Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Vesna Horvat

Razvoj programskega orodja za asistenco pri načrtovanju in vrednotenju grafičnih uporabniških vmesnikov

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Franc Jager

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Razvijte programsko orodje za pomoč načrtovalcem grafičnih uporabniških vmesnikov ob načrtovanju in vrednotenju uporabnosti grafičnih uporabniških vmesnikov. Programsko orodje naj bo razvito v okolju NetBeans in naj omogoča pomoč pri načrtovanju grafičnih uporabniških vmesnikov v smislu predstavljanja splošno sprejetih principov ter navodil načrtovanja grafičnih uporabniških vmesnikov in pomoč pri hevrističnem vrednotenje ter testiranju uporabnosti danega grafičnega uporabniškega vmesnika. Razvito programsko orodje testirajte na primeru hevrističnega vrednotenje in testiranja uporabnosti grafičnega uporabniškega vmesnika aplikacije Clicktrans.

Iskreno se zahvaljujem svojemu mentorju, podjetjema Infotrans d.o.o. in Kobal Transporti d.o.o. ter vsem dotičnim zaposlenim. Zahvaljujem se tudi Anji in Tini za strokovno pomoč, staršema, ki sta me zvesto podpirala ves ta čas, sestri, brez katere ne bi šlo, partnerju, ki je z mano delil breme študija, tastu in tašči ter vsem ostalim, ki so me med izdelavo diplome spodbujali in priganjali.

Mojemu G. To je to. Zdaj si pa ti na vrsti...

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
LCD	liquid-crystal display	tekočekristalni zaslon
OLED	organic light-emitting diode	organsko zasnovana svetleča dioda
UIUE	User Interface Usability Evaluator	Programsko orodje za asistenco pri načrtovanju in vrednotenju grafičnega uporabniškega vmesnika
VR	virtual reality	virtualna resničnost

Povzetek

Naslov: Razvoj programskega orodja za asistenco pri načrtovanju in vrednotenju grafičnih uporabniških vmesnikov

Avtorica: Vesna Horvat

V diplomskem delu so opisani vsi koraki načrtovanja in vrednotenja uporabnosti grafičnih uporabniških vmesnikov. Podrobneje so predstavljeni Normanovi namigi, Fittsov zakon, naloga pomikanja oz. vodenja, Mandelovi zlati principi, Nielsenovi principi, navodila načrtovanja in dimenzije uporabnosti. Hevristično vrednotenje in testiranje uporabnikov sta opisana teoretično kot tudi na praktičnih primerih. Vsa navodila in principi so, poleg hevurističnega vrednotenja in testiranja uporabnikov, vključeni v programsko orodje za asistenco pri načrtovanju in vrednotenju grafičnih uporabniških vmesnikov, ki smo ga razvili in testirali. Izdelali smo ga v razvojnem okolju Netbeans s kombinacijo Jave, Swinga in HTML. Programsko orodje se je v praksi obneslo dobro, z njegovo pomočjo pa smo sestavili primer hevurističnega poročila in poročila o testiranju uporabnikov za uporabniški vmesnik aplikacije Clicktrans.

Ključne besede: dimenzije uporabnosti, grafični uporabniški vmesnik, hevristično vrednotenje, navodila načrtovanja, Normanovi namigi, Fittsov zakon, naloga pomikanja oz. vodenja, Mandelovi zlati principi, Nielsenovi principi, testiranje uporabnikov, vrednotenje uporabnosti.

Abstract

Title: Development of a Programming Tool for Assistance During Development and Evaluation of Graphic User Interfaces

Author: Vesna Horvat

The thesis presents a description of the steps involved in designing and evaluating the usability of graphical user interferences. It presents in detail Norman's heuristics, Fitts's law, cursor guiding task, Mandel's golden principles, Nielsen's principles, design guidelines and usability dimensions. Heuristic evaluation and user testing are described theoretically and are furthermore presented by providing practical examples too. The guidelines and principles are included in a programming tool for assistance in designing and evaluating graphical user interferences that were developed and tested. The programming tool was created in the development environment NetBeans combined by Java, Swing and HTML. The programming tool performed well in practice. We used it to draw up an example of heuristic report and a user testing report of evaluating the user interface of the Clicktrans application.

