Uporabnost metafor v informatiki

Saša Kuhar, Marjan Heričko
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Smetanova ul. 17, 2000 Maribor
sasa.kuhar@uni-mb.si; marjan.hericko@uni-mb.si

Izvleček

Metafore pomagajo pri razumevanju novih in abstraktnih pojmov, zaradi česar jih zasledimo v večini informacijskih rešitev. Vse od njihove prve uporabe v grafičnih uporabniških vmesnikih so bile predmet pozitivnih in negativnih kritik. Metafore uporabljamo tudi v programskem inženirstvu, v katerem so uporabno orodje v ekstremnem in objektno orientiranem programiranju.

S pregledom literature smo ugotovili, da imajo metafore v informatiki pomembno vlogo. Raziskava se osredinja na prednosti in slabosti uporabe metafor pri zasnovi uporabniških vmesnikov. Za uporabnost uporabniškega vmesnika sta izbira primerne metafore in njena implementacija ključnega pomena, zato podamo napotke za izbiro metafor in njihovo vizualizacijo. Na podlagi rezultatov raziskave ugotovimo, da so metafore pomemben pripomoček pri prepoznavanju sistemske funkcionalnosti.

Ključne besede: metafore, uporabniški vmesniki, programsko inženirstvo, vizualizacija.

Abstract

On Usability of Metaphors in Informatics

Metaphors help to understand new and abstract concepts and have been for that reason implemented in nearly all computer systems. Since their very first appearance in graphic user interfaces and because of some inappropriate implementations, the metaphors have been a topic of diverse opinions about the benefits they supposedly produce in human-computer interaction. Metaphors have their role also in program engineering, as they are a helpful tool in extreme programming and object-oriented programming.

Literature review showed the important role metaphors play in user interfaces. In the article we show the benefits and disadvantages of metaphors used in informatics. Since the choice of a good metaphor is crucial for the success of it, we provided directions for choosing an appropriate metaphor as well as for visualizing it. With the help of a research we found out that metaphors do help users interact with the system.

Key words: metaphors, user interfaces, program engineering, visualization.

1 UVOD

Metafore se pogosto pojavljajo v verbalni komunikaciji ter v naših mislih, čeprav se jih večinoma ne zavedamo. Že v stari Grčiji so ugotovili, da so koristen pripomoček pri učenju in spoznavanju novih stvari ter konceptov. Metafore s pomočjo znanega pojma predstavijo lastnosti in značilnosti nove, še nepoznane stvari ali abstraktnega koncepta ter tako olajšajo pojmovanje in kvantificiranje abstraktnih stvari (Barr, 2003).

V informatiki se pojavlja veliko abstraktnih in marsikdaj nepoznanih stvari. Za predstavitev teh pojmov in njihovega delovanja bi bilo treba veliko napora, če ne bi poznali metafor. Že ob nastanku prvega namiznega računalnika, ki je uporabljal grafični uporabniški vmesnik, se pojavi metafora »namizje«, danes pa so metafore implementirane v vse uporabniške vmesnike. Kategoriziramo jih v tri skupine: orientacijske (gumbi naprej in nazaj, količinski drsniki, vrstice stanja), ontološke (poimenovanje in shranjevanje dokumentov, definiranje njihove velikosti) in – najpogostejše – strukturne metafore (namizje, koš, mapa, kalkulator itd).

Po začetnem navdušenju nad uporabo metafor v uporabniških vmesnikih so jih začeli načrtovalci uporabljati v prevelikem obsegu (Blackwell, 2006). Polom aplikacij, v katerih metafore niso bile primerno implementirane, je privedel do kritik njihove uporabe ter do prvih obširnejših raziskav na tem področju.

Metafore se v informatiki uporabljajo tudi pri programiranju. V nekaterih primerih se pojavljajo kot poimenovanja problemov, principov in metod v programiranju ali pa so pripomoček pri načrtovanju sistemov. V ekstremnem programiranju uporabljajo sistemske metafore, ki olajšajo komunikacijo med naročniki, vodstvom in razvijalci (Khaled & Noble, 2005).

V članku se najprej posvetimo teoriji metafor in te definiramo. Sledita razdelka o metaforah v programiranju in v uporabniških vmesnikih. Podamo napotke za izbiro primernih metafor ter za njihovo vizualizacijo. Članek sklenemo s predstavitvijo

rezultatov raziskave o prepoznavanju funkcionalnosti sistemov s pomočjo metafor.

2 KAJ SO METAFORE

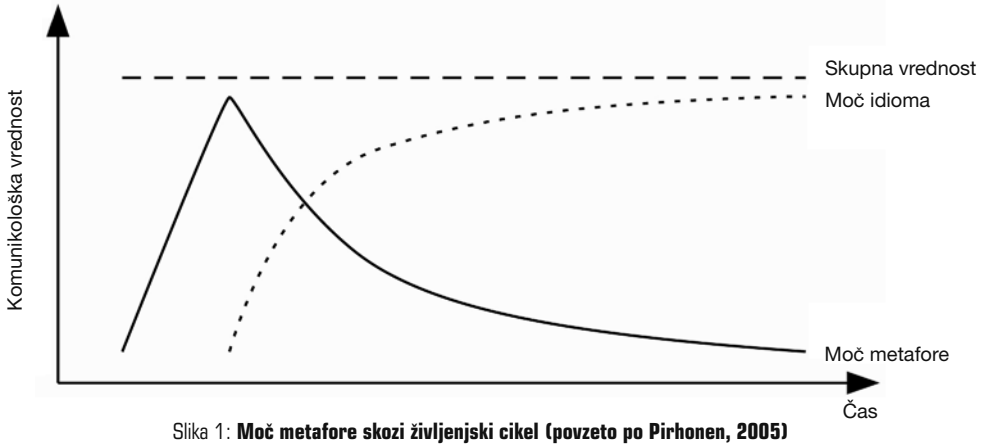
2.1 Definicije metafor

Že Aristotel je spoznal pomembnost metafor ter jih imel za ključne pri spoznavanju in učenju novih stvari in konceptov (Honeycutt, 2004). Moderne teorije (Lakoff & Johnson, 2003; Reddy, 1993), so prinesle nova spoznanja o našem mentalnem sistemu, ki naj bi bil že po naravi metaforičen: metafore ne obstajajo samo v jeziku, ampak tudi v naših mislih »na način, kako konceptualiziramo eno domeno skozi pomen druge« (Lakoff & Johnson, 2003).

