Místo návrhu uživatelského rozhraní v životním cyklu vývoje programového systému aneb systematický přístup k návrhu uživatelského rozhraní

Alena Buchalcevová, Milan Drbohlav

VŠE – Praha, Katedra informačních technologií, nám. W. Churchilla 4, 130 67 Praha 3

Abstrakt

Nedílnou součástí návrhu každého programového systému je i návrh uživatelského rozhraní. Ačkoliv existuje disciplína, která se tímto návrhem zabývá (HCI – Human-Computer Interaction) z mnoha hledisek (obecné principy návrhu, typy interakce člověk-počítač, ergonomie), stávající metodiky vývoje programových systémů obvykle s touto problematikou explicitně nepočítají, nebo ji odsouvají podobně, jako jiné kvalitativní aspekty programového systému do pozdějších fází životního cyklu vývoje systému. V praxi softwarové firmy řeší tuto problematiku zpravidla pouze na úrovni specifikace interních pravidel pro návrh uživatelského rozhraní. Některé z těchto specifikací jsou dokonce publikovány knižně (např. specifikace fy. Microsoft), či alespoň na WWW. Poměrně málo se hovoří o integraci návrhu uživatelského rozhraní a procesu vývoje software, jinými slovy o metodice návrhu uživatelského rozhraní a její integraci s metodikami návrhu programového systému. Co tyto signály znamenají? Existují metody návrhu uživatelského rozhraní? Pakliže ano, jaké jsou vlastnosti takové metody? Lze ji integrovat se stávajícími metodikami návrhu programového systému? To jsou otázky, na které se snažíme, alespoň z části hledat odpovědi v našem příspěvku.

Úvod

Uživatelské rozhraní[1] programových systémů je jedním z klíčových faktorů, které ovlivňují úspěšnost a použitelnost daného systému. V současné době se má uživatel možnost setkat s následujícími UI:

- příkazový řádek (zejména u mainframových systémů, či některých typů operačních systémů)
- textové uživatelské rozhraní – CUI (Character User Interface)
- grafické uživatelské rozhraní – GUI (Graphical User Interface)
- multimediální rozhraní (v současné době chápané zejména jako rozšíření GUI o nové typy prezentací, resp. vstupu dat)

[1] V dalším textu budeme pro pojem „uživatelské rozhraní“ používat zkratku UI (z anglického User Interface)

- rozhraní typu virtuální realita (jehož skutečné možnosti se dnes stále ještě hledají)

Příležitosti a hrozby GUI

Zejména s příchodem grafického uživatelského rozhraní (GUI) se nesmírně rozšířily jak možnosti vývojářů, tak uživatelů. Uživatel již není nucen vykonávat příkazy v předem určeném pořadí, ale stává se sám nositelem veškerého dění. Je to uživatel, který určuje, jaké akce bude provádět v jakém pořadí, s jakými objekty. Zatímco v tradičních systémech bylo možné interakci popsat paradigmatem *akce – objekt*, uživatel tedy vybral nejprve akci (z menu nebo pomocí funkční klávesy) a potom zadal data, dochází v grafickém uživatelském rozhraní k přechodu na nové paradigma *objekt – akce*. Uživatel nejprve vybere objekt (často pomocí grafické reprezentace) a teprve potom volí akci.

Neustálý vývoj prostředků pro tvorbu GUI vedl k tomu, že se snížila výrazně náročnost tvorby tohoto typu UI a při dodržení základních pravidel jeho použití *může* skutečně vést ke zvýšení produktivity práce koncového uživatele, *může* vést ke snazší a příjemnější práci s počítačem. Jednotné a konzistentní GUI[2] používané v řadě aplikací zkracuje výrazně dobu učení se aplikaci.

Bohužel tvrdá realita je poněkud odlišná. Programová rozhraní dnešních dodavatelů operačních systémů poskytujících GUI nejsou otevřená ve smyslu vzájemné interoperability a přenositelnost aplikací mezi jednotlivými platformami (Windows firmy Microsoft, Motif OSF, OS/2 firmy IBM, Macintosh firmy Apple) je problematická. A navíc veškeré aplikační programy vybudované nad GUI API jsou produkty, které mají vlastní GUI ušité na míru danému produktu. Aplikační programátoři vytvářejí vlastní ikony a píší vlastní kód obsluhující akce nad těmito ikonami.

