P o č í t a č k o m u n i k u j e s č l o v ě k e m

Ing Jaroslav Klečka

Usnadnění práce s počítačem spočívá především v tom, jak lehce či obtížně, lépe řečeno srozumitelně bude moci tento počítač komunikovat s člověkem - jeho uživatelem. Dobře formulované zprávy, vycházející z počítače, pomáhají člověku při jeho práci s ním. Naopak nevhodné zprávy práci uživatele s počítačem znesnadňují.

Jedním z důvodů, proč naše programy, zabezpečující dialog člověka s počítačem, vydávají nevhodně znějící zprávy, může být i to, že tyto zprávy jsou povinně krátké a my jako autoři programu nepředpokládáme, že jim jejich příjemce bude rozumět, aniž se bude muset na vysvětlení chybějícího významu podívat do provozního manuálu.

Člověk - uživatel tedy potřebuje od počítače zprávy, které se vysvětlují samy a každá z nich má svůj vlastní význam. Zprává, jejíž význam se musí vysvětlovat, nemůže sloužit ke komunikaci - nenese dostatečné množství informace.

Ze zkušenosti víme, jak se většina z nás pokouší vytvářet "dobré" zprávy. Obvykle k tomu dochází až poté, kdy máme svůj program hotov a teprve potom hledáme někoho, kdo by nám zprávy vydávané z našeho programu přestylizoval, protože teprve až po odladění programu vidíme, že tyto zprávy by měly vypadat úplně jinak. Obvykle se také pouze snažíme, aby všechny zprávy byly stručné, gramaticky správné, srozumitelné a předávaly požadovanou informaci. Tyto kvality nám však ještě nezajišťují, že výsledné zprávy budou také dostatečně závažné, vhodné svým významem, vydávané ve správném čase a budou uživateli pomáhat řešit jeho požadavky. Na tomto místě můžeme zodpovědně konstatovat, že druhou kategorii kvalit nemůžeme do již existujících zpráv nikdy později vložit.

Pokusme se tedy shrnout činnosti, které je nutno udělat, abychom vytvářeli zprávy, které budou vždy obsahovat obě výše uvedené kategorie kvalit.

1. Zpráva je určena člověku - uživateli.

Člověk - uživatel chce, aby mu počítač pomáhal a ne aby tomu bylo naopak. Jaké zprávy tedy musí konkrétní program vydávat? Můžeme doporučit, aby tato otázka byla zodpovězena ještě před tím, než onen konrétní program začneme tvořit . Musíme se rozhodnout, jaké zprávy mají být uživateli předávány, jaké informace budou obsahovat a jak je budeme uživateli předkládat. Abychom vytvořili program, jehož zprávy budou dostatečně závažné, budou odpovídat požadavkům uživatele, budou vydávány včas a současně pomohou uživateli řešit jeho konkrétní situaci, musíme dodržovat následující zásady :

1.1 buďme tolerantní k uživatelovým chybám.

Program musí vždy reagovat na data vkládaná uživatelem do počítače tak, aby vstupující data splňovala jednu z následujích podmínek:

- byla použitelná okamžitě,
- byla použitelná po provedení jednoduché opravy, přičemž program vkládaná data opraví, volitelně o své činnosti informuje uživatele a teprve pak pokračuje ve zpracování opravených dat,
- byla použitelná po složitější opravě, kdy náš program musí tato data opravit ve spolupráci s uživatelem, přičemž po provedení opravy musí uživatele požádat o pokyn k dalšímu pokračování nebo vkládání dalších dat,
- v případě, že jsou nepoužitelná, zajistit, že nebude možno ve zpracování pokračovat. To se stává tehdy, jestliže jsou vstupující data neinteligentní anebo zjevně špatná. Program pak musí uživatele požádat o zásah, přičemž mu musí sdělit, proč nemůže ve zpracování pokračovat.

1.2 pomáhejme uživateli opravit jeho chyby tak, jak to nejlépe dokážeme.

Ideální by bylo, kdyby náš program byl schopen uživateli pomoci opravit jeho chybu vynaložením menšího úsilí, než které potřeboval pro vložení původních chybných dat. Jestliže nám program nebude moci takto pracovat, měl by uživateli alespoň pomoci se ze vzniklé situace dostat pryč.