Lakoff in Johnson (Lakoff & Johnson, 2003) sta metaforo razdelila na dva dela: razlagajoči del metafore je izvorna, del, ki potrebuje razlago, pa ciljna domena. Definicijo lahko podrobneje razložimo na Shakespearovi metafori »Julija je sonce«, v kateri je sonce izvorna domena, Julija pa ciljna. Julija je tako topla, svetla, središče sveta itd. Avtorja sta poudarila, da je metafora samo delna, saj se ne prenese celotnega pomena ene domene na drugo. Julija namreč ni ogromna ali žgoča, kot je sonce.

Priročnik The Oxford Companion to Philosophy (Honderich, 1995) razkrije pomembno lastnost metafor. Pravi namreč, da ljudje interpretiramo metafore ter da jih interpretiramo različno. Metafore niso interpretirane samo z vidika posameznika, ampak v povezavi z njegovim kulturnim kontekstom, v katerem lahko kultura pomeni lokacijo bivanja, religijo, izobrazbo, pripadnost socialnim skupinam itd. Ljudje z različnim kulturnim ozadjem imajo lahko različne interpretacije enakih metafor. Število ljudi, ki razume metaforo, je odvisno od stopnje znanja (prirojeno, senzorno-motorično, kulturno ali strokovno), ki je potrebno za interpretiranje določene metafore.

Metafore imajo življenjski cikel, ki ima pet stopenj: rojstvo, pronicljivost, polna moč, splošnost in smrt. Ko je metaforični izraz interpretiran prvič, se metafora rodi. S tem aktivira interpretorja (ali izvor) in omogoči vpogled v lastnosti ciljne domene. Po takšni interpretaciji ima metafora največjo moč. Kmalu prvi vtis zaradi izkušenj postopoma zbledi, metafora postane splošna in na koncu se spremeni v idiom ali splošno poznano frazo (Pirhonen, 2005).



Slika 1: **Moč metafore skozi življenjski cikel (povzeto po Pirhonen, 2005)**

Ker metafore obstajajo v naših mislih in niso omejene na jezik, lahko zavzamejo različne oblike. Metafore so lahko verbalne, vizualne, glasovne, lahko so v obliki gest itd. (Barr, 2003).

Barr (Barr, 2003) je definiral štiri ključne lastnosti metafor:

1. metafore so po definiciji napačne;
2. metafore so lahko različno interpretirane, kar je odvisno od posameznikovega znanja in izkušenj ter kulturne pripadnosti;
3. metafore naj bi bile ena izmed ključnih metod za razumevanje novih pojmov;
4. metafore ne obstajajo samo v jeziku, ampak delujejo znotraj sistema mišljenja.

Pri pregledu literature smo seznamu dodali nadaljnji dve:

5. metafore čez čas izgubijo moč in se spremenijo v idiom;
6. metafore lahko zavzamejo različne oblike, tudi vizualne.

Definicijo metafore bi lahko povzeli v stavku: Metafora je besedna figura, v kateri se s procesom enakega poimenovanja pomen ene stvari prenese na kakšno drugo stvar.

3 METAFORE V PROGRAMSKEM INŽENIRSTVU

Metafore se v programskem inženirstvu pojavljajo kot poimenovanja problemov in principov (problem trgovskega potnika, nevronske mreže, odločitvena drevesa, evolucijsko in genetsko programiranje, podatkovno rudarjenje ter skladiščenje itd.) ali pa kot metode in pripomočki v programiranju (metode v ekstremnem in objektno orientiranem programiranju).

V nadaljevanju se osredinjamo predvsem na metafore kot metode, saj je poimenovanje principov s pomočjo metafor samo jezikovni aspekt in ga ne moremo uporabiti za namen razvoja aplikacij.

3.1 Sistemske metafore

Ena izmed dvanajstih temeljnih praks v ekstremnem programiranju je sistemska metafora. To je zgodba, s katero lahko vsak (stranke, programerji in vodstvo) opiše delovanje sistema (Beck, 2005).

Sistemska metafora je sredstvo za komunikacijo v projektu, ki jo razumejo naročniki in razvijalci in ki ne zahteva vnaprejšnje priprave ali strokovnega izobraževanja. S tem ko uveljavlja skupno besedišče (ki je poznano vsem udeležencem v projektu), olajša razpravo o sistemskih problemih in rešitvah. Sistemska metafora usmerja mentalne modele udeležencev projekta, s čimer pripomore h konsistentnemu poimenovanju elementov programa in izoblikovanju logične arhitekture sistema.

Najbolj poznana sistemska metafora je »plačilni sistem je montažna linija«. Ta metafora uporablja koncepte proizvodnje (zabojniki, linije, postaje in deli) za ponazoritev plačilnega sistema. Potek plačilnega sistema razloži tako: Plača osebe je kombinacija »delov«, pri čemer se deli nanašajo na delovne ure, iz česar se izračuna osnovna plača. Deli so vstavljeni v »vhodne zabojnike«, ki se pomikajo po »montažni liniji«, po kateri so prepeljani do »postaj«. Vse postaje si sledijo v določenem »zaporedju« in »obdelujejo« posamezne dele – od osnovne plače dodajajo ali odvzemajo zneske (davek, pokojnina, nadure, članarina v sindikatu itd). Po obdelavi na posamezni postaji je del vstavljen v »izhodni zabojnik«, ki ga linija odpelje do naslednje postaje. Končna plača je tako sestavljena iz vseh izhodnih zabojnikov, ki so nastali v procesih obdelav na liniji. Ta metafora je tipičen primer za poenostavitev razlage delovanja programa ter funkcij, ki jih mora vsebovati program (Khaled & Noble, 2005).

3.2 Usmerjevalne metafore

Nekateri avtorji ponujajo metaforo za pomoč v objektno orientiranem programiranju. Usmerjevalna metafora, kot jo poimenuje Heinz Züllighoven (Züllighoven, 2005), služi za razlago problema, kako ustvariti določen sistem ter kakšna podpora je potrebna ob tem. Metafora pomaga pri osnovni orientaciji vseh udeležencev v vsem procesu razvoja, podpira oblikovanje, uporabo ter ovrednotenje programa, lahko pa pomaga tudi pri učenju uporabnikov.

Züllighoven poda štiri primere metafor, ki jih lahko uporabimo v programskem inženirstvu, in jih aplicira na elemente razvoja (tabela 1).