Z výše uvedeného je evidentní, že GUI má potenciální kvalitativní přednosti, ale jejich dosažení nevyplývá automaticky z prostého použití GUI. Tvorbu GUI[3] je třeba řídit, je třeba se zaměřit na uživatele a jeho vztah k počítači, resp. programovému systému. Vztahem uživatele a výpočetního systému se zabývá zejména disciplína označovaná jako HCI (Human Computer Interaction), která zavádí i pojem návrhu zaměřeného na uživatele (User Centered Design) jako protiklad k návrhu zaměřeného na úlohy (Task Centered Design). Základními principy návrhu zaměřeného na uživatele jsou:

- transparentnost (kterou se rozumí zejména intuitivní ovládání a předvídatelné chování programového systému)
- pružnost (přizpůsobivost chování uživatele a jeho potřebám)
- přehlednost (snadná orientace uživatele v systému)
- produktivita (založená na jednoduchosti, přehlednosti obrazovek, zpětných vazbách)

[2] Domníváme se, že tato vlastnost je jedním z nejsilnějších argumentů hovořících ve prospěch GUI. Lze říci, že v se v podstatě jednotlivá GUI vzájemně napodobují. Například otevření složky se soubory ve Windows je velmi podobné provedení analogické operace na počítači Macintosh a základní grafické reprezentace objektů jsou (alespoň co do svého chování) víceméně totožné.

[3] Přirozeně, je nezbytné řídit tvorbu každého UI, v případě GUI je tato potřeba umocněna potenciálem, který v sobě „dobré“ GUI skrývá

- integrita (ochrana nebezpečných operací, implementace undo funkce a on-line nápovědy)

Kvalitativní požadavky

Uvedené principy lze rovněž popsat jako další kvalitativní požadavky na programový systém. Je neoddiskutovatelným faktem, že kvalitativní požadavky obecně představují zpravidla úzké místo realizace každého programového systému. Dle našeho názoru pro to existují v zásadě dva důvody:

- vlastnosti jako výkonnost, použitelnost, modifikovatelnost, přenositelnost jsou příliš obecné a při specifikaci zadání jsou jen velmi zřídka detailněji formulovány tak, aby nedošlo k pozdějším rozčarováním
- současné metodiky vývoje programových systémů se zaměřují zejména na uspokojení funkčních požadavků na systém s tím, že kvalitativní požadavky explicitně buď nezvažují, či úvahy o nich posouvají do pozdějších fází životního cyklu (implementace, testování).

Je v zájmu tvůrce programového systému, aby se snažil co nejdříve zjistit, jak konkrétně lze v daných podmínkách jednotlivé kvalitativní požadavky specifikovat, aby se snažil odhadnout, do jaké míry budou požadavky (funkční i *kvalitativní*) uspokojeny, a aby se snažil míru uspokojení také řídit. Jsme přesvědčeni, že úvahy o uspokojení kvalitativních požadavků patří principiálně do ranějších fází vývoje programového systému než je implementace Nechceme tím vyloučit rozhodování o nich z pozdějších fází životního cyklu, resp. jedné iterace (v případě dnes populárního iterativního přístupu k vývoji). Tato rozhodnutí provázejí všechny kroky vývoje systému, chceme pouze zdůraznit, že i s kvalitativními aspekty se musí zacházet jako s funkčními požadavky a musí být na zřeteli vývojářům v průběhu celého procesu vývoje.

Systematický přístup k návrhu UI

Existuje tedy systematický přístup k uspokojování kvalitativních požadavků na systém, v našem případě požadavků souvisejících s UI a tedy s „použitelností“[4] systému? V dalším textu ukážeme, že takový přístup existuje a jsme přesvědčeni o tom, že přináší nižší náklady, kratší dobu realizace a přispívá i k větší úspěšnosti projektů vývoje programového systému. Hlavním důvodem je fakt, že systematický přístup předpokládá ověřování míry uspokojování i kvalitativních požadavků na systém v průběhu celého životního cyklu a umožňuje tak upozornit na nevhodné cesty vývoje ještě před tím, než jsou realizovány.