Keywords: Usability Dimensions, Graphical User Interface, Heuristic Evaluation, Design Guidelines, Norman's Heuristics, Fitts's Law, Cursor Guiding Task, Mandel's Golden Principles, Nielsen's Principles, User Testing, Usability Evaluation.

Poglavje 1

Uvod

1.1 Namen dela

Namen diplomske naloge je predstavitev izbranih principov, namigov in navodil načrtovanja grafičnih uporabniških vmesnikov, razvoj programskega orodja, ki vsa ta navodila združuje in je tako ocenjevalcu v pomoč pri vrednotenju grafičnih uporabniških vmesnikov ter testiranje uporabniškega vmesnika aplikacije Clicktrans v smislu hevrističnega vrednotenja in preko testiranja uporabnikov.

Dober uporabniški vmesnik naj bi bil oblikovan z upoštevanjem navodil, principov in standardov, ki jih narekujejo strokovnjaki širom sveta. Ta so razpršena v različnih knjigah, priročnikih in najrazličnejših spletnih straneh. Naša ideja je bila združiti vsa ta navodila, principe in standarde na enem mestu. Programsko orodje, ki smo ga oblikovali v okolju Netbeans, združuje tako teoretično razlago principov in navodil načrtovanja, kot tudi praktične primere in protiprimere. Prav tako uporabniku omogoča, da svoja opažanja, ki jih odkrije v vlogi ocenjevalca grafičnega uporabniškega vmesnika, zabeleži. Sestavi lahko hevristično poročilo, poročilo testiranja uporabnikov ali pa preprost dokument z opažanji in drugimi komentarji, ki potencialno služi kot osnova za nadaljne izboljšave uporabniškega vmesnika, ki ga ocenjuje.

1.2 Načrtovanje grafičnih uporabniških vmesnikov

Načrtovanje uporabniških vmesnikov je poddomena področja raziskovanja o interakciji med ljudmi in računalniki (angl. human-computer interaction). Sodelovanje med načrtovalci in uporabniki mora biti vzajemno. Uporabniški vmesnik mora biti načrtovan tako, da bo čim bolj učinkovito zadovoljil potrebe uporabnika, načrtovalec pa mora pri tem upoštevati tehnične karakteristike in morebitne omejitve sistema [41].

Dober uporabniški vmesnik spodbudi enostavno in naravno interakcijo med uporabnikom in sistemom ter omogoča tekočo izvršitev nalog. Na ta način uporabnik ne razmišlja o tem, da se ukvarja z računalnikom, temveč se osredotoča le na svojo nalogo [10], kar hkrati razbremeni uporabnikov spomin in poveča njegovo produktivnost. Dober grafični uporabniški vmesnik je tudi estetsko privlačen, vsebuje smiselne in prepoznavne reprezentacije in ohranja konsistentnost. Omogoča direktno manipulacijo, takojšnjo povratno informacijo, za morebitne napake pa ne krivi uporabnika, temveč sistem [3].

Sam proces načrtovanja grafičnega uporabniškega vmesnika naj bo iterativen. Po pregledu obstoječega stanja, iskanju in prebiranju literature sledi zbiranje idej. Ko izločimo vse slabe in ostane le najboljša, izdelamo prototip. Teh je več vrst. Že na tej stopnji lahko prototip testiramo in po potrebi preoblikujemo. Ta postopek ponavljamo, dokler s prototipom nismo zadovoljni. Sledi izdelava vmesnika, pri čemer moramo izbrati primerne naprave za interakcijo, ustrezno načrtevati okna, menije in grafične gradnike za interakcijo, te ustrezno aranžirati in izbrati ter načrtovati raznorazne ikone. Izbrati je potrebno tudi primeren tekst, barve, slike in animacije ter razmisliti o podajanju povratnih informacij. Ko je vmesnik dokončan, produkt testiramo in kasneje odpravimo morebitne napake, vkomponiramo predloge in izdelek izpopolnimo. Tudi tokrat postopek večkrat ponovimo. Le na ta način, z upoštevanjem smernic in navodil načrtovanja, lahko ustvarimo dober grafični uporabniški vmesnik [10, 23].