1.3 dejme uživateli možnost řídit si předkládání zpráv z počítače

Od interaktivního programu lidé vyžadují v různém čase různé množství informací. Tak např. poskytování informací o počátečních hodnotách, zadávaných programem při jeho zahájení anebo o opravách, prováděných tímto programem na vkládaných datech je velice výhodné, když se jej uživatel učí obsluhovat. Ztrácí však zcela svůj význam při opakovaném použití programu, kdy již uživatel jeho obvyklou činnost zná. Pak již vlastně nepotřebuje některé zprávy znovu číst. Můžeme tedy vytvořit program tak, aby předkládání zpráv šlo zadáním parametrů potlačit. Přitom však vzniká nebezpečí, že některou důležitou zprávu ztratíme. Program potom musí uživatele alespoň varovat, že vznikla nevyklá situace, jejíž vysvětlení nebude předloženo.

Předkládáme-li uživateli na vědomí dlouhé zprávy, můžeme je v souvislosti s případnou omezenou rychlostí použitého terminálu dělit na menší části. Tyto části pak uživateli předkládáme postupně a povolíme mu, aby si toto předkládání volil sám.

1.4 nedělejme zprávy povinně krátké.

Ačkoliv termín "stručný a krátký" ve skutečnosti znamená "ne delší než je nutno", chápeme jej často jako "ne delší než nějaká předepsaná délka". Jestliže "řešíme" problém sdělení "informace" zprávou o povinné délce, dostaneme se do situace, že začneme tvořit nic neříkající zprávy. Jednou z možností omezení délky zpráv je stanovení jejich proměnlivé délky.

1.5 předvídejme, které zprávy bude uživatel potřebovat.

Vytváříme-li programy, které budou spolupracovat s člověkem, musíme nejen definovat, co se stane, když nevzniknou žádné chyby (ani v programu ani u vkládaných dat), ale současně musíme předvídat všechny snadno vznikající chyby a také plánovat, jak budeme zacházet s chybami neočekávanými. U posledně jmenovaných je nutno si vytvořit strategii předkládání informací o vzniklé situaci, při níž musí program vydávat různé zprávy pro :

- uživatele,
- pracovníky, odpovědné za zpracování programu,
- programátory, odpovědné za diagnozu chyb.

2. Používejme při psaní zpráv psychologii.

I když zajistíme, aby odpovídající zpráva byla předána odpovídající osobě včas, můžeme v této zprávě mít z různých důvodů ještě tyto další závady :

- její vyjádření není zcela srozumitelné,
- něco jí chybí,
- obsahuje příliš mnoho detailů,
- pohled programu není shodný s pohledem uživatele,
- předkládaná fakta jsou nejasná.

Abychom se uvedeným závadám vyhnuli, je užitečné vědět co možná nejvíce o lidech, kteří budou naše programy používat. Se značnou pravděpodobností nebudou s námi programátory sdílet náš zájem o výpočetní techniku. Jejich slovník může být chudší, ale i když bude stejně bohatý, ani pak nemusí znát, co je to třeba "byte".

Naši uživatelé budou většinou myslet a cítit takto :

2.1 jejich očekávání určuje způsob reakce za různých okolností.

Očekávání je produktem dřívějších zkušeností a budoucích potřeb každého člověka. Zpráva, která je výsledkem lidské činnosti, musí také této činnosti odpovídat. Zprávy, které berou ohled na uživatelovo očekávání pak musí :

- mít význam, odpovídající situaci, v níž se uživatel nachází. Není-li tomu tak, musí být zpráva postavena tak, aby z ní bylo poznat, komu je určena.
- obsahovat informace, odpovídající potřebám uživatele, kterému je zpráva předávána. Nelze-li to zabezpečit, určeme alespoň, co od uživatele požadujeme a nepoužívejme nedůležitých informací.
- používat vhodnou slovní zásobu pro předkládání informací. Slova a způsob jejich spojování musí být uživateli srozumitelný. Jestliže tomu tak není, musíme změnit svůj slovník a zprávu přepsat.