Zdroje systematického přístupu k návrhu UI

Systematický přístup k realizaci UI by dnes měl využívat čtyři klíčové zdroje:

1. mezinárodní normy týkající se návrhu UI,
2. doporučení kognitivní psychologie a psychologie práce

[4] Vlastnost „použitelnost“ bývá s uživatelským rozhraním spojována nejčastěji a lze ji definovat jako snadnost použití systému a výcviku uživatelů. Na této definici je vidět, nakolik jsou známé kvalitativní požadavky obecné. Je velmi vhodné použitelnost hodnotit na základě hodnocení dílčích kategorií, jakými jsou: naučitelnost, efektivnost práce, samovysvětlovací schopnost, ovládání uživatelem, subjektivní pocity uživatele

3. tzv. platformová pravidla návrhu UI,
4. firemní pravidla návrhu UI,

Mezinárodní normy

Máme zde namysli zejména ISO normu ISO 9241 a směrnici Evropské Rady 90/270/EWG, které se týkají ergonomie software, definují pojem použitelnosti a stanovují některá kritéria pro návrh UI. Tyto normy však obsahují jen velmi obecné nástroje pro hodnocení UI a jen základní nástin způsobu použití obecných kritérií při návrhu UI.

Doporučení kognitivní psychologie a psychologie práce

Vedle výše uvedených obecných norem týkajících se použitelnosti programového systému existuje celá řada pravidel formulovaných na základě výzkumů kognitivní psychologie a psychologie práce. Tato pravidla jsou publikována (např. v [1]), obvykle však neobsahují kontext, ve kterém je vhodné je použít. Pojem použitelnost programového systému se přitom evidentně vztahuje na konkrétní kontext, ve kterém je programový systém použit konkrétním typem uživatele jako nástroj pro realizaci určitého úkolu.

Platformová pravidla

Pojmem platformová pravidla označujeme to, co je běžně uváděno v literatuře pod názvem „style guides“. Jedná se o sadu doporučení k tvorbě UI na dané platformě vydaných významnými dodavateli software. Tato doporučení jsou opět zpravidla publikována (knižně – viz např. [2], nebo i elektronicky viz např. [3]). Nejznámějšími z těchto doporučení jsou:

- Common User Access (CUA) firmy IBM
- The Windows interface guidelines firmy Microsoft
- style guides firmy Apple (pro rozhraní vytvářené pro počítače Macintosh)
- style guide organizace OSF (pro Motif)

Hlavním účelem těchto platformových pravidel je zajistit specifický vzhled určitých (platformově závislých) typů UI. Tento přístup vychází vstříc i jednomu z ISO kritérií „konformita rozhraní s očekáváními uživatele“. Doporučení však nezohledňují specifika jednotlivých typů uživatelů, specifika jednotlivých typů aplikací, resp. použití programového systému jako nástroje. Tato doporučení se rovněž netýkají estetických vlastností UI.

Základními doporučeními pro dobrý návrh UI z této kategorie jsou:

1. ovládání systému uživatelem: uživatel musí mít pocit, že je to on, který řídí software
2. přímost: UI je třeba navrhnout tak, aby uživatel mohl přímo manipulovat softwarovou reprezentací objektů
3. konzistence: umožňuje uživateli využít existující znalost; jedná se přitom o všechny aspekty UI – jmenné konvence, vizuální prezentace, operace.
4. připomenutí: efektivní UI musí upozornit na dosud neprovedené klíčové operace

5. odezva: dobré UI musí vždy odpovědět na akci uživatele; vizuální nebo i audio odezva by měla doprovázet každou akci, aby potvrdila, že programový systém reaguje na vstup uživatele.
6. estetika: vizuální prvky ovlivňují chování uživatele a použitelnost programového systému.
7. jednoduchost: UI musí být jednoduché, snadno naučitelné.

