Tvorba uživatelských rozhraní

Prezentace přednášek

Ústav počítačové grafiky a multimédií

Téma přednášky

Experimenty a vyhodnocení rozhraní

Motto

„Jak jde měřit/vyhodnotit vlastnosti rozhraní?
Jak je posuzovat?“

Obsah:

- Přehled metod vyhodnocení uživatelských rozhraní
- Empirické metody
- Modelování
- Vnímání
- Iterační návrh rozhraní
- Shrnutí

Metody analýzy komunikace člověk-stroj

Aby bylo možno posoudit, zda je uživatelského rozhraní navrženo správně, je třeba analyzovat dialog člověka se strojem před posuzované rozhraní. K tomu lze použít různé metody. (Čerpáno z Ray E. Eberts: "User Interface Design", Prentice Hall Inc. 1994.)

Pro analýzu dialogu člověka se strojem je použitelná například metoda:

- Empirická
- Modelování
- Vnímání (cognitive)
- Antropomorfní (antrophomorphic)

Experimentální metoda

Empirická (experimentální) metoda je nejjednodušší metodou analýzy komunikace člověk-stroj. Nevyužívá žádných znalostí o způsobu zpracování komunikace na straně člověka.

Předpokladem úspěšné aplikace metody je především:

- Správná formulace otázky
- Identifikace závislostí
- Eliminace závislostí, případně jejich měření
- Výběr proměnných (nezávislé i závislé)
- Zhodnocení realizovatelnosti a nákladnosti experimentu

Každý experiment je třeba správně statisticky vyhodnotit.

Formulace otázky

Aby bylo možno seriózně provést analýzu, je třeba formulovat otázku poměrně přesně. Nestačí otázka typu "Je nové rozhraní lepší než staré?".

Typicky by měla otázka, na kterou chceme od analýzy odpověď obsahovat:

- Specifikaci předmětné skupiny uživatelů
- Určení zda bude analýza absolutní, nebo srovnávací, případně s čím bude rozhraní srovnáváno
- Jaké ukazatele rozhraní budou vyhodnocovány (čas, chyby...) a na jakém zadání.

Je zřejmé, že analýzu nelze provést bez znalosti úlohy, kterou analyzované rozhraní a programové vybavení řeší.

Identifikace závislostí

Před provedením experimentu je třeba identifikovat, co může experiment zkreslit. Například to může být:

- Průběh experimentu (událost, která ovlivní průběh)
- Vývoj uživatelů (věk, zkušenost, znalost problému...)
- Opakované provádění experimentu nebo posloupnost provádění experimentů
- Rozdílnost měřících metod či přístrojů
- Odchylky v experimentálních skupinách uživatelů
- Odlišnosti v době a způsobu přípravy na experiment

Závislosti je třeba eliminovat během přípravy experimentu. Pokud nelze eliminaci spolehlivě zajistit, je třeba zaznamenat průběh experimentu tak, aby bylo možno závislosti identifikovat dodatečně.

Výběr proměnných

Závislé proměnné experimentu jsou přímo určeny zadáním experimentu (otázkou). Je třeba pouze přesně definovat veličiny v nichž se budou vyjadřovat.

Nezávislé proměnné je třeba po eliminaci závislostí určit tak, aby jejich změnou bylo možno vystihnout podmínky reálného nasazení programového vybavení. Nezávisou proměnnou může být například:

- Množství dat
- Čas
- Pozornost věnovaná experimentu

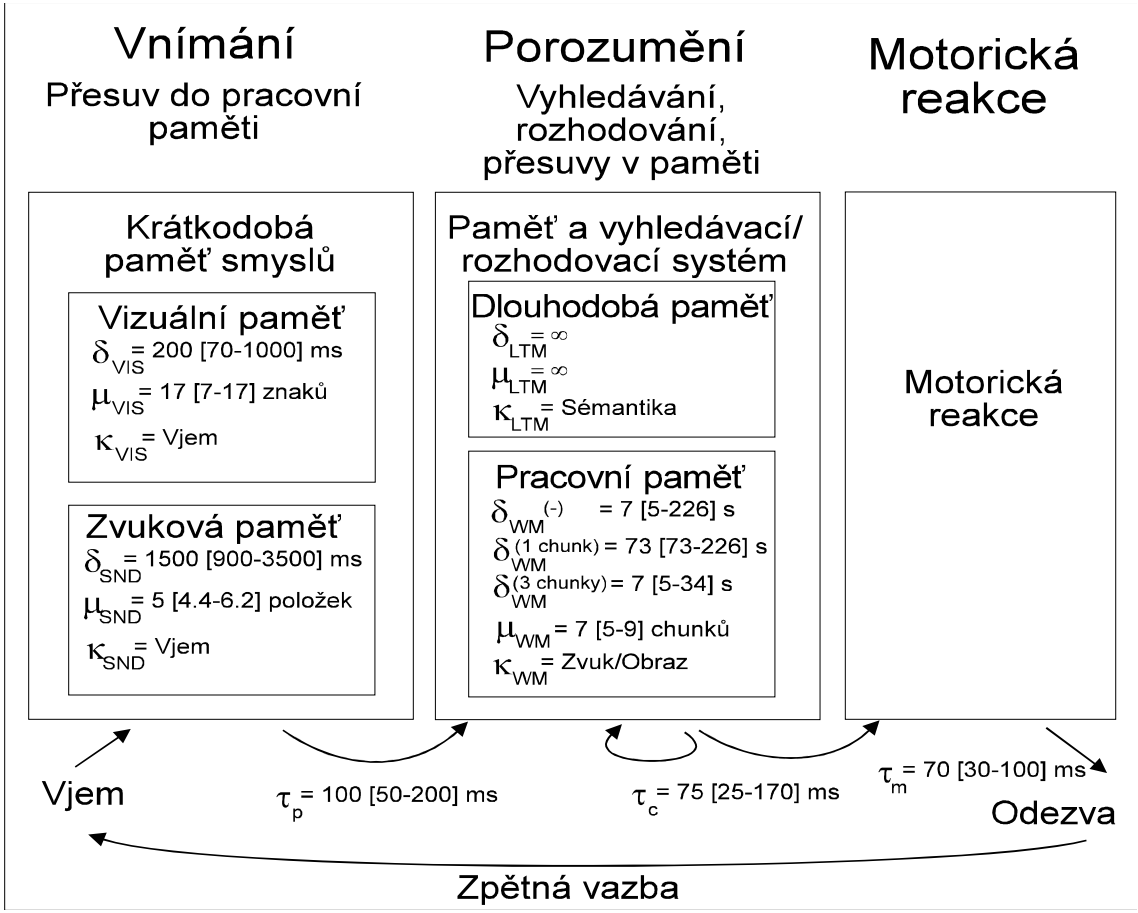
Zhodnocení realizovatelnosti experimentu

Je třeba připravit experiment dobře nejen z hlediska odborného, ale i komerčního. Z komerčního hlediska mohou nastat například následující problémy:

- Není dostupná (neexistuje) skupina uživatelů pro provedení experimentu. (Nelze najít experty na nový produkt.)
- Experiment by trval tak dlouho, že by čekání na jeho vyhodnocení bylo překážkou komerčního využití produktu.
- Experiment by byl tak drahý, že by se vůbec nevyplatilo jej dělat.

Diskrétní stavový model

- Diskrétní stavový model je syntézou experimentálně získaných dat z mnoha experimentů. (Wickensův model):



Vlastnosti – diskrétní stavový model

Aby bylo možno použít diskrétní stavový model, je třeba připustit, že jsou splněny následujících 6 předpokladů:

- Vnímání není okamžité
- Ke zpracování informace je třeba čas
- Zpracování informace je posloupností událostí v čase
- Vnitřní reprezentace informace je závislá na čase (fázi)
- Omezená kapacita zpracování v objemu i čase
- Jednotkou informace je "chunk" (balíček informací)

V praxi se diskrétní stavový model používá převážně pro následující účely:

- Analytické ověřování uživatelských rozhraní
- Zjišťování kompatibility mezi podněty a odezvami
- Vyhodnocování zátěže (pozornosti) uživatele

Rozklad na podproblémy

Interakce s počítačem se v řadě případů dá přirovnat k řešení problému. Problémy se obecně dají buď vyřešit přímo, nebo rozložit na podproblémy. Norman (1986) definoval model, podlě kterého uživatel řeší problém (interakci) v sedmi krocích:

- Stanovení cíle (přeskládat text tak, aby byl čtivější)
- Formulace plánu (přesunout odstavec 1 za odstavec 2)
- Specifikace sekvence akcí (podle typu editoru)
- Provedení akcí (podle typu editoru)
- Zjištění stavu po provedení akcí (čtení přeskládaného textu)
- Interpretace stavu (provedlo se přeskládání doopravdy?)
- Vyhodnocení splnění cíle (je text čtivější?)
- Uvedené kroky nemusejí být nutně prováděny sekvenčně

Model GOMS

Jedna z možných formalizací řešení problémů je navržena v modelu GOMS (Goals, Operators, Methods, Selection rules, Card et. al. 1983) a zdokonalena v modelu NGOMSL (Kieras 1988).