1.3 Vrednotenje uporabnosti grafičnih uporabniških vmesnikov

Po ISO 9241 standardu je uporabnost definirana glede na obseg oz. v kolikšni meri lahko določeni uporabniki uporabljajo sistem, izdelek ali storitev za doseganje določenih ciljev z učinkovitostjo, zmogljivostjo in zadovoljstvom v določenem kontekstu uporabe [10, 17]. Nielsen [20] pa je uporabnost definiral kot večdimenzionalno lastnost uporabniškega vmesnika, sestavljeno iz naslednjih spremljajočih atributov: naučljivost, učinkovitost, zapomljivost, varnost in zadovoljstvo.

Z upoštevanjem definicij uporabnosti lahko uporabnost vrednotimo na dva načina, z oceno strokovnjakov (hevristično vrednotenje) ali testiranjem uporabnosti. Obe tehniki sta učinkoviti, glede na dane okoliščine pa lahko vsako izmed njiju izvedemo na različne načine.

Oceno uporabnosti lahko podajo prijatelji, znanci, uporabniki ali pa za to izurjeni strokovnjaki. Slednje se je že v mnogih primerih [10] izkazalo za učinkovitejše. Vrednotenje lahko poteka v zgodnji ali kasnejši fazi izdelave vmesnika, poročilo pa naj bi vključevalo natančno definirane primere napak in pomanjkljivosti ter morebitna priporočila. Shneiderman in Plaisant [4, 5] navajata več različnih metod strokovnega ocenjevanja:

- *Hevristično vrednotenje.* Skupina strokovnjakov posamezno pregleda uporabniški vmesnik na podlagi določenih principov ali hevristik (npr. Normanovih namigov, Mandelovih zlatih principov, Nielsenovih principov idr.) in poda strokovno oceno v obliki poročila.
- *Ocena upoštevanja navodil načrtovanja.* Ocenjevalci pregledajo uporabniški vmesnik tako, da preverijo ali so navodila načrtovanja, ki jih je določilo podjetje, upoštevana in če je vmesnik oblikovan v skladu z zunanjo in notranjo podobo podjetja.
- *Pregled konsistentnosti.* Strokovnjaki preverijo upoštevanje vseh vidikov konsistentnosti uporabniškega vmesnika – terminologijo, pisavo, barvne sheme, postavitev, obliko vhodnih in izhodnih podatkov ter

ostale elemente.

- *Kognitivne vaje.* Strokovnjaki simulirajo uporabo vmesnika z vidika tipičnega uporabnika. Na ta način se pogosto preverja pravilno delovanje sistema in omogoča lažje razreševanje napak.
- *Metafora človeškega razmišljanja.* Strokovnjaki simulirajo uporabo vmesnika z vidika razmišljanja tipičnega uporabnika. Pri tem upoštevajo uporabnikove navade, tok misli, zavedanje in asociacije, povezavo med govorom in mislimi ter predhodno znanje.
- *Formalna ocena uporabnosti.* Na uradno organiziranem sestanku strokovni ocenjevalci izpostavljajo posamezne težave in jih nato skupaj z oblikovalci vmesnika predebatirajo. Na sestanku je lahko prisoten tudi posrednik, ki pomaga pri razrešitvi morebitnih nestrinjanj.