Tabela 1: **Uporaba usmerjevalnih metafor v programskem inženirstvu (Züllighoven, 2005)**

Usmerjalna metafora	Oblikovalski cilj	Vloga uporabnikov	Vloga razvijalcev
Objektni svetovi	Virtualni svetovi, naseljeni z aktivnimi objekti (agenti)	Sprožilci začetnih navodil	Oblikovalci minisvetov
Neposredna manipulacija	Manipulacijski objekti	Neodvisni delavci	Konstrukcija artefaktov
Tovarna	Vodenje in upravljanje z delom kot na montažni liniji	Upravljavci strojev	Izdelovalci strojev
Delovno okolje strokovnjakov	Podpora strokovnjakom z razvojem primerne opreme	Strokovnjaki	Oblikovalci orodij in okolja

3.3 Metafore v programskih knjižnicah

Znanje o določenem programskem jeziku je odvisno od razumevanja in aplikacije standardnih komponentnih knjižnic. Alan F. Blackwell (Blackwell, 1995) predlaga, da bi spoznanja iz kognitivne psihologije prenesli na komunikacijo med razvijalci jezika in programerji – specifično na knjižnice.

Blackwell je sistematsko preučeval metafore (kot temelje mentalnih modelov) v odnosu z javanskimi knjižnicami. Osredinil se je na besednjak, ki so ga

uporabljali avtorji javanskih dokumentov, s čimer je identificiral mogoče kandidate za metafore. Nekatere izmed metafor, ki jih je identificiral Blackwell, so »programi delujejo v preteklosti«, »programsko stanje je lahko kvantificirano«, »komponente so člani družbe«, »metode klicanja so dejanja govora«, »komponente imajo prepričanja in namene«, »programi operirajo v prostorskem svetu, ki ima notranje in zunanje okolje«.

Podobno je raziskoval Nikolas S. Boyd (Boyd, 2003), ki je preučeval metafore v programski kodi. Ugotovil je, da so metafore zmeraj prisotne, ne samo v programskih jezikih, ampak na splošno v besednjaku razvijalcev, kadar govorijo o sistemih.

4 METAFORE V UPORABNIŠKIH VMESNIKIH

Metafore imajo znotraj uporabniških vmesnikov nekoliko drugačne značilnosti kot v teoretičnem, jezikovnem smislu, vendar so osnovne značilnosti enake. Najbolj očitna vloga metafore v uporabniških vmesnikih je podobna kot pri splošni metafori: predstavljanje neke druge stvari; metafora je sredstvo za razlago ene stvari, s tem ko jo identificira z drugo.

Islovar (Koren, 2010) navaja definicijo metafore vmesnika tako: je »metafora, ki jo simulira vmesnik s prikazovanjem uporabniku že znanega z drugega področja in tako olajša delo«. Izvorna domena je običajno stvar iz realnega sveta, njene karakteristike pa opisujejo sistemsko funkcionalnost. Osrednja prednost metafor v uporabniških vmesnikih je, da uporabniku pomagajo ustvariti pričakovanja in spodbujajo predstave o obnašanju sistema.

Metafore v uporabniških vmesnikih lahko predstavimo s pomočjo različnih čutov in ne samo v jezikovnem smislu. Računalniške metafore so lahko slikovne (ikona mape) ali zvočne upodobitve (zvonjenje ob klicu), s pomočjo miške uporabljamo kinestetične metafore (vleci in spusti, angl. drag and drop), novejši uporabniški vmesniki in vnosne naprave, ki omogočajo upravljanje z večprstnim dotikanjem (angl. multi-touch), pa tudi izkoriščajo tip in geste (povečaj ali zavrti, angl. zoom in rotate) (Barr, 2003).

Predpostavka, da so metafore interpretirane, je z vidika uporabniških vmesnikov precej pomembna. Ko se namreč uporabniki srečajo z metaforo, si jo razložijo in interpretirajo znotraj svojega individualnega konteksta. Človeški miselni kontekst je pogojen z izkušnjami, s kulturo, z izobrazbo, s statusom ter z drugimi značilnostmi, ki izoblikujejo posameznika v specifično osebo. Na to morajo biti pozorni načrtovalci uporabniških metafor, saj posamezne uporabniške vmesnike uporablja veliko število ljudi, kar lahko posledično pripelje do različnih ali napačnih interpretacij (Barr, 2003).

4.1 Tipi metafor v uporabniških vmesnikih

V uporabniških vmesnikih najdemo vse tri tipe metafor, kot sta jih definirala (Lakoff & Johnson, 2003): orientacijske, ontološke in strukturne.

4.1.1 Orientacijske metafore

Uporabniški vmesniki obstajajo v dvoinpoldimenzionalnih zaslonih (okna se prekrivajo), zato lahko razvijalci s pridom uporabljajo orientacijske metafore. Orientacijske metafore služijo v sistemu dvema namenoma: navigacija in kvantifikacija.

V programski opremi se uporabnik večkrat sreča z navigiranjem po vsebini. Velikokrat se pojavita gumba »naprej« in »nazaj»« (angl. next in back), ki sta postavljena drug zraven drugega. Običajno imata vsak svojo manjšo puščico, ki vizualno nakaže smer. Gumba »naprej« in »nazaj« sta bila za potrebe dvodimenzionalnega prostora preoblikovana v »levo« in »desno« (slika 2).

Količinska opredelitev je v interakciji človeka z računalnikom pomemben koncept, zato je tudi vizualna ponazoritev količine, ki je predstavljena večinoma metaforično, pogost pojav. Eden izmed najpogostejših primerov količinske ponazoritve je drsnik (angl. slider). Vertikalni drsniki delujejo po principu »več je gor«, horizontalni drsniki pa so orientirani levo–desno, pri čemer desno pomeni več (slika 3). Razloge za takšno orientiranost lahko najdemo v preoblikovanju osnovne metafore »več je gor« za potrebe drugih dimenzij.

Orientiranost se navezuje na pozicijo posameznika na časovnem traku. Uporabniki si ob poteku določene operacije želimo informacije o tem, kje se trenutno nahajamo. Vmesniki zato s pomočjo vrstic napredka (angl. progress bar) ponudijo informacijo o časovnem poteku, ki deluje kot časovni trak, na katerem je označeno trenutno stanje.

Slika 2: **Gumba Nazaj in Naprej v operacijskem sistemu Kubuntu**

Slika 3: **Drsnik za prilagoditev glasnosti v programu iTunes**

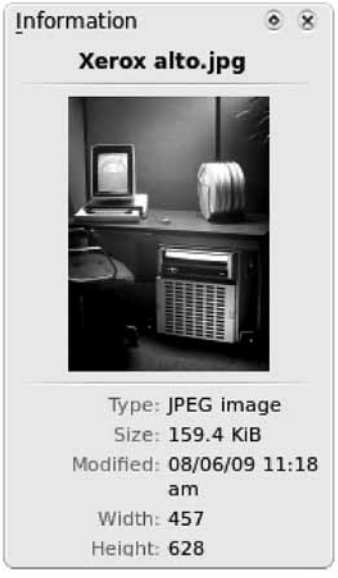
4.1.2 Ontološke metafore

Sistemske aplikacije so večinoma neoprijemljivi abstraktni koncepti, zato so lahko ontološke metafore v uporabniških vmesnikih zelo uporabne.

