I N T E R A K T I V N Í P R O G R A M Y A P R A V I D L A D I A L O G U

Ing. Josef Tvrdík, CSc. - Krajská hygienická stanice Ostrava

1. Úvod

S rozvojem osobních počítačů, terminálových a počítačových sítí a podobných technických vymožeností se stává pro větší okruh lidí stále důležitější přímá komunikace koncového uživatele s výpočetním systémem. A právě snadnost či nesnadnost této komunikace často rozhoduje o tom, zda aplikační program je u uživatele úspěšný či nikoliv. Přívlastek „ user friendly " je nyní samozřejmou povinností při propagaci jakéhokoliv programového produktu; i programů, které se chovají k uživateli spíše nepřátelsky a navíc nesplňují základní a triviální požadavek - užití počítače má být pro uživatele výhodnější než neužití počítače.

Pro navrhování programů a vlastní programování různých aplikačních oblastí máme k dispozici již několik obecně známých a široce využívaných technologií. Pro návrh dialogu (komunikace) mezi programem a uživatelem tomu tak dosud není. Základní informace o interakci programu a uživatele lze nalézt v literatuře - viz např. /1,2/, na tomto semináři příspěvky /3,4/, avšak s těmi při návrhu dialogu většinou nevystačíme.

Designer programu (programátor, analytik) při návrhu rozhraní programu a uživatele často pociťuje, že musí řešit úlohu, pro kterou se mu nedostává znalostí nebo alespoň dostatek praxí ověřených doporučení. A právě touto problematikou se podrobně zabývají Rubinstein a Hersh v knize /5/.

2. Konceptuální model a vnější mýtus

Člověk si obvykle chování nějakého složitého systému (zpočátku pro něj černé skříňky) snaží sám sobě vysvětlit nějakým jeho modelem. Důmyslnost modelu a použité představy (koncepty) jsou ovlivňovány jednak vstupy a výstupy systému, jednak znalostmi člověka, který tento konceptuální model vytváří. Konceptuální model bývá podstatně jednodušší než systém, jehož je obrazem. Přesto umožňuje alespoň částečně poznat, pochopit a vysvětlit chování složitého systému.

Příkladů i mimo oblast programového vybavení můžeme najít celou řadu, připomeneme alespoň jeden učebnicový - Bohrův model atomu.

Uživatelé si vytvářejí konceptuální modely programů. Vytvářejí si je na základě svých znalostí a zkušeností a především podle uživatelského rozhraní programu (vstupy, výstupy, jejich vztah). Toto uživatelské rozhraní, jeho obsah, forma a další vlastnosti se nazývá vnější mýtus /5/. Starostí návrháře programu je navrhnout a implementovat takový vnější mýtus aplikace (t.j. programu, systému, paketu programů), aby podporoval vytváření vhodných konceptuálních modelů aplikace u většiny jejích uživatelů, aby vnější mýtus a konceptuální model byly v souladu v celé dané aplikaci. Dobrý vnější mýtus usnadňuje uživateli zvládnutí systému, zkracuje dobu zácviku, minimalizuje listování v manuálu nebo helpech, snižuje možnost zadání nežádoucích činností a zvyšuje spolehlivost a odolnost systému.

3. Přístupy k návrhu dialogu člověk - počítač

Můžeme rozlišit tři základní přístupy k návrhu rozhraní uživatele a programu /5/:

1. intuitivní - opírá se o důkladnou znalost problému uživatele, jeho svyklostí atd. a metodou pokus - omyl se přibližuje požadovaným nebo alespoň přijatelným vlastnostem řešení
2. dodržování obecných pravidel dialogu
3. vytvoření modelu uživatelova chování a aplikace teorií z oblasti psychologie a pod.

Všeobecně je možno doporučit kombinaci všech tří přístupů s převahou druhého. Tak lze dosáhnout s nejmenším úsilím uspokojivých výsledků - interaktivních programů s dobrým uživatelským rozhraním.

Základním pravidlem návrhu rozhraní je : Poznejte uživatele. Jeho požadavky, svyklosti a znalosti jsou většinou dosti odlišné od představ a návyků řešitelů programového vybavení. Na druhou stranu, řešitelskou aktivitu a zodpovědnost za řešení není možné přenést na uživatele - není v této oblasti ani odborníkem, ani profesionálem.

Řešitel může vyhovět přáním uživatele. Ale jen těm, které uživatali prospívají nebo neuškodí. Dobrou alternativu řešení musíme uživatelům vnutit i přes jejich počáteční nevoli - viz podrobněji Jiříček /6/.