Test uporabnosti je metoda za iskanje napak na vmesniku. Cilj testiranja je najti čim več težav med samim testiranjem, da jih lahko popravimo še pred končno izdajo produkta. S testiranjem uporabnosti lahko torej izdelek popravimo ali izpopolnimo [31]. Shneiderman in Plaisant [4, 5] sta mnenja, da lahko testiranje uporabnosti izvedemo z eno samo ali s kombiniranjem naslednjih metod:

- *Papirnati modeli in prototipi.* Ekipa razvijalcev na enem ali več uporabnikih testira papirnati model ali prototip aplikacije. Vse, kar bi storil računalnik, ponazori človek, nekdo drug pa si pozorno beleži uporabnikove pripombe in komentarje. Tako testiranje je poceni in učinkovito.
- *Skopo testiranje uporabnosti.* Testiranje uporabnosti se izvede na manjšem številu uporabnikov (3 do 6), testiranje pa se lahko izvede večkrat v različnih fazah razvoja.
- *Tekmovalno testiranje uporabnosti.* Pri testiranju uporabnosti uporabniki primerjajo produkt s prejšnjimi različicami ali podobnimi produkti. Na ta način dobimo boljše rezultate, potrebnih je manj sodelujočih, samo testiranje pa lahko traja dlje časa.

- *Univerzalno testiranje uporabnosti.* Najbolj razširjena metoda, pri kateri testiramo čim bolj raznolike uporabnike (glede na starost, spol, izobrazbo, predznanje, itd.) na različni strojni in programski opremi ter omrežjih. Z univerzalnim testiranjem uporabnosti lahko izdelamo produkt, ki bo primeren za širši nabor uporabnikov.
- *Testiranje na terenu in prenosni laboratoriji.* Testiranje se izvede direktno v uporabnikovem službenem okolju (npr. v njegovi pisarni) ali pa se laboratorij opremi tako, da se uporabnik v njem počuti kar se da sproščeno.
- *Oddaljeno testiranje uporabnosti.* Testiranje poteka oddaljeno, preko spleta. Uporabniki se lahko povežejo na strežnik, kjer rešujejo naloge takrat, ko jim to najbolj ustreza. Metoda omogoča testiranje večjega števila uporabnikov, kar je posledično časovno manj potratno, uporabniki pa so med reševanjem testa bolj sproščeni, saj jih nihče neposredno ne opazuje.
- *Testiranje težavnosti.* Testiranje težavnosti se najpogosteje uporablja pri testiranju novih iger, namenjeno pa je predvsem iskanju kritičnih napak in pomanjkljivosti na področju varnosti sistema. Pri testiranju uporabnike pozovejo, da naj poiščejo čim več napak v delovanju in jih skušajo zaobiti na različne načine.

Poglavje 2

Metode

V literaturi in na spletu je javno dostopnih ogromno različnih pravil oblikovanja grafičnih uporabniških vmesnikov, ki jih strokovno imenujemo hevristike. Med njih uvrščamo: Normanove namige, Fittsov zakon, nalogo pomikanja oz. vodenja, Mandelove zlate principe, Nielsenove principe, navodila oblikovanja, ki jih navajajo Stone in ostali [10, 30]. V diplomski nalogi se bomo podrobneje dotaknili le nekaterih izmed njih.

2.1 Normanovi namigi

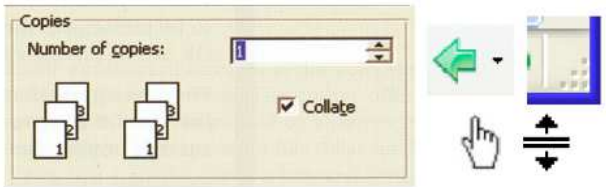
Norman [9, 7] je mnenja, da si mora vsak načrtovalec v procesu načrtovanja zastaviti sedem bistvenih vprašanj. Kaj želimo doseči? Kakšno bo alternativno zaporedje izvajanja akcij? Katere akcije lahko izdelamo takoj? Kako jih bomo izdelali? Kaj se je zgodilo? Kaj pomeni? Ali je to v redu in smo dosegli svoj cilj? Na podlagi teh vprašanj je določil sedem glavnih namigov uspešnega načrtovanja: vidljivost, povratne informacije, konceptualni model, pomagljivost, oznake, naravne preslikave in omejitve. Ker za načrtovanje grafičnih uporabniških vmesnikov konceptualni model in omejitve nista tako bistvena, bomo v nadaljevanju podrobneje predstavili ostalih pet namigov.