Eden izmed namenov ontoloških metafor je nanašanje na nekaj. S tem ko prepoznamo sistemske koncepte kot entitete in objekte, lahko uporabnikom bolj preprosto predstavimo njihov obstoj in značilnosti. Dokumenti imajo tako imena, nahajajo se na določeni lokaciji, definiramo jim lahko velikost, uporabniki jih lahko kopirajo, prilepijo in prestavijo v koš (slika 4).

Pomemben del uporabniških vmesnikov je obveščanje uporabnika o procesih, pri čemer je običajno vključena tudi pojasnitev vzroka programskih dogodkov. V računalniških okoljih dojemamo koncepte kot resnične fizične objekte in entitete, ki nam kot taki dobro služijo za obrazložitev vzrokov. Programi lahko tako vsebujejo napake, sistem pa nam lahko ponudi razlago vzrokov različnih operacij.

Metafore vsebnikov so v uporabniških vmesnikih prav tako precej pogoste. Za primer lahko podamo »vnosno polje« (angl. text-box), »v« katerega uporabnik vpiše besedilo, nato pa polje »shrani« vpisano besedilo.



Slika 4: **Prikaz lastnosti slike v operacijskem sistemu Ubuntu**

4.1.3 Strukturne metafore

Kadar govorimo o metaforah v uporabniških vmesnikih, imamo običajno v mislih strukturne metafore. Strukturne metafore so znotraj uporabniških vmesnikov ustvarjene iz koncepta fizičnih stvari, pri čemer se del strukture realne stvari uporabi za lažje razumevanje abstraktnega programskega koncepta.

Primer računalniške metafore, ki je zlahka prepoznan objekt iz realnega sveta, je »koš za smeti« (angl. trashcan ali trash). Koš pooseblja strukturno metaforo, ko »datoteko izbrišemo tako, da jo vržemo v koš«.

Čeprav so strukturne metafore najbolj podobne realnim stvarem, pa vseh konceptov in atributov realne stvari vendarle ne moremo prenesti na abstraktni sistemski element. Računalniški koš tako nima vseh atributov fizičnega koša: ne more se npr. prevrniti (slika 5). Prav tako pa realni koncepti nimajo vseh lastnosti, ki jih imajo metafore v vmesnikih. V Applovem operacijskem sistemu Mac OS lahko uporabnik zunanji pomnilnik izvrže tako, da ikono pomnilnika premakne v koš. Takšne ali podobne funkcije nima resničen koš.

Ker so strukturne metafore najbolj specifične pri predstavljanju novih konceptov, so lahko posledično tudi najbolj občutljive za kulturne razlike.

4.2 Kritike uporabe metafor v uporabniških vmesnikih

Kritike metafor so se začele pojavljati že ob njihovi prvi uporabi v uporabniških vmesnikih. V nadaljevanju navajamo pet temeljnih kritik uporabe metafor v uporabniških vmesnikih, nato pa povzemamo še mnenja zagovornikov metafor.

1. Metafore uporabniških vmesnikov ne morejo nikoli prenesti celotnega pomena koncepta realnega sveta na sistemski koncept. To lahko povzroči nepravilno dojemanje funkcionalnosti, saj lahko uporabniki pričakujejo, da so abstraktni objekti enako zmogljivi kot realni. Metafore so torej zavajajoče (Saffer, 2005).
2. Ciljna domena lahko vsebuje več funkcionalnosti, kot jih ima izvorna domena. Posledično lahko uporabniki spregledajo nekatere značilnosti vmesnika, saj jih izbrana metafora morda ne sporoča (Carroll, 1984).
3. Metafore se ne morejo razširiti ali zmanjšati. Čeprav so lahko trenutne funkcije aplikacije smiselne znotraj določene metafore, se lahko zgodi, da morebitne nove funkcije ne bodo več (Neale & Carroll, 1997).

4. Po določenem času metafore zgubijo svojo moč in se spremenijo v idiom.
5. Metafore so v sisteme implementirane prepogosto. Čeprav morda olajšajo delo novim uporabnikom, pa lahko ovirajo napredne uporabnike (Neale & Carroll, 1997).

Zagovorniki metafor v uporabniških vmesnikih (Lakoff & Johnson, 2003; Reddy, 1993) pravijo, da so metafore del našega mentalnega sistema in bi zato brez njih težko izražali naše (abstraktne) misli. Računalniki so večinoma abstraktni sistemi, zato so metafore naravno orodje pri razlagi sistemskih značilnostih.

Če ne bi metafor v vmesnike implementirali že razvijalci, lahko predvidevamo, da bi si jih za razlago sistemskih značilnosti izmislili uporabniki sami. Vprašanje je, ali bi bile metafore uporabnikov enako primerne, kot so metafore razvijalcev. Uporabniki običajno namreč niso strokovnjaki na področju razvoja uporabniških vmesnikov (Cooper, 1995).

Alty (Alty, Knott, Anderson, & Smyth, 2000) pretvori kritiko, ki zavrača metafore zaradi neenakosti funkcij in lastnosti med izvorom in ciljem, v pripomoček pri ugotavljanju primernosti metafor. Avtor je na primeru definiral, koliko se lastnosti izvorne domene prenesejo na ciljno domeno ter koliko se lastnosti cilja prekrivajo z lastnostmi izvora. S tem je definiral tri kazalnike: 1) v kolikšnem obsegu bo metafora povzročala nepravilno sklepanje o sistemski funkcionalnosti, 2) koliko funkcionalnost programa sovpada z lastnostmi metafore ter 3) v kolikšnem obsegu se funkcionalnost programa izraža v metafori.

Pirhonen (Pirhonen, 2005) trdi, da idiomi niso problem. Mrtve metafore lahko namreč koristijo novim uporabnikom (ko jim ponudijo znanje o abstraktnih konceptih), napredni uporabniki pa jih lahko uporabljajo kot idiome (ki jih ne ovirajo eksplicitno).

Kljub kritikam metafore še zmeraj veljajo za uporabno orodje v interakciji človeka z računalnikom in so zaradi tega implementirane v večino vmesnikov.

5 IZBIRA IN VIZUALIZACIJA METAFOR

5.1 Napotki za izbiro primernih metafor

Veliko avtorjev ponuja napotke za izbiro primernih metafor v uporabniških vmesnikih. Kot je omenjeno v tretjem razdelku, se metafore ne nanašajo samo na vizualne objekte v uporabniških vmesnikih, ampak razkrivajo tudi funkcionalnosti in delovanje sistema. Zato lahko napotke, ki jih povzemamo v nadaljevanju, uporabimo tudi v programskem inženirstvu.