V práci /5/ je uvedeno několik desítek obecných pravidel, zásad, návodů a doporučení, kterými je užitečné se řídit při návrhu a implementaci interaktivních programů. Výběr z těchto pravidel následuje v dalších odstavcích.

4. Pravidla pro fázi návrhu

1) Oddělte návrh od implementace.
2) Formulujte a zapište poznatky z analýzy úlohy dříve, než je odložíte nebo zapomenete. Důležité pro návrh je ujasnit si :
   - kdo je uživatel
   - jaké úlohy nyní provádí
   - jaké má vzdělání a znalosti
   - jaké je jeho pracovní prostředí
   - jaký je vztah uživatele a dat
   - jaké jiné nástroje uživatel užívá
   - jak komunikují uživatelé navzájem
   - jak často se úlohy vykonávají
   - jaký je časový tlak na úlohy
   - co se stává při poruchách techniky, nehodách a jiných potížích.
3) Vytvořte model užívání - navrhujte pro skutečného uživatele, ne pro sebe.
4) Udržujte konzistentní vnější mýtus v celém navrhovaném systému.
5) Omezte rozsah systému
   - systém musí mít jasné a uživatelem pochopitelné hranice; systém nemůže umět všechno pro každého.
6) Omezte počet stavů systému a viditelně je rozlište
   - neužívejte stejný vstup pro více účelů, pozor na CTRL a SHIFT klávesy a podobné zdroje zmatků.
7) Minimalizujte množství informací nutných k užívání systému.
8) Uživatelskou příručku napište dříve, než začnete s implementací.

5. Jazyk komunikace

Zkušenost ukazuje, že lidé při interakci s počítačem se chovají tak, jako by platila pravidla rozhovoru člověka s člověkem. Proto :

1) Respektujte pravidla mezilidské konverzace
   - program by měl být kooperativní
   - konverzace není konfrontace

2) Neskákejte do řeči
   - nedůležitá zpráva by neměla rušit důležitou činnost jako je editování textu, zadávání příkazového řádku, vyplňování formuláře apod.

3) Neodpovídejte ani příliš dlouze, ani příliš krátce

4) Uživatel má vždycky pravdu
   - lepší je přístup „ nerozumím " než „ tys udělal chybu "

5) Nepředstírejte schopnosti programu, které nemá
   - nepokoušejte se „ vcítit " do nálady a pocitů uživatele
   - neužívejte ve zprávách 1. osobu jednotného čísla

   Zpráva „ Já nechápu, proč dnes už 10 x opakuješ tuto chybu " je jedním z odstrašujících příkladů.

6. Syntaxe jazyka komunikace

Obvyklá a odjinud známá pravidla činí syntaxi snadno zapamatovatelnou. Běžný jazyk nemá přepínače oddělené lomítky a pod.

1) Buďte konzistentní v užití jazyka
   - podobné operace podobným způsobem, stejná syntaktická pravidla

2) Syntaktická pravidla učte pomocí příkladu, ne pomocí formalismu

3) V příkazovém jazyku neužívejte speciální znaky ($@, \$, \&, *$).

7. Sémantika jazyka

Význam užitých pojmů musí být jasný. Pozor na slova souznačná a zejména na víceznačná.

1) Užívejte terminologii uživatele;
   - často se chybně užívají slova známá designerovi, nikoliv uživateli, např. pracovní paměť, zdrojový program, default, ukazatel (pointer), spojování (link), boot.

2) Užívejte běžného jazyka
   - čím méně nových umělých termínů, tím snadněji se uživatel jazyk naučí

3) Koordinujte všechny odpovědi systému
   - užívejte jednotnou terminologii

4) Vyhněte se užití slov, která mají odlišný význam ve světě uživatele než v programování, např. adresa, cyklus, maska, knihovna, klíč, přepínač atd.

5) Neužívejte výrazy s falešným nebo negativním přídechem
   - např. osudová chyba (fatal error), zabité soubory (killed files) atd.

6) Pravopis (spelling) příkazu opravujte jen tehdy, kdy je náprava jasná - podrobněji viz /2,3/.

7) Eliminujte zjevně i méně zjevně špatné termíny
   - uživatel by měl z příkazu usoudit na jeho význam a z významu usoudit alespoň na přibližný tvar příkazu

8) Vyhněte se nutnosti užití složitých dotazů - uživatelé nemají rádi složité podmínky výběru, negace logických výrazů, pletou si AND a OR, kvantifikátory atd.