1. Vidljivost (angl. visibility)

Uporabniki ob pregledu uporabniškega vmesnika odkrivajo, katere funkcije se lahko izvajajo z ukazi, ki so na voljo [9]. Slika 2.1 prikazuje nekaj standardnih funkcij. Če so funkcije sistema dovolj preproste in očitne, da jih uporabnik lahko razume in uporablja, uporabniku razumevanje sistema ne bi smelo povzročati težav [23]. Pri nalogah, ki zahtevajo več korakov za izvršitev, so jim pri izbiri naslednjega koraka lahko v veliko pomoč dobro označeni ukazi na vidnem mestu [7].

Princip vidljivosti namiguje na to, da se uporabnost in učljivost povečata, če uporabnik zlahka vidi ukaze in možnosti, ki so mu na razpolago. Ukazi morajo biti jasno vidni in ne skriti. Najbolje je, če so postavljeni tam, kjer jih uporabniki pričakujejo. Postavljanje ukazov na nepričakovana mesta je enako, kot da bi jih skrili. Funkcionalnosti, ki so brez vidne predstavitve, je težko odkriti in najti. Na primer bližnjice na tipkovnici prihranijo čas strokovnim uporabnikom, če pa je bližnjica na tipkovnici edini način aktivacije ukaza, ga bodo novi uporabniki lahko odkrili samo po sreči ali pa bodo bližnjico za ukaz morali poiskati v uporabniškem priročniku [9, 24].

Principa vidljivosti ni nujno interpretirati tako, da mora imeti vsaka funkcija programa pripadajoči gumb na zaslonu. V primeru zahtevnejše aplikacije bi to pomenilo, da bi zaslon hitro postal prenasičen z gumbi in posledično bi bila vidljivost pravega gumba otežena. Primer kompromisa je izvlečni meni, pri katerem so ukazi vidni, ko je meni odprt, drugače pa so skriti. Pri aplikacijah z večjim številom funkcionalnosti je pametno razmisliti, da uporabniku ponudimo samo ukaze, ki jih potrebuje za trenutno opravilo [24].



Slika 2.1: Preproste in očitne funkcije sistema.

2. Povratne informacije (angl. feedback)

Če se v primeru pritiska na gumb ne zgodi nič, se uporabniki sprašujejo, ali je bil pritisk gumba dejansko zaznan, ali naj pritisnejo ponovno, ali je med pritiskom na gumb in izvedbo pričakovane akcije zamuda [9, 7].

Princip povratne informacije namiguje, da bi med uporabo sistema uporabnik moral dobiti takojšen in očiten znak, da je bila njegova izvedba ukaza zaznana oz. je dala nek rezultat. Ti znaki so lahko zvočni ali vidni. Slika 2.2 prikazuje različne primere vidnih povratnih informacij. Povratna informacija mora dati uporabniku vedeti, ali je bila akcija izvedena uspešno, ter ali je njegovo delo pravilno [23]. Ločimo dve vrsti povratne informacije [24]:

- Aktivna povratna informacija potrdi, da je bil ukaz aktiviran uspešno. Potrdi npr. da je bil gumb pritisnjen, da je bila možnost menija izbrana, ali da je bil drsnik prestavljen na novo mesto. Dokaz tega je lahko predstavljen z vidno povratno informacijo, npr. gumb na zaslonu se lahko osenči tako, da daje občutek, da je bil dejansko pritisnjen, igralna konzola pa zavibrira, ko uporabnik pritisne na določen gumb. Slušno povratno informacijo lahko zagotovimo z zvočnim efektom.

- Vedenjska povratna informacija potrdi, da je bil ukaz izvršen z nekim učinkom na sistem. Tak primer je pošiljanje spletne pošte. Potem, ko pritisnemo gumb pošlji, dobimo potrditveno obvestilo, da je bilo sporočilo poslano, pošta pa se prestavi v mapo s poslanimi sporočili.