Neale in Carrol (Neale & Carroll, 1997) priporočata tri korake za izbiro metafore, in sicer:
1. razmislek o mogočih kandidatih za metaforo (analiza obstoječih programov, opazovanje uporabnikov ali tvorjenje nove metafore),
2. opredelitev primernosti kandidatov za metafore (definiranje neujemanj med izvorno in ciljno domeno),
3. razrešitev neujemanj.

Erickson (Erickson, 1995) priporoča drugačen pristop, s katerim lahko razvijalci poiščejo primerno metaforo za vsak del sistema, v katerem se pojavi problem v funkcionalnosti. Za ugotavljanje primernosti ponudi definiranje petih atributov: količine strukture, aplikativnosti strukture, reprezentativnosti, uporabniške primernosti in razširljivosti.

Alty (Alty et al., 2000) predlaga šest korakov za načrtovanje uporabniških vmesnikov:
1. identifikacija sistemske funkcionalnosti,
2. tvorjenje in opisovanje potencialnih metafor,
3. analiza parov metafora–sistem,
4. implementacija,
5. vrednotenje,
6. pridobivanje povratnih informacij uporabnikov.

Za drugi korak (tvorjenje in opisovanje potencialnih metafor) predlaga avtor več tehnik: oblikovanje novih metafor, razširjanje obstoječih metafor, viharjenje možganov (angl. brainstorming), analizo trga in študijo delovnih mest.

Madsen (Madsen, 1994) ponudi za oblikovanje primernih metafor zbirko hevrističnih metod. Za generiranje metafor predlaga opazovanje uporabnikov, nadgradnjo obstoječih metafor, uporabo predhodnika artefakta ali iskanje dogodkov v realnem svetu, ki izražajo ključne aspekte funkcionalnosti. V procesu vrednotenja naj bi razvijalci poiskali metaforo z bogato strukturo (izogibati se je treba preveč splošnih in abstraktnih metafor), ovrednotili aplikativnost strukture (izbrana metafora mora ponujati relevantne rešitve za problem) ter izbrali tisto metaforo, ki ima široko poznan literarni pomen in je primerna za občinstvo. Metafora mora imeti konceptualno razdaljo med izvorno in ciljno domeno, vendar prav tako najmanj en povezovalni koncept. Med razvojem sistema predlaga avtor uporabo na metafori temelječega ključnega vidika, prestrukturiranje percepcije realnosti, tvorjenje predvidevanj, prepoznavanje neizkoriščenih delov

metafore, generiranje nasprotujočih si delov metafore in pripovedovanje zgodbe metafore.

5.1 Vizualizacija metafor

Metafore v programskem inženirstvu v veliki večini obstajajo samo kot jezikovni pripomočki za razlago konceptov, zato se ta razdelek nanaša predvsem na metafore v uporabniških vmesnikih.

Za uspeh metafore je pomembna njena sposobnost predstavitve objekta, ki prenaša informacijo do uporabnika. Pri načrtovanju metafore je treba pretehtati vse možnosti predstavitve določenega koncepta: grafični element, animacijo, zvok, interakcijo itd.

Pomembno je, da uporabnik metaforo prepozna čim hitreje, saj je to prvi korak pri spoznavanju funkcionalnosti programa, ki naj bi ga izražala metafora. Pri vizualizaciji metafor so pomembni predvsem trije dejavniki: realističnost, konsistentnost ter primernost (Alty et al., 2000).

5.1.1 Realističnost metafor

Uporabnik naj bi kar najhitreje prepoznal objekt, ki ga izraža metafora, zato bi bilo smiselno metaforo upodobiti čim bolj realistično. Tako bi naj bila uporabnikova obstoječa predstava o objektu iz realnega sveta zlahka prenesena na računalniški koncept. Vendar pa takšno sklepanje ni vedno pravilno. Če se metafora prikaže kot dobesedna stvar iz realnega sveta, morebitne nove funkcije računalniškega koncepta niso takoj vidne. Uporabnik lahko dobi občutek, da sistem deluje v nasprotju z intuicijo.

Primer neprimerne realističnosti metafore se je pojavil v sistemu ROME, v katerem je bila za funkcionalnost shranjevanja dokumentov izbrana metafora kovček. Uporabniki so lahko v kovčku hranili svoje dokumente ali jih javno delili z drugimi uporabniki v sobi. Kovček je bil v sobi (ob prisotnosti drugih uporabnikov) odprt, zaradi česar so uporabniki dvomili v zasebnost svojih dokumentov. Odprt kovček je nakazoval, da so datoteke v njem vidne vsem prisotnim, kar pa se ni skladalo s funkcionalnostjo sistema.

Pri izbiri stopnje realističnosti vizualne metafore je zato treba upoštevati prekrivanje izvorne in ciljne domene ter se zavedati, da bolj realističen videz metafore ni vedno najboljša izbira.

5.1.2 Konsistentnost metafor

Poznamo dva tipa konsistentnosti metafor: znotraj metafor in med metaforami.

Konsistentnost znotraj metafore se nanaša na konsistentnost posamezne metafore. Enake metafore v različnih vmesnikih naj vedno težijo k podobnosti, saj imajo uporabniki že izdelano predstavo o delovanju specifične metafore, ki so jo pridobili iz izkušenj delovanja v drugih vmesnikih.

Znotraj posameznega vmesnika je običajno aktivnih več metafor hkrati. Večina vmesnikov, ki upravlja z besedilom, podpira uporabo metafor »izreži« (angl. cut), »kopiraj« (angl. copy) in »prilepi« (angl. past). Ko uporabnik kopira besedilo iz enega dokumenta, pričakuje, da bo drugi vmesnik podpiral funkcijo »prilepi« in da bo tako lahko prenesel besedilo v drugi dokument. Konsistentnost je treba vzdrževati med različnimi metaforami in različnimi vmesniki (Alty et al., 2000).

5.1.3 Primernost metafor

Čeprav izbrana metafora pravilno izraža funkcionalnost programa, je lahko neprimerna. Neprimerna metafora vsebuje sporno konotacijo, kar lahko povzroči neustreznost. Najbolj očitne so rasne, spolne ali starostne konotacije, obstajajo pa še druge.

Metafora, ki je stilistično podobna otroški risanki, se lahko na prvi pogled zdi primerna, vendar meče slabo luč na razvijalsko podjetje in na uporabnike programa.

Pri izbiri metafore je treba upoštevati tudi dejstvo, da metafora, ki je primerna danes, v prihodnosti mogoče ne bo več (Alty et al., 2000).