8. Uživatelská dokumentace a další náležitosti

1) Uživatelskou příručku a ostatní prostředky výuky systému uvažujte jako součást systému. Všechny jeho složky musí pomáhat k vytváření konzistentního uživatelova konceptuálního modelu.

2) Uživatelskou příručku napište již podle specifikace návrhu. Potíže spojené se psaním uživatelské příručky ukáží na slabá místa návrhu. V tomto ranném stádiu řešení lze návrh změnit nejsnadněji. Přezkoumejte úplnost a srozumitelnost uživatelské příručky. Závěry uplatněte případnou změnou v návrhu systému.

3) Podporujte vytvoření jasných konceptuálních modelů dokumentace. Učebnice má vysvětlovat. Vysvětlení má začínat příkladem, zobecňování z příkladu je pro čtenáře snadnější, formální pravidla jsou užitečná až pro specifikování a konsolidaci již naučeného. Příručka slouží jako uživatelův rádce, proto má umožňovat snadné vyhledávání. K tomu slouží : obsah, rejstřík, jasná a průhledná struktura.

4) Ověřte, že helpy skutečně pomáhají. Neúčinný help je horší než žádný. Helpy musí být v jazyku uživatele.

5) Řešte jednotně helpy a ošetření chybových stavů.

9. Způsoby komunikace

Uživatel s interaktivním programem může komunikovat třemi základními způsoby :

- příkazovým jazykem
- pomocí menu
- přímým vkládáním hodnot proměnných (většinou vyplňováním formuláře)

Pro návrh způsobu komunikace se doporučuje dodržovat tato pravidla :

1) Vyhněte se alternativním způsobům v jedné aplikaci. Dáme-li uživateli např. možnost volby mezi komunikací příkazovým jazykem a nebo menu, přináší mu to jenom další složitost do jeho představ o systému. Snadnost ovládání programu nezávisí ani tak na způsobu komunikace, jako na dokonalosti řešení zvoleného způsobu. Je lépe mít program s jedním, ale dobře vyřešeným způsobem komunikace než se dvěma nedomyšlenými.

Příklad :

Výběr s uspořádané množiny (dejme tomu indexů) je v mnoha programech řešen přímým vkládáním hodnot takto (bohužel) :

```
POČET VYBÍRANÝCH PRVKŮ :      25
          1. PRVEK     :       3
          2. PRVEK     :       4
          .
          .
         25. PRVEK     :      50
```

To je přímo školní ukázka nedomyšlenosti, která je v rozporu s jazykem uživatele. Takový požadavek téměř každý člověk zapisuje ve tvaru podobném tomuto :

VYBRAT : 3 - 15, 18, 20, 41 - 50

Pokud to může takto zadat, není přímé vkládání hodnot horší než příkazový jazyk.

Pro příkazový jazyk :

2) Nedopusťte, aby interpunkční a speciální znaky byly nositeli významu v příkazovém jazyku.
   Uživatel má mít možnost užívat zkratek, program má opakovat plný text příkazu.

Pro menu :

3) Neužívejte dlouhé menu.
   Často užívaná menu nemají mít více než 5 - 6 položek. Krátké menu má být uspořádáno podle frekvence užití, dlouhé podle abecedy.

4) Položky menu opatřete návěštím (např. číslem) před textem menu. Návěští oddělte od ostatního textu.

Pro přímý vstup :

5) Odpovídejte v reálném čase (t.j. ihned).

6) Umožněte pohyb a automatizované nastavování kurzoru po vyplňovaných položkách formuláře.

7) Pevné texty a úprava formuláře jsou součástí vnějšího mýtu.

Pro všechny způsoby komunikace je nutné dovolit uživateli, aby mohl volit pořadí vyplňování položek, volbu činností, pořadí odstavců příkazů ap. všude, kde je to možné. Dále je nutné pamatovat na snadný únik z komunikace bez destrukčních následků.

10. Odpovědi

Během komunikace potřebuje uživatel vědět, co se děje na straně počítače. Potřebuje, aby na svoje akce dostával plynulé, správné, jasně formulované a zjevné odpovědi. Jinak znervózní, domnívá se, že počítač dobře nefunguje a začne bušit do klávesnice nebo se chovat podobným nerozumným způsobem. Pro odpovědi jsou užitečná následující pravidla:

1) Rychle registrujte a oceňte uživatelovu akci.
2) Poskytněte snadnou možnost úniku.
3) Žádná překvapení.
4) Potvrzujte informační obsah, ne znakový vstup.
5) Neodvádějte pozornost a nezneklidňujte uživatele.