Slika 2.2: Uporabnik mora dobiti takojšen in očiten znak, da je bila njegova izvedba ukaza zaznana (sprememba kurzorja, barve, oznake ipd.).

3. Pomagljivost (angl. affordance)

Pomagljivost je vidni atribut grafičnega gradnika ali ukaza, ki daje uporabniku namige, kako naj grafični gradnik ali ukaz uporablja. V realnem svetu npr. okrogla kljuka namiguje, da jo zavrtimo, ravna plošča nakazuje, da vrata odrinemo od sebe, ročaj pa vabi, da vrata povlečemo proti sebi [9, 24].

Pri grafičnih uporabniških vmesnikih lahko uporabimo koncept pomagljivosti tako, da damo grafičnim gradnikom s pomočjo vidnih znakov občutek dotakljivosti oz. možnosti pritiska. Pogosta tehnika je, da gumbe in ostale grafične gradnike prikažemo kot tridimenzionalne, s pomočjo uporabe senčenja in barv, s katerimi simuliramo svetlobo [24, 27]. Kadarkoli je le mogoče, uporabljamo grafične gradnike, ki jih ponuja operacijski sistem in s tem zagotovimo večjo prepoznavnost, saj jih uporabljajo tudi ostale aplikacije sistema [7, 24].

V sistemih z namizjem, ki uporabljajo kazalne naprave, kot je npr. miška, se pomagljivost pokaže kot sprememba miškine puščice v primeru lebdenja le te nad besedilom. Težje prikazljiva pomagljivost pa je npr. prikaz možnosti premika elementa iz enega mesta na drugo ali pa možnost prikaza kontekstnega menija z desnim klikom na element. V takih primerih lahko pomagljivostne znake zagotovimo z oblikovnim dogovorom. Nekaj primerov pomagljivosti prikazuje slika 2.3. Na primer podčrtano modro besedilo je standardni dogovor za besedilno povezavo na spletno stran. Pri podčrtanem modrem besedilu nič samo po sebi ne kaže na to, da se besedilo lahko klikne, vendar so se uporabniki zaradi splošne uporabe tega naučili [24].

Slika 2.3: Zagotovite namige, kako naj se grafični gradnik uporablja.

4. Oznake (angl. signifiers)

Kadar preproste vsakdanje stvari potrebujejo slike, oznake ali označevalce, so slabo načrtane [11], zato imajo oblikovalci pri načrtovanju pogosto praktične težave. Vedeti morajo, kako oblikovati stvari, da bodo razumljive. Ukazi se lahko izvajajo z miško, posebnim pisalom ali s prstom. Nekateri sistemi se odzovejo na gibanja telesa, kretnje in izgovorjene besede brez dotika kakršne koli fizične naprave. Če pomagljivosti smiselno ne prikažejo v celoti, morajo na njeno prisotnost opozoriti z nekim načinom signalizacije. Iz tega razloga so oznake veliko pomembnejše od same pomagljivosti [9, 7].

Oznake pri grafičnih uporabniških vmesnikih določajo, kako ljudje odkrivajo njihove funkcionalnosti. So znaki ali zaznavni signali o tem, kaj je mogoče storiti [9, 7]. Na ta način lahko izboljšajo pomagljivost oz. prepoznavnost [11], kakor velja za ikone na sliki 2.4.

Slika 2.4: Primeri nestandardnih ikon z oznakami.

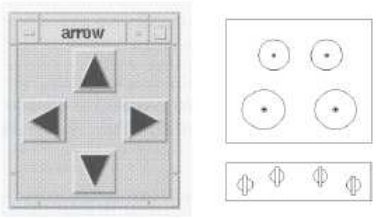
5. Naravne preslikave (angl. mapping)

Naravne preslikave namigujejo, da mora obstajati neko logično razmerje med tem, kako se sistem uporablja in kako je prikazan. Bolj kot je razmerje jasno, hitreje se uporabniki navadijo sistema. Problem, ki se lahko pri tem pojavi je, da bolj kot je sistem zapleten, težje ga je narediti lahkega za uporabo. Če v račun ne vzamemo prejšnjih povezav, bodo uporabniki zmedeni glede uporabe naključnih preslikav [9, 23].