6 PREPOZNAVANJE FUNKCIONALNOSTI SISTEMA S POMOČJO METAFOR

6.1 Opredelitev problema

V članku smo večkrat zapisali, da metafore vsebujejo informacije o sistemskih funkcionalnostih in so zaradi tega predvidoma koristen pripomoček v interakciji človeka z računalnikom. Proti metaforam je bilo podanih tudi nekaj kritik (povzeli smo jih v razdelku 4.2), zato smo s svojo raziskavo želeli ugotoviti, ali metafore dejansko pomagajo pri prepoznavanju sistemskih funkcionalnosti.

6.2 Cilji in namen raziskave

Z raziskavo smo skušali ugotoviti, kako dobro uporabniki prepoznavajo funkcionalnost sistema s pomočjo metafor v uporabniških vmesnikih. Zanimalo nas je, ali so metafore same po sebi dovolj dober pripomoček za prepoznavanje funkcionalnosti sistema

oz. ali je za prepoznavanje potrebna dodatna pomoč. Ugotoviti smo tudi želeli, ali uporabniki prepoznajo metafore, ki se v uporabniških vmesnikih ne pojavljajo pogosto oz. s pomočjo katerih metafor uporabniki največkrat prepoznajo funkcionalnost sistema.

6.3 Potek raziskave

Za namen raziskave smo izvedli anketo o prepoznavanju sistemskih funkcionalnosti s pomočjo metafor. Uporabili smo vprašalnik s slikami 21 metafor iz različnih aplikacij in operacijskih sistemov. Metafore, izbrane za anketo (slika 5), predstavljajo različne stopnje pogostosti pojavljanja v sistemih.

V prvi fazi so anketiranci navedli funkcionalnost sistema, ki jo po njihovem mnenju izraža določena metafora. V drugi fazi so opredelili, koliko se strinjajo s podanimi odgovori o funkcionalnosti sistema, ki jo izraža določena metafora (po Likertovi lestvici od 1 – se ne strinjam do 5 – zelo se strinjam). Na koncu so sledila še splošna vprašanja o izkušnjah anketirancev v različnih operacijskih sistemih in programih ter z razvojem informacijskih rešitev.

Slika 5: **Metafore, prikazane v anketnem vprašalniku (* = metafora se pojavi v spletnem brskalniku)**

Anketirali smo 49 uporabnikov, od katerih dva nista v celoti izpolnila ankete, zato smo izključili njune odgovore. Anketiranci so bili stari 19 do 24 let in so predstavniki generacije, ki je dobro seznanjena z različnimi uporabniškimi vmesniki. Od 47 anketirancev se jih 28 ukvarja z razvojem programske opreme (v povprečju 2,7 leta).

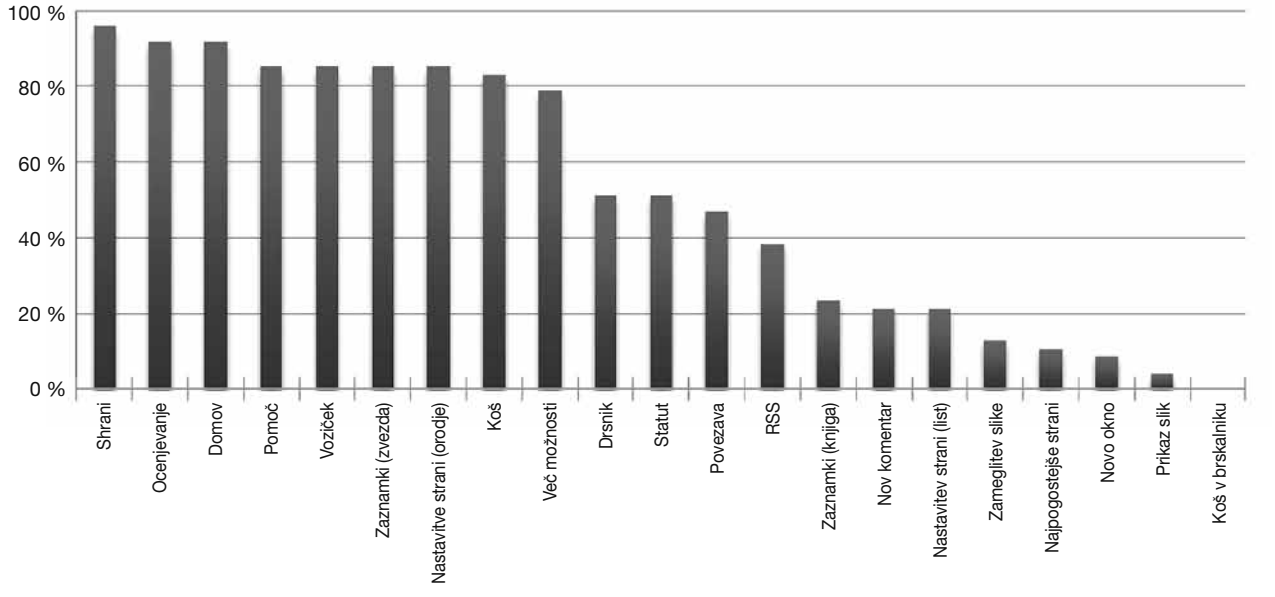
Dve metafori (drsnik in zameglitev slike) smo anketirancem najprej pokazali brez sistemske pomoči, nato pa z njo. Metafora drsnik je v uporabniškem vmesniku Microsoft Expression Design upodobljena tako, da na prvi pogled izgleda kot prikazovalnik količine. Šele ko se uporabnik z miško premakne nad polje, se kurzor spremeni in uporabnik ugotovi, da je količina nastavljiva s premikom miške. Pri metafori zameglitev slike je v drugem primeru prikazan namig, ki se pojavi ob postavitvi kurzorja na orodje.

Metafore anketirancem niso bile prikazane znotraj njihovega običajnega okolja, zato pomoč, ki jo običajno ponujajo sistemi, ni bila zajeta v raziskavi (razen v drugem prikazu drsnika in orodja za zameglitev).

Odgovorov, v katerih je bilo navedeno ime metafore (npr. koš), nismo šteli za pravilne, saj niso navajali funkcionalnosti (npr. brisanje datotek).