Většina platformových pravidel se omezuje na aplikaci výše uvedených doporučení pro dobrý návrh při vizuálním návrhu UI. V [2] je věnována pozornost i speciálním oblastem v rámci návrhu UI jako je:

- zvuk
- internacionalizace software
- síťové zpracování
- integrace s operačním systémem
- pomoc uživateli

Firemní pravidla

Jedná se o jakousi precizaci a rozšíření platformových pravidel, která realizuje každá softwarová firma. Tato pravidla by měla být nedílnou součástí firemních standardů každé softwarové firmy. Cílem je zajistit specifický vzhled programových systémů dodávaných příslušnou firmou. Pravidla musí zohledňovat estetické aspekty, jakož i přejímat (či upravovat) platformová pravidla. Konkrétním projevem je existence standardizovaných masek obrazovek. Firemní pravidla se však zaměřují pouze na standardní vzhled a nesnaží se obvykle zohledňovat specifické požadavky jednotlivých typů uživatelů. Dalším velmi běžným neduhem je problematická aktualizace těchto pravidel, zejména při přechodu na jinou platformu (tj. aktualizace části věnované převzatým, resp. upraveným platformovým pravidlům).

Metody návrhu UI

Metoda návrhu UI představuje systematický přístup k této problematice. Podobně, jako u kterékoliv jiné metody, potýká se vývojář i u metody návrhu UI se dvěma klíčovými problémy:

a) volbou vhodné metody jedná se o poměrně mladou oblast a stávající metody nejsou obvykle ještě dopracované (zejména po stránce formalizace či automatizované podpory)
b) integrací zvolené metody do používané metodiky vývoje UI metodiky vývoje programového systému obvykle neobsahují metodu pro návrh UI a tento kvalitativní aspekt zůstává buď nepovšimnut, nebo „na okraji“ v podobě obecných doporučení či odvolání (které nejsou často ani explicitně vyjádřené); klíčovým faktorem je, zda používaná metodiky umožňuje absorbovat nové metody

V dalším textu stručně popíšeme tři metody návrhu UI. Na základě vyhodnocení předností a nedostatků těchto tří metod jsme dospěli k formulaci vlastního přístupu k návrhu UI, na kterém chceme dále pracovat jak po stránce metodické (integrace s konkrétními metodikami vývoje programového systému), tak i technologické (vytvoření automatizované podpory).

Human – Computer Interface Design process

Curriculum Vitae metody
- vznik: 1992
- autor: Carlow International Inc.
- vytvořeno na zakázku pro Goddard Space Flight Center
- základní princip: identifikace uživatelských požadavků a kritérií návrhu HCI a jejich integrace s požadavky na vývoj SW
- bližší popis v [4]

Kroky metody
1. analýza funkcí
2. srovnávací analýza
3. analýza úloh
4. návrh konceptů
5. prototypování a testování

Základní postup

V kroku „analýza funkcí“ se identifikují základní funkce, které se budou s pomocí programového systému provádět, a další atributy těchto funkcí jako je jejich pořadí, frekvence provádění, nutnost účasti operátora apod. Analýza se provádí iterativně.

Pokud se navrhuje UI pro systémy, které již existují, podrobí se analýze (2. krok – srovnávací analýza) rozhraní existujícího systému a zaznamenávají se všechny pozitivní i negativní aspekty, zejména:
- problémy s použitelností,
- pozitivní aspekty UI, které mají být zachovány,
- pro každou funkci se diskutují s uživateli silné a slabé stránky rozhraní,
- zjišťuje se použitelnost systému měřením výkonu.

Pro jednotlivé funkce systému jsou ve 3. kroku (analýza úloh) identifikovány tzv. uživatelské úlohy a pořadí jejich provádění. Požadavky úloh jsou identifikovány jako:
- požadavky na informace, tj. informace potřebné pro provedení úlohy a jejich charakteristika
- požadavky na realizaci, tj. kritéria pro každou úlohu, limity časové, limity pracnosti, tolerance k chybám
- požadavky rozhodování, tj. pravidla rozhodování, možné volby a odezva na každou úlohu
- požadavky na podporu vykonání úlohy z hlediska systémových zdrojů

V dalším kroku jsou požadavky úloh přehodnocovány a zabudovány do návrhu.