- Model řešení problmů je založen na rozkladu problému do hierarchické struktury podproblémů s takovou výškou, která odpovídá stupni rozčlenění problému.
- Model GOMS vychází z předpokladu, že je možné rozklad problému na podproblémy řešit pomocí zásobníkového automatu. Předpokládá se strategie "depth-first".
- Je-li problém na vrcholu zásobníku přímo řešitelný, aplikuje se operace "pop". V opačném případě se rozloží. Všechny podproblémy vzniklé rozkladem se uloží do zásobníku operacemi "push".
- Pokud je pravda, že každý problém (podproblém) je možno buď přímo vyřešit, nebo rozložit, dojde v konečném k vyprázdnění zásobníku (vyřešení problému).
- Modelem GOMS lze analyzovat interakci uživatele s počítačem na různých úrovních. V krajním případě až do "Keystroke-level", kde jsou "za přímo řešitelnéproblémy" považovány akce definované ve Wickensově modelu (diskrétní stavový model).

Model GOMS

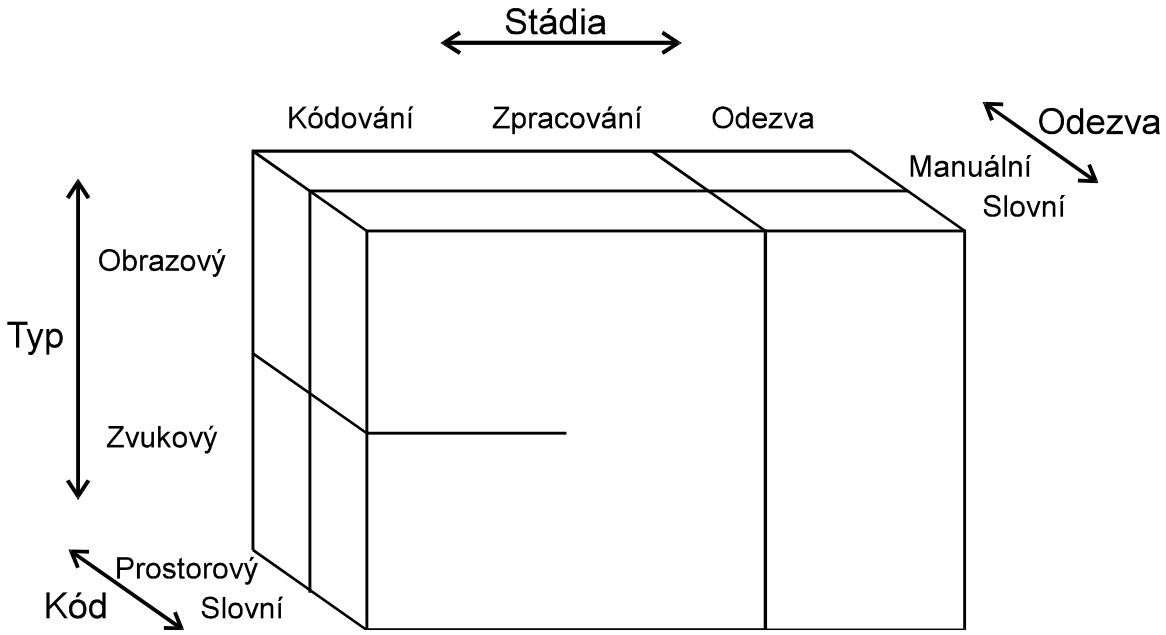
Model GOMS používá k definici stavu zásobníku:

- Cíle (Goals) - ty určují počáteční stav a obsah zásobníku
- Operátory (Operators) - akce, které jsou spojeny s operacemi "pop" na ásobníku - řešení elementárních problémů
- Metody (Methods) - reprezentují rozklady cílů do elementárních operací a do podcílů (obdoba pravidel gramatik)
- Pravidla výběru (Selection rules) - určují, která z metod (splňujících určité cíle) bude použita

Vnímání jako spotřeba zdrojů

Tento model zdůrazňuje možnost "rozdělení pozornosti". Ve starších pracech (Moray 1967) se typ zdrojů nerozlišoval. Později se ukázalo, že zdrojů je několik typů zdrojů (v nejhrubějším dělení zvuk, obraz a zpracování).

