

[HON80] Honzík,J.M.: Nové směry v programovacích technikách, Habilitační práce FE VUT v Brně, 1980

[HON91] Honzík,J.M.:Technologie programování, skripta FE VUT v Brně, 1991 (v tisku)

[SOM89] Sommerville,I.: Software Engineering, Addison-Wesley, Reading,Massachusetts, 1989

[WEI71] Weinberg,G.M.: The psychology of Computer programming Van Nostrand Reinhold, New York, 1971

Autor: Doc. Ing. Jan M. H o n z í k, CSc.
Katedra informatiky a výp. techniky, FE VUT,
Božetěchova 2,
612 66 Brno 12
tel. 05 - 746 111/kl. 31