Předmětem 4. kroku (návrh konceptů) je tvorba alternativních návrhů založených na analýzách z předchozích kroků. Jedná se o iterativní a tvůrčí proces, který je možné rozdělit do těchto činností:
- vývoj konceptů zobrazení – kódování, formát, vztahy mezi obrazovkami

- vývoj konceptů interakce/transakcí[5] – koncepty pro dialogy.
- vývoj konceptů pro podporu rozhodování

Navržené koncepty jsou v posledním kroku (provádění uživatelských testů) testovány, aby se zjistilo, zda rozhraní odpovídá potřebám uživatelských úloh. Cílem je odhalit eventuální problémy ještě před distribucí systému.

Method for User Interface Engineering (MUSE)

Curriculum Vitae metody
- vznik: 1993
- autor: Prof. Peter Gorny, Univerzita v Oldenburgu
- vytvořeno v rámci projektu Ministerstva pro vědu a kulturu Dolního Saska
- od 1997 existuje ve verzi MUSE II
- základní princip: východiskem teorii chování, která považuje za základní rysy lidského chování cílevědomost, věcnost a sociální cítění; v tomto smyslu člení metoda rozhodnutí o návrhu systému do třech fází (pohledů), které lze aplikovat libovolně často
- bližší popis v [4], [9]

Kroky metody
1. definice požadavků
2. pojmenování potenciálních možností
3. sběr informací
4. použití pravidel
5. zaznamenání rozhodnutí

Základní postup

Uvedené kroky je třeba použít při úvahách o UI. Úvahy by měl návrhář při použití metody MUSE dělit do třech fází (pohledů):
1. účelnost
2. interakce
3. prezentace

Ve fázi účelnosti dochází k překlopení výsledků analýzy stávajícího systému práce do rozhodnutí o požadovaných funkcích programového systému. Tyto funkce jsou přitom řazeny do čtyřech kategorií: uživatelské, řídící, přizpůsobovací a metafunkce. Rozhodnutí by se měla opírat jednak o zkušenosti s podobnými systémy a dále pak o
- konkrétní znaky úloh, které je třeba řešit
- uživatele (jeho individuální vlastnosti abstrahované do tzv. „rolí“)
- způsoby užití nástroje

Tato trojice {úloha, role, způsob užití nástroje} určuje ergonomické požadavky na UI. Výsledkem fáze účelnosti by měl být seznam funkcí softwarového nástroje nezbytných pro realizaci příslušných aktivit

[5] Interakce jsou specifické výměny informací a příkazů mezi uživatelem a počítačem. Transakce zahrnují vstup dat, zobrazení, manipulaci, zpracování, načtení a uložení dat , které je spojeno s každou interakcí

Ve fázi interakce se realizují rozhodnutí o způsobu interakce uživatele a stroje, jakož i o struktuře dialogu[6] (včetně podmínek). Základními formami interakce jsou:

- dialog pro vstup dat
- příkazový dialog
- výběrový dialog
- přímá manipulace

Specifikovaným funkcím jsou přiřazovaný jednotlivé výše uvedené formy interakce, či jejich kombinace.
Výše uvedené podmínky ve struktuře dialogu umožňují realizovat i rozhodnutí o povolených posloupnostech pracovních kroků při vyřizování úloh, resp. daném způsobu použití příslušného produktu. Výsledkem fáze interakce je tedy přiřazení forem interakce jednotlivým funkcím nástroje a vyspecifikování vstupních podmínek pro konkrétní použití jednotlivých forem.

Cílem fáze prezentace je stanovení konkrétního vzhledu jednotlivých forem interakce. Například pro formu interakce „výběrový dialog“ jsou k dispozici prostředky jako menu, seznam, radio-tlačítka, zaškrtávací pole. V této fázi lze s výhodou uplatnit publikované závěry kognitivní psychologie či psychologie práce[7].
V této fázi je třeba rovněž rozhodnout o předpokládaných vstupních/výstupních zařízeních, neboť řada forem interakce je realizovatelná jen na speciálních zařízeních. Výsledkem třetí fáze je stanovení konkrétního způsobu realizace pro jednotlivé formy interakce.