Podrobnější model je na obrázku (Winckens 1984)–SR a SCR



Iterační tvorba uživatelských rozhraní

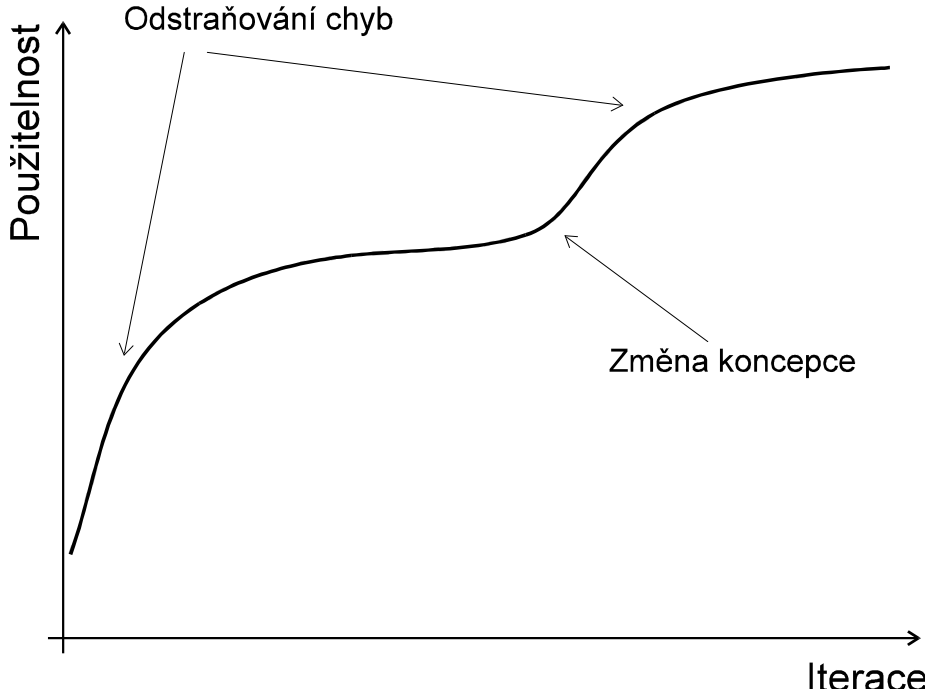
Ani nejlepší expertni nejsou schopni vytvořit optimální uživatelský interface. Proto je nutno interface odlaďovat postupně - v cyklech návrh/testování (Nielsen J: Iterative User-Interface Design, IEEE Computer, November 1993, page 32).

Za "normálních okolností" obvykle návrhář interface vymění tu část programu, která způsobuje problémy. Pokud si může vybrat z několika možností, na základě nějakých testů vybere přijatelnější variantu.

Je proto třeba používat vhodnou metodu pro měření "použitelnosti" interface. Hlavní je ovšem rychlé přizpůsobení samotných rozhraní uživatelovým potřebám.

Vývojová křivka interface

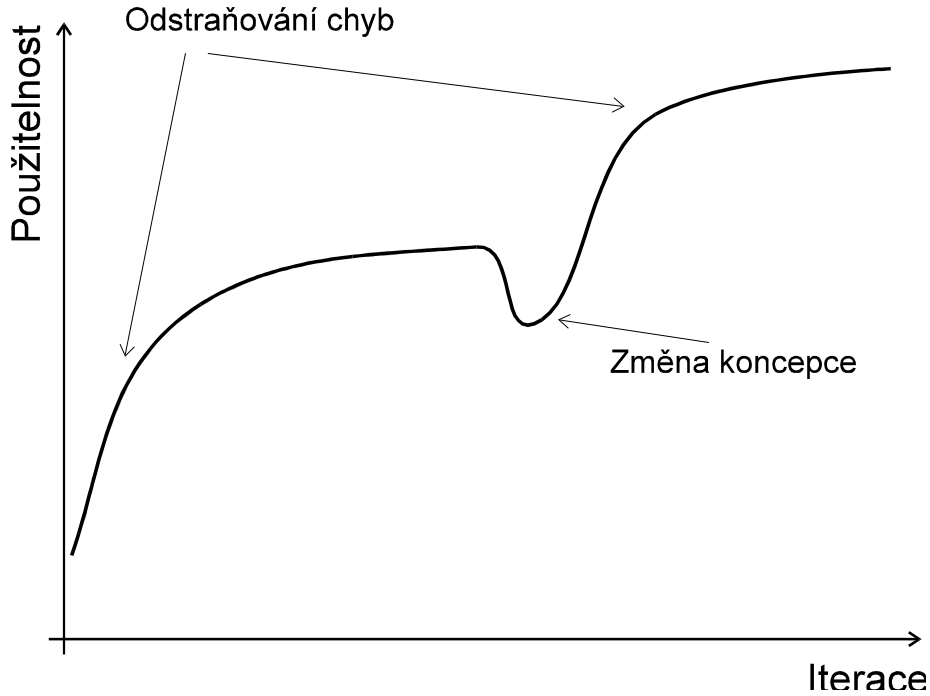
Vývojové cykly interface lze znázornit následujícím grafem:



V počátcích vývoje aplikace se jistě podaří odstranit řadu chyb a tím v nejhorším případě alespoň minimalizovat počet "katastrof". Po několika iteracích však jsou možnosti úprav vyčerpány a jediným řešením je promyslet znovu koncepci, což může vést k prudšímu nárůstu použitelnosti.

Vývojová křivka interface II

- Může ovšem snadno dojít i k dočasnému (jak návrhář jistě doufá) poklesu použitelnosti, jak je znázorněno v grafu:



- Bohužel, není známo, jak dlouho může takový proces pokračovat, neboť pojem "optimální interface" je velmi vágní. Navíc se často může v průběhu vývoje aplikace změnit částečně i cíl.
- Příkladem změny koncepce může být například vývoj programování (binární kód, assemblery, vyšší programovací jazyky, objektové prograomvání, vizuální programování, ...).

Měřítko použitelnosti

Ačkoli použitelnost byla v grafech znázorněna jako jednoduchá křivka, ve skutečnosti to není jednorozměrná veličina. Aby byl systém vysoce použitelný musí být:

- jednoducché si osvojit práci se systémem
- efektivní - musí zkušenému uživateli (expert user) dovolit dosáhnout vysoké produktivity
- snadno zapamatovatelný aby uživatelé, kteří se systémem pracují zřídka, mohli rychle "najít ztracenou nit„
- odolný proti chybám (a také by měl umět rychle odstranit následky uživatelovy chyby)
- příjemný a práce s ním musí uživatele "uspokojovat"

Pro uvedené charakteristiky lze nalézt i číselnou metriku. Problémem může být realistické vyhodnocení. Často se používá testování existujícího systému náhodně vybranými uživateli.

Priority měřítek použitelnosti

- Již v nejranějších fázích návrhu programového systému je nutno stanovit priority pro typy měřítek použitelnosti.
- Například firma Nynex (telefonní společnost obsluhující NewYork a Novou Anglii) vyčíslila, že při zkrácení doby spojování hovoru spojovatelem o pouhou jednu sekundu na hovor ušetří 3 000 000 $ ročně. Pro tyto aplikaci je tedy jednoznačně kritériem efektivita zkušeného uživatele.
- Zcela jiným případem je například situace v jaderných elektrárnách, kde je přirozeně nejvyšší důraz kladen na co nejnižší frekvenci chyb způsobených uživatelem.
- Na základě rozboru aplikace je tedy nutno zvolit vhodné váhy pro jednotlivé "rozměry" použitelnosti.

Vyčíslování použitelnosti

- Vyčíslení použitelnosti podle jednotlivých kritérií je většinou prováděno experimentálně. Analytické metody jsou prozatím daleko méně úspěšné a bohužel, například pro vyčíslení spokojenosti uživatele jich vůbec nelze použít.
- Naopak, je velmi obtížné zjišťovat například efektivitu zkušeného uživatele vhodným experimentálním způsobem. Není sice problém předložit systém uživatelům k ověření a například po jedno měsíci provést vyhodnocení, ale uvedený způsob by drasticky zpomalil vývoj a byl by finančně náročný.
- Možným způsobem, jak nahradit takové testování je použít uživatele předchozí verze systému, nebo použít vývojáře systému namísto zkušených uživatelů.