2.2 uživatel se pokouší spojovat si sám jednotlivé dílčí informace.

Každý člověk se snaží spojovat si jednotlivé dílčí informace dohromady tak, aby získal celkový názor na situaci. Jak mu v tom můžeme pomoci :

- postupujme od obecného ke konkrétnímu,
tak např. zpráva

ABCOOOI/seznam parametrů/

našemu uživateli mnoho neříká a většina lidí nebude bez doplňujícího vysvětlení vědět, co s takovouto zprávou udělat. Přepíšeme-li však zprávu alespoň takto:

ABCOOOI DATA NEMOHLA BYT CTENA /seznam parametrů/

řekneme toho uživateli mnohem více a bude pak schopen rozhodnout o svém dalším postupu,

- zacházejme s podobnými myšlenkami podobným způsobem. Informace předkládané uživateli určitým způsobem vyvolávají u něj očekávání, že mu budou stejným způsobem předkládány i při všech dalších příležitostech. Jestliže mu např. chybové zprávy nejprve vysvětlí jeho chybu a pak mu navrhnou způsob jejího odstranění, bude očekávat stejné pořadí i u všech dalších chybových zpráv, které mu předložíme později. Zpráva s opačným pořadím informací uživatele zaskočí.

- signalizujme konec jakékoliv akce. Každý uživatel má rád, když jej ubezpečíme, že byl počítačem vyslyšen a jak byla jeho činnost přijata. Vždyť vkládání vstupních dat není izolovanou činností, ale je to pokus o komunikaci a ta není ukončena do té doby, dokud se uživatel nedozví, že zpracování jím vkládaných údajů nebylo ukončeno. Informujme jej tedy, jaký byl výsledek jakékoliv jeho činnosti.

2.3. žádný člověk nečte rád jakoukoliv zprávu vícekrát.

Při čtení čehokoliv dává člověk jednotlivým slovům, frázím a odstavcům jejich význam. Stejně tak bychom měli při tvorbě zpráv určit jednoznačný význam informacím, které pomocí zpráv předkládáme uživatelům.

Někdy se nám to však nepodaří. Použijeme-li kupříkladu více slov, než bylo nutno a přitom jejich vzájemné spojení může i zastírat námi zamýšlený obsah, přestává nám uživatel rozumět a žádá, aby mu zpráva byla zopakována, případně si ji musí celou nebo její část přečíst znovu.

- píšme zprávy pomocí termínů, s nimiž je uživatel obeznámen,
- poskytujme uživateli konkrétní a přesné údaje,
- poukazujme na vzájemné vztahy mezi předkládanými informacemi.

Jestliže se budou uživatelé - čtenáři našich zpráv - cítit bezpečně, protože našim srozumitelným a konkrétním informacím rozumí a přitom jsou schopni si spojovat vzájemné vztahy jednotlivých dílčích údajů, nebudou celou výslednou zprávu muset opakovaně číst a bude se jim s počítačem daleko lépe pracovat.

2.4. uživatel má rád pohodlí.

Každý program musí o své činnosti poskytovat vhodné informace. Každý uživatel chce v každém okamžiku vědět, co se právě v počítači děje a co od něj může v příštím okamžiku očekávat. To je důležité zejména tehdy, když musí na výsledky zpracování delší dobu čekat. Výhodné je rovněž, když uživateli ponecháme možnost řídit další průběh zpracování, dojde-li k jeho pozastavení.

Špatné je, když si uživatele znepřátelíme kupříkladu tím, že jej budeme informovat o nejasných skutečnostech nejasnými zprávami anebo jej budeme vůbec ignorovat. Bude si pak myslet, že našim informacím se nedá rozumět a přestane počítači věřit. Nevhodné je rovněž jakékoliv manipulování s uživatelem, neposkytnutí jiných alternativ dalšího postupu, použití slangu anebo zlehčování uživatelových znalostí. Vždyť použití fráze : "ZAPOMNEL JSTE...." zní sice úderně, ale co si uživatel bude myslet, když jej o tom budeme informovat pětkrát za minutu.

Doporučujeme proto, aby naše programy byly tolerantní k lidským chybám. Bude-li náš program sám opravovat uživatelovy chyby, naučí se jej uživatel daleko rychleji obsluhovat.