Na základě takto získaných informací lze již vytvořit prototyp UI. Napojení UI na vlastní aplikační systém nelze přirozeně v počátečních fázích vývoje realizovat, místo toho je třeba přístupy k datům a funkcím aplikace implementovat technikou „Mock-up“. Na závěr dojde k vyhodnocení prototypu za účasti uživatelů, ze kterého vzejdou podněty (ve formě požadavků) k vypracování nové verze UI.

Zobecnění

Teoreticky by mělo při použití metody MUSE vzniknout speciální UI pro každou trojici {úloha, role, způsob užití nástroje} (díky rozdílům v jednotlivých znacích). To však přirozeně není ani ekonomické, ani praktické. Proto je úkolem návrháře spojovat podobné požadavky a redukovat tak počet UI. Extrémní případ „jedno UI pro všechny uživatele a úlohy“ vede pak ke standardizovanému programovému systému. Je však třeba poznamenat, že tento extrémní případ nikdy nenastane, neboť i vývojáři tzv. standardizovaného programového systému mají – i když často nevědomky – svůj pohled na uživatele a na úlohy, které je možné systémem řešit.

SALVO
Curriculum Vitae metody

- vznik: 1997
- autor: Prof. Wilson, V., University of Wisconsin, Connolly, J., California State University

[6] Dialogem se rozumí posloupnost pracovních kroků
[7] Příklad: menu by nemělo obsahovat více jak 10 položek; místo toho je vhodné použít hierarchicky uspořádaný menu systém či seznam, či vybrat nejčastěji používané položky a umístit je samostatně.

- vytvořeno pro potřeby výuky návrhu UI
- základní princip: integrace návrhu zaměřeného na uživatele a návrhu zaměřeného na úlohy
- bližší popis v [8]

Kroky metody:
1. specifikace determinantů UI
2. přijetí systémových standardů
3. využití zvyklostí uživatelů
4. vizualizace návrhu
5. testování

Základní postup

V rámci kroku „specifikace determinantů UI“ se určují nejdůležitější faktory, které ovlivňují formu a použitelnost UI. Ty jsou označovány jako determinanty UI a patří mezi ně:
- uživatelé, tj. kdo bude systém užívat ?
- úlohy, které bude systém podporovat, tj. na co se bude systém používat
- prostředí, ve kterém systém pracuje, tj. kde a jak se bude systém používat

Krok „přijetí systémových standardů“ znamená přijetí platformových, resp. firemních pravidel pro návrh UI a má zajistit zejména konzistenci navrhovaného UI.

3. krok „využití zvyklostí uživatelů“ si klade za cíl zmapovat dosavadní způsoby práce uživatelů a využít získané poznatky k návrhu UI. Pokud např. mají uživatelé zkušenosti s vyplňováním papírových formulářů, bude pro ně jistě jednodušší vyplňovat formulář o podobném formátu i na obrazovce. Pro dosažení tohoto cíle je možné postupovat v těchto krocích:
- stavět na stávajících dovednostech uživatelů,
- minimalizovat množinu potřebných dovedností,
- stejné dovednosti využívat kdekoli je to možné,
- používat zpětnou odezvu systému tak, aby se dobře odlišil různý kontext

Krok „vizualizace návrhu“ zdůrazňuje přednosti vizualizace jako nástroje pro názornou dokumentaci a objasnění navrhovaného UI. Vizualizaci (prototypování) lze relizovat dvojím způsobem:
1. „Cocktail napkin“ technika se používá v raných fázích návrhu a jejím výsledkem jsou jednoduché, zatím nepřesné obrázky jednotlivých obrazovek bez vzájemného propojení
2. Mock-up je technika prototypování celého systému nebo jeho vybrané části předpokládající vytvoření jednotlivých obrazovek a jejich vzájemné propojení s tím, že přístup k datům a vlastní aplikace jsou pouze simulovány; techniku lze realizovat off-line (na papíře) či on-line na obrazovce s využitím RAD vývojových nástrojů jako jsou Visual Basic, Delphi nebo nástrojů pro prototypování.Ve velkém systému je obtížné vytvořit mock-up návrhy pro celý systém. Proto se doporučuje vytvářet horizontální a vertikální návrhy založené na úlohách a scénářích a dalších UI determinantách. Horizontální návrhy zachycují aplikaci ze šířky, menu,

dialogy a okna. Vertikální návrhy slouží k zachycení detailů určitých částí systému a jsou reakcí na určitý scénář.