Pritisk gumba ali aktiviranje ukaza ponavadi sproži izvajanje neke funkcije v sistemu. Do tega pride, ker obstaja povezava oz. naravna preslikava

med ukazom in njegovim učinkom. Preslikave naj bodo vedno čim bolj jasne in eksplicitne. To lahko dosežemo z uporabo opisnih oznak ali ikon na gumbih in elementih menija ter s konsistentno uporabo ukazov. Če imamo urejen seznam zaporedja ukazov, naj si ti sledijo od leve proti desni ali od zgoraj navzdol [7, 24].

Ukazi naj bodo logično postavljeni, da so čimbolj podobni objektom stvarnega sveta. Na primer, drsnik za nadzor ravnovesja med levim in desnim zvočnikom poveča glasnost levega zvočnika, če ga premaknemo levo [24]. Slika 2.5 prikazuje naravno preslikavo smernih puščic in kuhalne plošče.



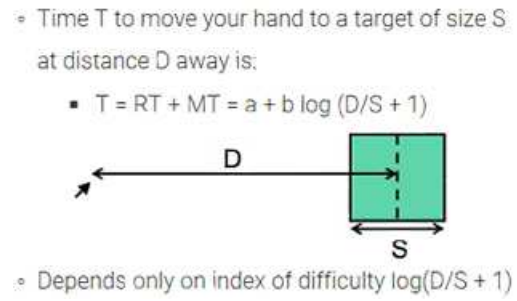
Slika 2.5: Preslikave naj bodo čimbolj podobne objektom stvarnega sveta.

2.2 Fittsov zakon

Fittsov zakon določa, kako hitro lahko premaknemo roko do cilja določene velikosti na določeni razdalji (v dolžini rok). Je temeljni zakon človeških senzornih sistemov in motoričnega sistema, ki so ga ponovile številne študije. Fittsov zakon velja tudi za prikaz točke na zaslonu z uporabo miške. V enačbi, ki je prikazana na sliki 2.6 je RT reakcijski čas oz. čas, ki ga porabimo, da roko premaknemo, MT pa motorični čas gibanja oz. čas, porabljen za premikanje roke.

2.2.1 Obrazložitev Fittsovega zakona

Fittsov zakon se nanaša na nadzor zaprte zanke. Predpostavimo, da je $D >> S$, zato je roka sprva daleč od cilja. V vsakem ciklu motorični sistem

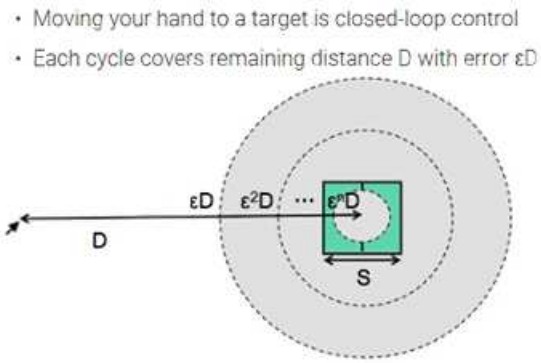


Slika 2.6: Fittsov zakon: Čas premika roke na razdalji D do cilja velikosti S je sestavljen iz reakcijskega časa in časa gibanja.

naroči roki, da naj se premakne za celotno preostalo razdaljo D. Motorični sistem lahko to izvede v enkratnem gibanju razmeroma konstantnega časa T_m, vendar je natančnost tega posameznega gibanja sorazmerna premaknjeni razdalji, zato se roka lahko znajde znotraj določene napake cilja ED (malo pred ciljem ali za njim). Napako cilja predstavlja sivo območje na sliki 2.7. Senzorni sistemi zaznajo, do kam se je roka premaknila v konstantnem času T_p, kognitivni sistem pa to lokacijo primerja s ciljem v času T_c. Nato motorični sistem izda popravek, da naj se premakne za preostalo razdaljo ED, kar tudi naredi, vendar spet s sorazmerno napako, tako da je roka v tem trenutku znotraj E^2D. Ta postopek se ponavlja, pri čemer se napaka geometrično zmanjšuje, vse dokler n iteracij ne prinese roke znotraj cilja.