3. Píšme zprávy pro odpovídající kategorie pracovníků a situace, v nichž se právě nachází.

Kategorizace zpráv pro různé skupiny pracovníků, kteří s počítačem pracují, má své opodstatnění. Určuje totiž lidi, kterým budou různé zprávy posílány. Program pak musí generovat více různých zpráv, určených operátorům, programátorům a konečně i vlastním uživatelům.

Plánujeme-li ve svém programu zápis jakékoliv zprávy, musíme nejprve analyzovat potřebu přesných a odpovídajících informací, které budeme určitému člověku za dané situace posílat. Musíme vědět, co daná informace tomuto člověku řekne a jaké části celkové skutečnosti musí znát. Pro každou část informace pak také musíme přesně identifikovat její zdroj i příjemce.

Přitom je samozřejmě nutno zjistit, zda daný člověk danou část informace potřebuje, zda nepozná potřebnou informaci z kontextu anebo zda informace nemusí být člověku předkládána explicitně, případně zda nejde odvodit z některé dílčí zprávy.

Musíme také vzít v úvahu skutečnost, že ne každý člověk je schopen odvození informace z kontextu zprávy. Při psaní zpráv si musíme být vědomi rozdílů mezi jednotlivými kategoriemi pracovníků i mezi jednotlivými pracovníky uvnitř těchto kategorií. Vždyť se postupně budeme setkávat s lidmi novými a nezkušenými na jedné straně a na druhé straně s pracovníky zkušenými a samostatně uvažujícími, znalými výpočetní techniky. Je proto vhodné, když do zpráv na začátku vložíme více informací, než se nám zdá být nutné, protože nadbytečná sdělení se ze zpráv daleko snadněji vylučují, než se do nich později vkládají.

Pokusme se nyní definovat typy zpráv, které by mohli různí lidé potřebovat :

3.1 hlášení o reakci programu na vstupující data.

Každý program by měl přijmout jakákoliv vstupní data, zadávaná člověkem. člověk pak musí obdržet jednu z následujících odpovědí:

a) program data přijal
- zpracování proběhne až do konce rychle - informujme uživatele o dokončení zpracování, respektive o možnosti vkládat další data,
- zpracování bude trvat delší dobu - pak je vhodné o tom uživatele uvědomit a současně mu dát vědět, jak se zpracování vyvíjí, protože jinak začne mačkat různá tlačítka terminálu nebo telefonovat na všechny strany.

b) program data nepřijal
- data nejsou správná - pak je nutno povolit jejich opravu,
- data jsou sice správná, ale přesto nejsou z nějakého důvodu přijata. Potom pokud to je alespoň trochu možné, řekněme uži-

vateli raději co jde udělat (čekat, telefonovat každých 30 min., volat systémové programátory a pod.), než že něco udělat nejde (DATA NEPRIJATA a pod.).

3.2. hlášení o převzetí vstupních dat.

Při práci s některými programy může mít uživatel mlhavou představu o tom, co se stane v daném okamžiku. Nedovede si však představit, co se s jeho daty bude dít později.

Budeme-li předpokládat, že člověk nemusí třeba souhlasit s počátečními podmínkami, zadávanými počítačem, musíme jej o zadání těchto podmínek informovat, dát mu možnost je změnit a teprve pak vydat povel pro další pokračování ve zpracování. Je-li počátečních podmínek více, nesmíme uživatele nutit opakovat i ty podmínky, které nemění, ale umožníme mu jen jím požadovanou změnu.

3.3. hlášení o chybách programu nebo nepříznivých podmínkách pro jeho zpracování.

Chyby programu anebo nepříznivé podmínky pro jeho zpracování nás nutí vydávat takové zprávy, které budou potřebné nejen pro uživatele samotného, ale současně pro každého, kdo bude tento stav analyzovat. Potřeby rozdílných kategorií pracovníků jsou velice rozdílné :

- uživatelům nemusíme za této situace říkat, že došlo k chybě programu nebo že program nelze v tomto okamžiku zpracovat, ale musíme jim sdělit, jak se z této nepříznivé situace mají dostat, co mohou spravit a pod. Např. :

  BEZPECNOSTNI KOPIE NEPROVEDENA PRO ZAVAZNOU CHYBU PROGRAMU.
  OPAKUJTE ZPRACOVANI AZ PO NOVEM ODSTARTOVANI PROGRAMU.