Je nutno počítat s tím, že uživatel dělá chyby. Je to normální. V numerických datech bývá okolo 1% chybných údajů. Při komunikaci s programem jsou chyby dvojího druhu

- překlepy
- chybná rozhodnutí

Oba druhy chyb potřebují přiměřené ošetření. Doporučují se tato pravidla:

6) Chyby oznamujte jednoduše a ve správném kontextu (ne např. opožděně). Nevydávejte redundantní chybové zprávy. Dump nebo traceback je pro počítačového profesionála, ne pro uživatele.
7) I v chybových zprávách buďte slušní. Programy nemají být ani pyšné ani emocionální.
8) Neblamujte uživatele.
9) Neužívejte chybových kódů.
10) Označte místo zjištěné chyby (ukazovátkem, kurzorem a pod.).

11) Jasně rozlišujte mezi úspěšnou a neúspěšnou uživatelovou akcí.

12) Zprávu o chybě vyjadřujte v souladu s vnějším mýtem systému (ne v termínech designera).

13) Umožněte snadnou opravu chyb (i k tomu přiměřené helpy).

14) Odstranění chyb je součástí návrhu systému.

11. Doba odezvy

Uživatel požaduje, aby komunikace s výpočetním systémem byla plynulá, podobně, jako je konverzace mezi lidmi. S partnerem, který nereaguje do 2 - 4 sekund, si moc nepohovoříte. Avšak každá odpověď nemusí být stejně rychlá a definitivní. Pro dobu odezvy výpočetního systému se doporučují tato pravidla :

1) Umožněte uživateli pracovat plynule.
2) Oznamujte delší zdržení.
3) Čekání mají mít smysl - v ideálním případě by měla být doba odezvy úměrná uživatelem očekávanému úsilí počítače.

Následující tabulka uvádí některé akce podle nároku na dobu odezvy :

IHNED	pohyb kurzoru, odezva z klávesnice na obrazovku
KRÁTKÁ (~ 1 sec)	Help, získání menu, rolování obrazovky, detekce syntaktické chyby, detekce překlepu v klíčovém slově, výpis aktuální věty na obrazovku, vyhledání věty podle klíče, uložení nové věty, první odezva na delší požadavek
DLOUHÁ (několik sec)	vyhledání podle hodnoty (ne podle klíče), tisk jednostránkového dokumentu, zpráva o dostupnosti požadované části výpočetního systému (např. programu)
VELMI DLOUHÁ (desítky sec)	vyhledání v archivu, tisk dlouhého dokumentu

12. Presentace výsledků

Pro presentaci výsledků z počítače platí podobná pravidla jako pro tištěnou a grafickou presentaci textu a kvantitativních údajů bez užití počítače. V textech se mají užívat velká i malá písmena. Text jen z velkých písmen se čte pomaleji. Rozdělování slov zhoršuje jejich čitelnost. Proto vlastní jména, klíčová slova a důležité termíny nerozdělujeme. Zarovnání textu doprava vložením mezer nezlepšuje čitelnost textu, pokud mezery mezi slovy nejsou alespoň přibližně stejné. V českých a slovenských textech v opravdu dobrých aplikacích má být použita národní abeceda (diakritická znaménka).

Další doporučovaná pravidla :

1. Seznamy presentujte vertikálně (t.j. po sloupcích, ne po řádcích).

správné pořadí		špatné pořadí	
1	4	1	2
2	5	3	4
3	6	5	6

2. Identifikujte výstup (datum, název souboru atd.). Užívejte samoidentifikaci - např. 6. července 1908 je lepší než datum : 06/07/08

3. Čísluje se od 1, měří se od 0.

4. Užívejte formátování a řazení ke zvýraznění struktury výsledků. Číselné výstupy uspořádat do sloupků, tabulek, po každých 5 - 6 řádcích oddělit, desetinná tečka na pevné pozici, žádné vedoucí nuly u numerických údajů.

5. Užívejte grafů a diagramů.

6. Nezahlcujte uživatele informacemi ani vnějšími efekty.

Není dobré nacpat na jednu obrazovku mnoho informací. Výsledky by měly být presentovány jednoduše a užitečně. V rámci jednoho systému patří stejné věci na stejná místa obrazovky. Světelnými atributy lze rozlišovat různé typy informace, ale moc kontrastů ruší. Pozornost vyvolává změna, lidé snadno přivykají ustálenému stavu. Člověk je schopen dobře rozlišit jen

2 - 3 úrovně jasu. Příliš tmavé je nečitelné, příliš výrazné je nepříjemné. Blikání je velmi výrazné, proto by se mělo vyhradit pro signalizaci nějakého nebezpečí nebo neobvyklého stavu. Při užívání barev je potřeba dodat i další informaci pro barvoslepé. Tvar a velikost grafických značek jsou dobře rozlišitelné atributy. Důležité by mělo být větší než nedůležité. Tvar značky výrazně odlišný od ostatních přitahuje pozornost.