Relativní použitelnost

- Neexistuje definice použitelnosti systému, podle níž by bylo možno stanovit absolutní číselnou hodnotu použitelnosti. Proto je nutno definovat takové měřítko použitelnosti, které by alespoň postihovalo změny použitelnosti aplikace při jejím vývoji. Za počáteční stav se přitom považuje první verze systému, která má definovánu použitelnost 100.
- To je poměrně jednoduché pokud jde o číselně dobře postihnutelné "rozměry" použitelnosti, například frekvenci chyb. Udělá-li uživatel ve verzi 2 průměrně 3 chyby a ve verzi 1 (první verze systému) průměrně 4 chyby, je použitelnost verze 2 125%. Totéž platí je-li počet chyb ve verzi 1 8 chyb a ve verzi 2 6 chyb.
- Obtížnější situace nastává, jde-li například o spokojenost uživatele, kterou lze zjišťovat prakticky pouze dotazníkem. Je-li spokojenost bodována stupnicí 1-5, není obecně pravdou, že dostane-li verze 1 programu známku 1 a verze 2 známku 2, je použitelnost verze 2 200.
- V takovém případě je nutno velmi citlivě stanovit hodnocení bodovacích stupnic.

Celková použitelnst

- Použitelnost lze vyhodnocovat nejen u systému jako celku, ale i u jeho jednotlivých částí nebo podúloh.
- Celková použitelnost systému však zdaleka není dána průměrnou použitelností jednotlivých jeho částí.
- Aby bylo možno stanovit celkovou použitelnost, je nutno pro daný systém stanovit typickou úlohu (úlohy), která reprezentuje typickou činnost uživatele se systémem.
- Takových typických úloh může být v některých případech i více, zejména tehdy, může-li se systémem pracovat několik odlišných skupin uživatelů.

Příklad I – Domácí bankéř

Domácí bankéř je systém, jehož experimentální použití bylo vyhodnocováno v Dánsku. Umožňuje uživateli ze svého domácího terminílu provádět úkony jako:

- zjišťovat stav bankovního účtu, vkládat a vybírat
- převádět peníze z jednoho účtu na jiný
- převádět vklady v jedné měně do jiné měny
- speciální úlohy (dotazy na úroky apod.)

Příklad I – Domácí bankéř

Vyhodnocení efektivity typické úlohy je v následující tabulce:

Verze	Stav, v/v	Převod	Směna	Spec.	Celkem
1	199	595	245	182	1220
2	404	442	296	339	1480
3	211	329	233	317	1090
4	206	344	225	213	1088
5	206	323	231	208	967

Příklad I – Domácí bankéř

Vyhodnocení spokojenosti uživatelů při hodnocení 1-5 jako odpovědi na otázky: Jak se vám systém líbí?, Raději používáte tento systém nebo reálnou banku? (1 nejlepší, 5 nejhorší). Spolu se spokojeností jsou v tabulce počty chyb a katastrof:

Verze	Spokojenost	Chyb/uživ.	Katastrof/uživ.
1	1.92	9.2	2.56
2	1.83	4.3	1.00
3	1.78	2.5	0.44
4	1.86	1.5	0.43
5	1.67	1.5	0.17

Příklad I – Domácí bankéř

Celkové výsledky experimentu "Domácí bakéř" jsou shrnuty v následující tabulce, která shrnuje přírůstky jednotlivých parametrů v závislosti na verzích (vše v relativních číslech):

Verze	Efektivita (1/čas)	Subjekt. spokojen.	Správnost (1/chyby)	Omezení katastrof	Celkové zlepšení
1	0	0	0	0	0
2	-18	6	114	156	48
3	12	9	268	582	126
4	12	4	513	495	155
5	26	17	513	1406	242

Příklad II – Hypertextový systém

Dryhým příkladem je hypertextový systém pro technické texty. Předpokládá se, že takový systém bude používán zejména pro vyhledávání požadovaných informací. Kritickými kritérii jsou čas a přesnost vyhledání požadované informace.

Následující tabulka shrnuje absolutní výsledky:

Verze	Čas (minuty)	Přesnost (%)
1	7.6	69
2	5.4	75
3	4.3	78

Příklad II – Hypertextový systém

Druhá tabulka vyjadřuje normalizované zlepšení interface v závislosti na verzi:

Verze	Čas	Přesnost	Celkové zlepšení
1	0	0	0
2	41	9	24
3	77	13	41

Shrnutí

- Pro efektivní vyhodnocení uživatelského rozhraní je třeba stanovit metriku a metody vyhodnocení
- Je třeba vzít v úvahu dynamiku vývoje rozhraní
- Nejčastěji se užívá empirické vyhodnocení – v něm je ale třeba dbát na správnou definici a provedení úlohy