- kontrolní pracovníci, zadávající a kontrolující zpracování pro uživatele musí znát celý problém a musí vědět, jak rychle musí tento problém vyřešit. Musí vědět, jak mají zrestaurovat potřebné zdroje co nejrychleji, aby zpracování mělo odpovídající podmínky, jako třeba :

  SPOLECNA PRACOVNI OBLAST JE PLNA, PREPLNENY JSOU I UZIVATELOVY

PRACOVNI OBLASTI. ZPRACUJTE PODLE PROVOZNIHO MANUALU
PRACOVNI POSTUP XXXXXXX.

- konečně programátoři musí znát přesné detaily o chodu programu, zejména pak v případě, že tento nedopadl dobře. Aby se zorientovali v činnosti programu před jeho abnormálním koncem, musí v něm mít zabudovány vnitřní indikátory, umožňující sledovat jeho chod. Zpráva pak může znít např. takto :

VNITRNI CHYBA FUNKCE XXXXXXXX V MODULU YYYYYYYY.

Takováto informace prakticky nic neříká předchozím dvěma kategoriím pracovníků. Při tvorbě programu je velice obtížné stanovit místa, kam zprávy posledního typu vkládat, aby nám pomohly orientovat se v programu po jeho abnormálním ukončení. Zprávy budou samozřejmě do určité míry zpomalovat chod programu a neměly by být tištěny, je-li průběh zpracování normální. Můžeme konstatovat, že tuto složitou situaci částečně řeší moderní operační systémy.

3.4. požadavky na pokračování ve zpracování.

Program by měl uživatele požádat o pokyn pro další pokračování, jestliže :

- činnost, která má být provedena, bude mít pravděpodobně výsledky, které nemohou být opravovány odděleně anebo jejich oprava je proveditelná jen obtížně,
- počáteční podmínky, zadávané programem anebo následující oprava chyb bude konfliktní s uživatelovými předpoklady,
- vedlejší účinky zpracování nejsou tím, co uživatel od zpracování očekával.

Ve všech těchto případech musíme uživateli dát možnost odpovědět programu "NE, NEPOKRACUJ !". Současně se však musí dozvědět:

- co již bylo provedeno,
- s jakou další činností má vyslovit souhlas,
- jaké budou následky, odpoví-li "ANO" a jaké budou, řekne-li "NE",
- jaký význam tomuto "ANO" nebo "NE" přiřkne.

Tak třeba ve zprávě :

POZADAVEK NA CTENI JE PRECHODNE POZASTAVEN, POZADOVANA VETA JE JIZ ZPRACOVAVANA JINYM UZIVATELEM A MOHLA BY BYT PRAVE AKTUALIZOVANA. CHCETE PRESTO TUTO VETU CIST?

jsme se dopustili několika nepřesností, z nichž vzniknou následující problémy :

- zpráva neidentifikuje konkrétní větu, přičemž tato identifikace může být podkladem k rozhodnutí,
- zpráva neobsahuje ani požadavek na vyřčení "NE", nechceme-li pak větu číst. Požadavek na okamžité čtení věty může znamenat, že věta nebyla druhým uživatelem ještě aktualizována a příjemce zprávy tedy obdrží nesprávnou verzi. Je také nejasné, zda byl program pozastaven tak, že odpovíme-li "NE", že bude čekat do té doby, dokud druhý uživatel větu nezaktualizuje. Ze zprávy také nelze poznat, jak dlouho má příjemce zprávy čekat. Co se stane, když program nebude čekat vůbec?
- zpráva také nic neříká o tom, jak bychom na ni měli odpovědět.

3.5 požadavky na výběr z různých možností.