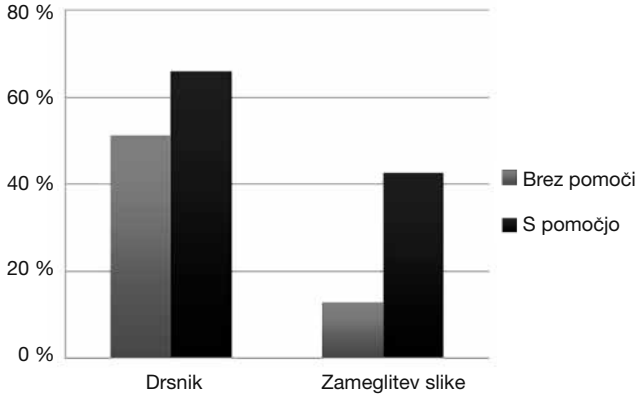
6.4 Ugotovitve

Z anketo smo ugotovili, da je v povprečju 53 odstotkov anketirancev prepoznalo funkcionalnost sistema s pomočjo metafor. Kot lahko opazimo na sliki 6, je večina anketirancev prepoznala funkcionalnost s pomočjo metafor, ki se v uporabniških vmesnikih pojavljajo pogosto (shrani, ocenjevanje, domov, pomoč, voziček itd.). Nizko prepoznavnost lahko opazimo pri metaforah, ki se pojavljajo redko in v specifičnih programih (koš v brskalniku, prikaz slik v brskalniku, novo okno, najpogostejše strani ter zameglitev slike).

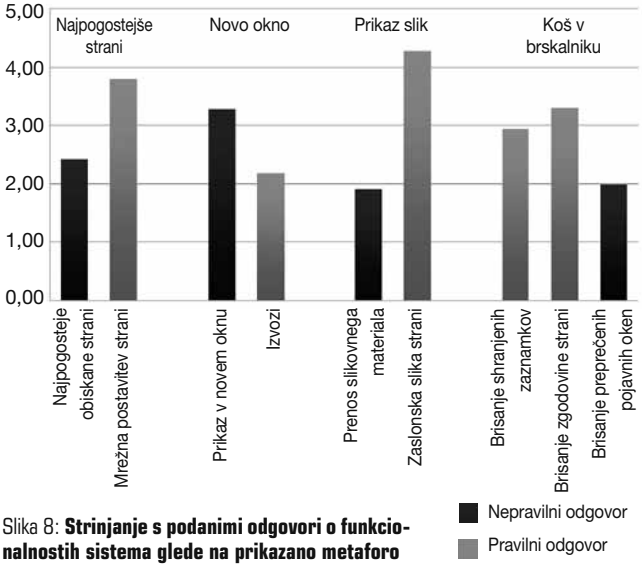


Slika 6: **Uspešnost anketirancev pri prepoznavanju funkcionalnosti sistema s pomočjo metafor**

S pomočjo ankete smo ugotovili, da sistemska pomoč izboljša prepoznavanje funkcionalnosti s pomočjo metafor (slika 7). Uporabniki so bili pri prepoznavanju drsnika in orodja za zameglitev slike uspešnejši, kadar je bila prikazana vizualna pomoč.



Slika 7: **Uspešnost anketirancev pri prepoznavanju funkcionalnosti sistema s pomočjo metafor z in brez sistemske pomoči**



Slika 8: **Strinjanje s podanimi odgovori o funkcionalnostih sistema glede na prikazano metaforo**

Štiri metafore, ki so se v prvi fazi najslabše odrezale pri prepoznavanju, so bile slabo prepoznane tudi v drugi fazi, ko so anketiranci navajali stopnjo strinjanja s podanimi funkcionalnostmi (slika 8). Anketiranci so se v večji meri strinjali z napačnimi odgovori predvsem pri metaforah najpogostejše strani (ki jih je spominjala na mrežno postavitev strani) in prenos slik (pri kateri so pomislili na zajemanje zaslonske slike) ter pri metafori koš v brskalniku (za katerega so menili, da izraža brisanje shranjenih zaznamkov ali brisanje zgodovine strani).

V tabeli 2 so prikazani rezultati prepoznavanja dodatne funkcionalnosti koša v operacijskem sistemu Mac OS. Samo štirje anketiranci so se strinjali oz. zelo strinjali s funkcionalnostjo izvrzi zunanji pomnilnik, četudi je enajst anketirancev v preteklosti že uporabljalo Mac OS, trije pa ga uporabljajo tudi sedaj. Prav tako ni zaznati povezave med uporabniki, ki so že uporabljali Mac OS, in strinjanjem s prepoznavanjem funkcionalnosti.

Tabela 2: **Prepoznavanje dodatne funkcionalnosti koša v operacijskem sistemu Mac OS**

Strinjanje s funkcionalnostjo: izvrzi zunanji pomnilnik	Uporabniki, ki so že uporabljali Mac OS	Uporabniki, ki še niso uporabljali Mac OS	Uporabniki, ki uporabljajo Mac OS	Uporabniki, ki ne uporabljajo Mac OS
Se ne strinjam – 1	9	30	2	37
Delno se ne strinjam – 2	1	2	1	2
Sem neopredeljen – 3	1	0	0	1
Se strinjam – 4	0	2	0	2
Zelo se strinjam – 5	0	2	0	2

6.5 Povzetek raziskave

Anketiranci so večinoma zlahka prepoznali metafore, ki se v vmesnikih pojavljajo pogosto, čeprav jim je bila onemogočena običajna sistemska pomoč. Sklepamo lahko, da poznane metafore uporabnikom olajšajo delo z uporabniškimi vmesniki, saj uporabniki zlahka prepoznajo funkcionalnosti.

Uporabniki so prepoznali manj funkcionalnosti s pomočjo metafor, ki se pojavljajo v specifičnih uporabniških vmesnikih (Adobe Photoshop), imajo neobičajne oblike (drsnik v orodju Microsoft Expression Design), neobičajne izvorne pomene (knjiga za zaznamke in list za nastavitve brskalnika) ali neobičajne pomene (izvrzi zunanji pomnilnik s premikom ikone na koš v operacijskem sistemu Mac OS).

Ugotovili smo, da nekatere metafore izražajo drugačne funkcionalnosti, kot naj bi jih (najpogostejše strani, prikaz slik ter koš v brskalniku). S tem ko

uporabnike spominjajo na napačno funkcionalnost, jih zavajajo in tako otežujejo interakcijo z računalniki. Zato je nadvse pomembno, da razvijalci uporabniških vmesnikov preudarjeno uporabljajo metafore ter stremijo k izbiri najprimernejše.

Kadar je bila uporabnikom ob novih oz. neobičajnih metaforah prikazana pomoč, je bilo prepoznavanje funkcionalnosti višje. Tako se lahko – predvsem ob vpeljavi nove ali drugačne metafore – razvijalci zanašajo na sistemsko pomoč, ki uporabnikom olajša prepoznavanje.