Poslední krok „testování“ návrhu UI uživateli v raných fázích vývoje může snížit celkové náklady projektu. Uživatelé odhalí neočekávané problémy, jejichž odstranění až po implementaci by bylo mnohem nákladnější. Pro testování lze využít techniku „think-aloud“, která předpokládá, že vývojáři najdou reprezentativní uživatele pro testování systému. Měli by to být uživatelé, kteří se neúčastnili návrhů UI. Technika probíhá v následujících dílčích krocích:

1. uživateli se zpřístupní systém, resp. jeho prototyp a popíše se scénář úloh, které má vykonat; scénář popisuje co se má vykonat nikoli jak.
2. uživatel začne se systémem pracovat s cílem realizovat specifikovaný scénář; je nezbytné, aby přemýšlel nahlas a sděloval všechny své úvahy vývojářům; vychází se přitom z přesvědčení, že pokud má uživatel potíže není to jeho problém, ale problém systému.
3. vývojář pouze sleduje a zaznamenává postup práce uživatele; zvýšenou pozornost je přitom přirozeně třeba věnovat zejména oblastem, s nimiž má uživatel potíže, nemůže např. pokračovat, vrací se zpět, aby zjistil správný postup, použije nesprávný příkaz nebo nástroj, či volá on-line nápovědu.

Hodnocení sledovaných metod

Z našeho pohledu důležité vlastnosti sledovaných metod uživatelského rozhraní jsme zachytili ve srovnávací tabulce (viz Tab. 1). Z hodnocení vyplývá, že společnými nedostatky metod návrhu UI jsou zejména:

- jen velmi obecně popsaný způsob integrace metody s metodikami vývoje programových systémů
- zatím nedostupná automatizovaná podpora
- špatná podpora zobecňování v oblasti návrhu UI, kdy je tato činnost ponechána plně na schopnosti návrháře

Vlastnost	**HCI**	**MUSE**	**SALVO**
automatizovaná podpora	nezjištěna	nezjištěna	nezjištěna
formalizace jednotlivých kroků	A	N	N
uplatnění ve fázi analýzy	N	A	A
uplatnění ve fázi návrhu	A	A	A
práce se vzory	A	A	pouze pravidla
integrace s metodikou vývoje	N	N	N
multidimenzionalita	A	A	A
zohlednění dalších kvalitativních aspektů	A	N	N
testování návrhu	A	N	A
podpora zobecnění	N	N	N
zohlednění kulturních rozdílů	N	A	A

Tab. 1: Hodnocení sledovaných metod návrhu UI

Poznámky k tabulce:

- práce se vzory: uplatňuje metoda znalosti získané z analogických systémů, uplatňuje platformové či firemní pravidla?
- integrace s metodikou: obsahuje metoda konkrétní návod na integraci s nějakou metodikou vývoje programového systému?
- multidimenzionalita: zaměřuje se metoda na více determinant UI?
- testování návrhu: popisuje metoda konkrétní techniku testování návrhu UI?
- podpora zobecnění: podporuje metoda vyhledávání podobných scénářů a redukci počtu UI?
- zohlednění kulturních rozdílů: podporuje metoda internacionalizaci UI, tj. tvorbu vícejazyčných UI

Rozšíření metody MUSE
Uvedené nedostatky nás inspirovaly k formulování vlastní představy o metodě návrhu UI, která je v podstatě rozšířením metody MUSE.
Systematický přístup k návrhu UI by měl obsahovat dvě základní kategorie kroků:

- platformově nezávislé kroky, které je vhodné realizovat již v raných fázích vývoje systému (sběr požadavků, analýza)
- platformově závislé kroky, které je možné realizovat ve fázi návrhu po rozhodnutí o architektuře programového systému; je účelné je realizovat i před rozhodnutím o platformě, na níž bude programový systém realizován

Platformově nezávislé kroky

Fázi sběru požadavků je v kontextu návrhu UI vhodné rozšířit zejména o specifikaci (nebo spíše rozšíření specifikace) uživatelů; máme zde na mysli jednotlivé typy uživatelů, u kterých je vhodné zjišťovat (vždy průměrnou hodnotou) zejména věk, jazykové znalosti, dosavadní zkušenosti s rozhraním k programovému systému.