Nabídka různých alternativ dává uživateli možnost výběru. Program pak analyzuje situaci a směruje člověka na své dílčí činnosti. Program může uživateli poskytnout možnost výběru mezi alternativními činnostmi nebo volbami, které řídí další požadavky na zpracování anebo mu nabídne alternativní postupy oprav chybných vstupních dat či alternativy k nejasně interpretovaným informacím. Uživatel pak musí také mít možnost zamítnout všechny nabízené alternativy ve prospěch alternativy vlastní, jíž dává přednost. Ve všech těchto situacích musí uživatel vědět :

- co se bude dít a co je již hotovo,
- mezi kterými alternativami si může vybírat,
- proč je mu dávána možnost výběru,
- jaké budou následky volby jednotlivých alternativ,
- jaké důsledky bude mít odmítnutí všech nabídek,
- jakým způsobem si má uživatel zvolenou alternativu vybrat.

3.6 požadavky na chybějící informace.

Program nemusí za všech okolností dostatečně porozumět pokynu pro

další pokračování. Jestliže potom žádá doplňující informace, musí se uživatel dozvědět :

- co se bude dít a co je již hotovo,
- o jakou doplňující informaci jej program žádá,
- co se stane, když požadavku programu nevyhoví,
- jak má svou odpověď realizovat.

Protože uživatel našeho programu nemusí být za všech okolností schopen programu požadované informace dát, musíme mu pomoci a umožnit mu doplňující informace zadat. Výsledkem může sice být použití vlastních, programem automaticky zadávaných hodnot, ale i ukončení programu. Uživatel může také program požádat o návrh dalšího postupu.

3.7 Požadavky na opravu nebo upřesnění vstupujících dat.

Skutečné chybové zprávy by měly být vydávány pouze jako poslední východisko z nouze. Jestliže je program schopen si chyby opravit nebo požádat o vyjasnění vstupujících dat, není nutno vydávat žádnou chybovou zprávu. Program může totiž uživateli nabídnout několik různých variant pokračování nebo provedení opravy vkládaných dat či jejich vyjasnění.

Někdy se však přece jenom stane, že program není schopen dostatečně porozumět významu vstupujících dat tak, aby je při tom mohl interpretovat. Nemůže se také dostatečně srozumitelně vyjádřit o svých potížích s takto vkládanými daty. Jedinou možností pak bývá zkoumání syntaxe vstupní informace, případně jiných konvencí jejího zadání. Programátoři však obvykle tuto poslední možnost ve svých programech nerespektují. A přitom termín "syntaxe" a jeho použití je jejich denním chlebem a součástí jejich profesního světa.

Naproti tomu neprogramátoři nebývají s nutností používat syntakticky správné vyjádření obeznámeni, i když se zcela neuvědomněle se syntaxí setkávají trvale ve svém rodném jazyce. Je proto nutné, abychom svoje zprávy tvořili tak, že v nich bude spíše záležet na významu, než na jejich syntaktickém vyjádření. Je rovněž možno doporučit, abychom celkově vydávali chybových zpráv co nejméně a namísto toho tvořili spíše zprávy, které uživatele žádají o doplňující informace anebo mu nabízejí několik alternativ pro další pokračování.

Nemáme-li však jinou možnost a program musí uživatele požádat o opravu nebo vyjasnění vkládaných dat, musíme uživateli dát vědět :

- co se bude dít a co je již hotovo,
- co je na vstupních datech špatného nebo nejasného,
- co se stane, když data nebudou opravena nebo vyjasněna,
- jak má být oprava nebo vyjasnění dat provedeno.

Za ideálních okolností by také program měl oznámit a popsat jen jednu chybu, týkající se jedné skutečnosti. Není-li však program schopen rozpoznat jen jednu chybu, musíme vydat zprávu tak, aby identifikovala všechny dílčí chyby a vydala návrh na opatření, potřebného k celkové opravě.

4. Ověřme si použitelnost zpráv ještě před jejich zakódováním.

Poté co jsme si stanovili všechny cíle a kdy jsou již všechny zprávy napsány, jsme připraveni ověřit si jejich vhodnost. To musíme udělat ještě před jejich začleněním do zdrojového textu programu. Měli bychom vyhledat všechna slabá místa svých zpráv a přitom vzít v úvahu, že ověřování zpráv na použitelnost není totéž, jako jejich testování na správnou programovou funkci.