Iz raziskave lahko povzamemo, da pogoste in poznane metafore uporabnikom olajšajo delo, saj lahko brez sistemske pomoči prepoznajo funkcionalnosti. Kadar se uporabniki srečajo z novo metaforo, jih ta ne sme zavajati. Zato morajo razvijalci sistemov dobro premisliti, kaj izraža izbrana metafora in ali se sklada s funkcionalnostjo sistema.

7 SKLEP

Čeprav se neprestano srečujemo z metaforami, se jih velikokrat ne zavedamo, zato jih v razvoju uporabniških vmesnikov ne bi smeli ignorirati. Nekatere teorije poudarjajo njihov doprinos k interakciji človeka z računalnikom, kar smo delno potrdili tudi z raziskavo. Pogoste metafore namreč omogočajo prepoznavanje funkcionalnosti, kadar sistemska pomoč ni na voljo. Metafore, ki so konsistentno uporabljene med različnimi vmesniki, lahko pripomorejo pri interakciji uporabnikov s sistemi.

Prepoznavanje novih metafor je prav tako mogoče, vendar morajo biti te skrbno izbrane. Samo če dobro komunicirajo sistemsko funkcionalnost, so lahko v pomoč uporabnikom.

Običajno se ob metaforah nahajajo poimenovanja funkcionalnosti, večina aplikacij pa vsebuje tudi pomoč, s katero se lahko uporabnik kadar koli posvetuje. Dokazali smo, da je pomoč primerna še posebno pri neznanih ali preoblikovanih metaforah, saj olajša prepoznavanje. Ob redni uporabi pa dobi metafora polno moč in takrat jo lahko uporabniki s pridom izkoriščajo.

Nadaljnje raziskave na področju metafor bi se med drugim morale osrediniti na vpeljavo novih metafor – od ideje in vizualne podobe do same implementacije – ter raziskati, koliko in kdaj naj bodo metafore realistične, kaj naj izražajo ter kako testirati njihovo primernost.

8 VIRI, LITERATURA

[1] Alty, J. L., Knott, R. P., Anderson, B. & Smyth, M. (2000). A framework for engineering metaphor at the user interface. *Interacting with Computers*, *13*(2), 301–322. Elsevier.

[2] Barr, P. (2003). User-Interface Metaphors in Theory and Practice. Wellington.

[3] Beck, K. (2005). *Extreme Programming Explained*: Embrace Change. Boston: Addison-Wesley Professional.

[4] Blackwell, A. F. (1995). Metaphors we Program By: Space, Action and Society in Java. In R. M. Baecker (Ed.), *Proceedings of 18th PPIG, Brighton*. Brighton.

[5] Blackwell, A. F. (2006). The reification of metaphor as a design tool. *ACM Transactions on Computer-Human Interaction*, *13*(4), 490–530.

[6] Boyd, N. S. (2004). *Software Metaphors*. Dostopno na: http://www.educery.com/papers/rhetoric/metaphors.

[7] Carroll, J. M., Mack, R. L. (1984). Learning to use a word processor: by doing, by thinking, and by knowing. In Thomas J., Schneider M. L. (Eds.), *Human factors in computer systems*. 13–51. Norwood: Ablex Publishing Corp.

[8] Cooper, A. (1995). The Myth of Metaphor. *Originally Published in Visual Basic Programmer's Journal*. Dostopno na: http://web.cs.wpi.edu/~dcb/courses/CS3041/Myth-of-Metaphor.pdf.

[9] Erickson, T. D. (1995). Working with interface metaphors. In R. M. Baecker, J. Grudin, W. A. S. Buxton & S. Greenberg (Eds.), *The art of Human-Computer Interface Design*. 147–151. San Francisco: Morgan Kaufmann Publishers.

[10] Honderich, T. (1995). *The Oxford Companion to Philosophy*. Oxford: Oxford University Press.

[11] Honeycutt, L. (2004). *Aristotle's Rhetoric*. Dostopno na: http://www.public.iastate.edu/~honeyl/Rhetoric/rhet3-10.html.

[12] Khaled, R. & Noble, J. (2005). Extreme programming system metaphor: A semiotic approach. In Billinghurst M (Eds.), *Proceedings of the 7th International Workshop on Organisational Semiotics*. 109–117. Newcastle.

[13] Koren, S. (2011). *Slovar informatike*. Dostopno na: http://www.islovar.org/iskanje_enostavno.asp.

[14] Lakoff, G. & Johnson, M. (2003). *Metaphors We Live By*. University Of Chicago Press.

[15] Madsen, K. H. (1994). A guide to metaphorical design. *Communications of the ACM*, *37*(12), 57–62.

[16] Neale, D. C. & Carroll, J. M. (1997). The Role of Metaphors in User Interface Design. In Helander Marting G. (Eds.), *Handbook of Human-Computer Interaction.* 441–462. New York: Elsevier.

[17] Pirhonen, A. (2005). To simulate or to stimulate? In search of the power of metaphor in design. In Pirhonen A et al (Eds.), *Future Interaction Design*. 105–124. New York: Springer.

[18] Reddy, M. (1993). The conduit metaphor: A case of frame conflict in our language about language. In A. Ortony (Ed.), *Metaphor and Thought.* 164–201. Cambridge: Cambridge University Press.

[19] Saffer, D. (2005). *Saffer, D."The Role of Metaphor in Interaction Design."* Thesis*.* Carnegie Mellon University.

[20] Züllighoven, H. (2005). *Object-oriented construction handbook: developing application-oriented software with the tools & materials approach*. Heidelberg: Morgan Kaufman.

▪

Saša Kuhar je asistentka za področje informatike na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, kjer je leta 2009 diplomirala s področja metafor. Trenutno je študentka bolonjskega doktorskega študija Medijske komunikacije, v okviru katerega se ukvarja s temi raziskovalnimi področji: vizualizacija podatkov, pregledne plošče, uporabniški vmesniki, interakcija človek–računalnik. Aktivno se udeležuje domačih in tujih konferenc s področja informatike.

▪

Marjan Heričko je redni profesor za informatiko na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, kjer je nosilec več predmetov, ki so v pristojnosti Inštituta za informatiko. Je namestnik predstojnika Inštituta ter vodja Laboratorija za informacijske sisteme. Doktoriral je leta 1998 na Univerzi v Mariboru na področju zagotavljanja kakovosti objektno orientiranega razvoja programske opreme. Njegovo raziskovalno delo zajema vsa področja razvoja sodobnih informacijskih rešitev in storitev s poudarkom na naprednih pristopih k modeliranju in načrtovanju informacijskih sistemov, načrtovalskih vzorcih in metrikah. V zadnjem obdobju je aktiven tudi na področju storitvene znanosti in storitvenega inženirstva kot tudi uporabniško usmerjenega razvoja.