Další významnou determinantu UI, uživatelské požadavky, je možné plně převzít z informací získávaných pro účely specifikace funkčních požadavků s tím, že je třeba přesně rozlišovat, od kterého typu uživatelů daný požadavek pochází. Jednotlivým uživatelským požadavkům je třeba následně přiřadit možné formy interakce. Rozlišujeme přitom následující typy interakce:

- interakce vyvolaná systémem (oznámení skutečnosti zobrazením na obrazovce)
- interakce vyvolané uživatelem
  - vstup
  - výběr
  - příkaz

Dále je třeba definovat i požadované posloupnosti specifikovaných interakcí. Pakliže se fáze analýzy zabývá specifikací uživatelských scénářů, je možné do značné míry vyjít právě z nich.

Platformově závislé kroky

Výstupy těchto kroků je možné uplatnit jednak pro ověření uspokojení uživatelských požadavků na použitelnost systému (podobně jako výstupy kroků platformově nezávislých) a dále pak pro prezentaci variantních návrhů UI uživatelům programového systému (ještě před rozhodnutím o platformě, na které bude systém realizován). Na výstupech těchto kroků již budou zřetelné odlišnosti realizace UI na jednotlivých platformách.

Prvním krokem z této kategorie je vymezení možných způsobů realizace jednotlivých forem interakce. Využít by se zde měla zejména *repository* s uloženými scénáři definujícími posloupnosti forem interakce a s jejich konkrétními řešeními (tj. specifikací způsobů realizace jednotlivých forem interakce). Stejné scénáře indikují potenciální možnost použití stejných způsobů realizace příslušných forem interakce. Odlišné scénáře jsou naopak potenciálními kandidáty pro zařazení do repository.

Dalšími zdroji pro vymezení možných způsobů realizace jednotlivých forem interakce jsou dosavadní zkušenosti návrháře[8] a firemní pravidla zohledňující jak pravidla platformová, tak i doporučení z oblasti psychologie práce.

Posledním krokem je vytvoření prototypu UI technikou „mock-up“ a jeho ověření v diskusi s uživatelem.

Směry další práce

V současné době se zaměřujeme zejména na metodické dopracování uvedeného postupu a to zejména ve dvou oblastech:

1. formalizace jednotlivých kroků
2. postup integrace metody se stávajícími metodikami vývoje programového systému; v této oblasti vycházíme ze srovnání metamodelu popsané metody a metamodelů používaných sledovanými metodikami. Konformnost těchto metamodelů je základním předpokladem bezproblémové integrace metody návrhu UI do metodiky vývoje programového systému

Aktuální výsledky práce budou prezentovány v průběhu konference.

Dalším krokem bude pak vývoj automatizované podpory metody návrhu UI integrované do zvolené metodiky vývoje programového systému.

Literatura

[1] Smith, S., Mosier, J.: Guidelines for designing user interface software, Bedford, MA, The Mitre Corp., 1986

[2] Microsoft Corp.: The Windows interface guidelines for software design, Redmond, WA, Microsoft Press, 1995.

[3] http://www.microsoft.com/win32dev/uiguide

[4] Carlow International Inc.: Human-Computer Interface Guidelines, Falls Church, Virginia, 1992

[8] Odvoláváme se zde na známé pravidlo, kdy dobré zkušenosti vedou k opakování, špatné zkušenosti způsobují pravý opak

[5] GUI Design, LBMS Process Engineer Version 2.0
[6] Sarna, D., Febish, G.: Rychlé navrhování aplikací pro Windows, Unis Publishing, 1994
[7] Gorny, P., Viereck, A., Qin, L., Daldrup, U.: Slow and Principled Prototying of Usage Surfaces: a Method for User Interface Engineering, sborník konference RE´93 – Prototyping, Bonn, 1993
[8] http://vance.mis.uwec.edu/saldocs/salvo.htm
[9] Gorny, P.: Tutorial zur Fachtagung D-CSCW'94, Marburg (http://www-cg-hci-e.informatik.uni-oldenburg.de/MUSE.html)