Tato doporučení platí nejen pro presentaci výsledků, ale i v ostatních částech vnějšího mýtu. Mnohobarevný formulář zaplňující celou obrazovku doprovázený dvojhlasnou melodií rozhodně není ideální řešení.

13. Testování

Kvalitu rozhraní posuzujeme podle toho, jak vyhovuje skutečnému uživateli.

Spolehlivě to lze zjistit jen empiricky, t.j. testováním na vhodném vzorku uživatelů. Pro spolehlivost závěrů je důležitá dostatečná velikost vzorku. Uživatelé vybraní pro testování nesmějí být zainteresování na výsledku. Návrhu a hodnocení testu se má zúčastnit řešitel rozhraní. Při testování funkční správnosti programů je obyčejně snaha řešitele z testování vyloučit. Při zkouškách vlastností rozhraní však nejde o chyby tak jednoznačně rozlišitelné. Právě řešitel má pro takový druh chyb lepší postřeh. Navíc závěry z testů může ještě uplatnit ke zlepšení produktu.

Podobně by měla být testována uživatelská dokumentace a další náležitosti již ve fázi návrhu.

14. Pravidla_s_nad_poměry *

V knize /5/ jsou uvedena i pravidla, nad kterými se obyčejný devizový tuzemec může prozatím jen usmát :

1) Navrhujte klávesnici jako součást systému.
2) Zjednodušte rozložení kláves a minimalizujte jejich počet.
3) Oddělte textové klávesy od řídících.
4) V případě, že uživatel má zaměstnané ruce nebo oči, uvažujte v návrhu hlasový vstup/výstup.

5) Při testování programu zaznamenávejte činnost uživatele na videorekordér.

15. Závěrečné úvahy

Návrh a implementace dialogu člověk-počítač je inženýrská činnost. Proto nemůžeme v této činnosti zapomenout na základní principy inženýrství:

- hledat nejjednodušší řešení
- uvažovat důkladně (t.j. systémově)
- inženýrství je kompromis
- dokonalé je nepřítelem dobrého.

Heslovitá pravidla z předcházejících odstavců nejsou ani ucelenou technologií, ani zárukou úspěšnosti řešení komunikace člověk-počítač. Pouze pomáhají snížit pravděpodobnost neúspěchu.

Mnohého čtenáře během čtení tohoto textu napadly přinejmenším tyto dvě otázky :

1) Proč se tak důkladně zabývat komunikací uživatele a programu ?
2) Jak vůbec vzít pro řešení dostatek času ?

K první otázce jen ve stručnosti připomeňme : Řešitelé programů jsou ekonomicky závislí (těsně či volně, zjevně nebo skrytě) na tom, zda se jejich programové produkty používají. Uživatel o funkční správnosti programu nepochybuje (doufejme, že většinou oprávněně). Kvalifikovaně posuzuje pouze vnější mýtus programu Má-li možnost výběru, vybere si lepší možnost.

Na druhou otázku je v podstatě jednoduchá odpověď: Je potřeba užívat vhodných softwarových nástrojů. Samotné běžné programovací jazyky takovým vhodným nástrojem nejsou. Je však k dispozici řada formulářových systémů, modulů pro ovládání kurzoru, generátorů příkazového jazyka a dalších prostředků, ze kterých lze postupem zdola nahoru potřebné nástroje vytvořit /7/.

L i t e r a t u r a :

1. Martin J., Design of Man - Computer Dialogues,
Prentice Hall, 1973

2. Dean M., How Computer Should Talk to People,
IBM System Journal, 21, 424 - 453, 1962

3. Klečka J., Počítač komunikuje s člověkem,
Sborník Programování 84, str. 155 - 168, DT Ostrava, 1984

4. Bébr R., Recept na jídelníček anebo metodika dialogu formou menu,
ditto, str. 143 - 154

5. Rubinstein R., Hersh H.M., (with the assistence of H.F. Ledgard),
The Human Factor - Designing Computer Systems for People,
Digital Press, 1984

6. Jiříček P., Racionální programování složitých systémů,
Sborník Programování 86, str. 23 - 29, DT Ostrava, 1986

7. Lexa I., Shora - dolů nebo zdola - nahoru ?,
Sborník Programování 87, str. 3 - 12, DT Ostrava, 1986