Dříve než začneme program kódovat, musíme se na prvním místě zajímat o to :

- Které zprávy jsou připraveny k vysílání?
- Jsou tyto zprávy adekvátní potřebám uživatele?
- Nebude uživatel potřebovat ještě nějaké další informace?
- Podávají všechny zprávy správné informace?
- Jsou při tom všechny zprávy dostatečně jednoznačné?

Abychom si ověřili všechny výše uvedené skutečnosti, můžeme postupovat jako při divadelní zkoušce a při prvním čtení hry z listu. Použijeme dvou "herců", z nichž jeden "hraje" program a komunikuje s uživatelem - druhým "hercem". Tato metoda je podstatně levnější, než budování prototypu programu a může být dokonce i efektivnější. Tato cesta se nám může zdát kuriozní, protože nejsme na něco podobného zvyklí. Daleko častěji přeci ladíme činnost programu, než testujeme činnost jeho modelu. Přitom je daleko horší objevit nevhodné postupy uvnitř pracujícího programu, než je objevit na jeho modelu.

5. Upravujme zprávy do vhodného jazyka.

Závěrečná úprava či redakce zpráv není totéž, jako jejich testování na použitelnost ačkoliv citlivý a důrazný "redaktor", který je při tom také uživatelem počítače, uvidí při své práci mnoho cest ke zlepšení zpráv co do jejich funkčnosti a snadnějšího použití celého programu. Náš "redaktor" však musí znát všechny souvislosti použitých zpráv. Přitom nám ale vždy může pomoci v následujících bodech:

- jsou zprávy napsány dobře?
- je použitá terminologie vhodná pro předpokládaného příjemce, který bude s programem komunikovat?
- je použito věcného konverzačního tónu, bez zbytečných emocí?
- odpovídají si všechny zprávy navzájem?
- je použito běžných interpunkčních znamének?
  (nedoporučuje se používat středník a pomlčka)

6. Vytvářejme tedy programy, které komunikují.

Jestliže tedy známe své cíle a máme připraveny hotové zprávy, které má program posílat uživateli, nejsme již daleko od vlastního zápisu do zdrojového textu programu. Přesto si však ještě musíme položit některé otázky technického charakteru, které musíme zodpovědět dříve než začneme program kódovat :

- jak budeme předkládat více úrovní detailů?
- jak necháme uživatele žádat o více informací?
- jak uživateli umožníme opravit vadný vstup bez toho, aby musel celou informaci opakovat?
- ke kterým různým určením budeme vztahovat vysílané zprávy a jak toto přiřazení budeme identifikovat?
- které části programu budou zprávy vysílat a zda toho také tyto části programu budou vědět dost o uživateli, s nímž mají komunikovat?
- jak budeme tvořit zprávy, které obsahují proměnlivé informace?
- jak budeme sledovat, zda jsme dosáhli svých cílů?

7. Ladíme zprávy společně s běžícím programem.

Testování pracujícího programu na jeho použitelnost je rozdílné od jeho ladění na správnou funkčnost. Při ladění funkčnosti testujeme vlastní kód programu. Testování jeho použitelnosti má jiné cíle a musíme podle toho také postupovat.

Použijme k tomu skutečného uživatele a dejme mu k dispozici program, jako nástroj jeho komunikace s počítačem. Teprve tento uživatel posoudí konečnou správnost našeho postupu a řekne nám, co se mu na našem postupu nelíbilo.

8. Závěr.

Předložený článek je shrnutím některých známých i neznámých postupů a doporučení pro tvorbu zpráv, sloužících pro komunikaci dvojice "člověk - počítač". Uvědomujeme si, že ne všechna doporučení mohou být realizována na počítačích s omezenými zdroji (malá paměť, pomalé terminály a pod.). Věříme však, že článek může pomoci ke zlepšení vztahu našich uživatelů k výpočetní technice.

Článek byl zpracován podle :

/1./ J. Martin, "Design of Man-Computer Dialogues", Prentice-Hall, Inc., Englewood Cliffs, NJ (1973)

/2./ M. Dean, "How a computer should talk to people", IBM Systems Journal 21, No 4, str. 424 - 453, (1982)