2.2.2 Implikacije Fittsovega zakona

Rob zaslona samodejno zaustavi kazalec miške, tako da ne potrebujemo več kot enega popravljalnega cikla, da ga zadanemo. Rob zaslona sam po sebi pravzaprav deluje kot cilj neskončne velikosti. Podobno, če postavimo grafične gradnike na robove zaslona, morajo biti ti aktivni vse do roba, da popolnoma izkoristimo površino in povečamo učinek. Fittsov zakon tudi pojasnjuje, zakaj so krožni kontekstni meniji hitrejši za uporabo kot linearni pojavni meniji. Pri krožnem kontekstnem meniju (angl. pie menu), kakor ta



Slika 2.7: Napaka cilja je ponazorjena s sivim območjem v obliki krožnice, ki v več plasteh poteka okoli cilja.

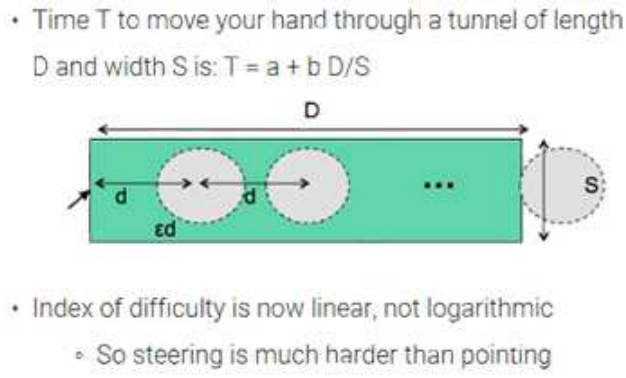
na sliki 2.8, je vsak element, kot košček pite, od kazalca miške oddaljen za enako razdaljo D, njegova velikost S (v radialni smeri) pa je primerljiva z D. Po drugi strani pa imajo elementi linearnega menija večji D in manjšo S (višino) [30].

Slika 2.8: Primer krožnega kontekstnega menija.

2.3 Naloga pomikanja oziroma vodenja

Naloga pomikanja oz. vodenja, ki jo prikazuje slika 2.9, od nas zahteva visoko pozornost nad napako, ki jo lahko naredi roka. Namesto, da bi iterativno zmanjševali napako, dokler ne pade pod velikost tarče, je potrebno paziti, da bo med premikanjem napaka manjša od velikosti predora. Motorični sistem

lahko vsak cikel roko premakne samo za kratko razdaljo d, tako da je napaka Ed nižja od S. Skupna razdalja D se zato doseže v $D/d = ED/S$ ciklih. Posledično je čas sorazmeren z D/S in ne z $logD/S$. Zaradi tega za klik na gradnik menija v kaskadnem podmenuju potrebujemo eksponentno več časa, kot bi ga, če ne bi bili omejeni s premikanjem po tunelu [30].



Slika 2.9: Naloga pomikanja oz. vodenja: čas premika roke skozi tunel dolžine D in širine S je sorazmeren z D/S.

2.4 Mandelovi zlati principi

Theo Mandel je predpostavil 24 principov oblikovanja, ki jih je razdelil v tri večje skupine. Poimenoval jih je "zlati principi oblikovanja" [33, 34].

1. Zagotovi nadzor uporabnika nad opravili in postopki

Skupina zajema premišljeno uporabo načinov, uporabnikovo izbiro med uporabo miške ali tipkovnice in spreminjanjem fokusa, nazoren prikaz sporočil in besedil, reverzibilnost akcij, dobro povratno informacijo, zagotovitev smiselnih poti, prilagodljivost različnim potrebam in stopnjam znanja uporabnikov, transparentnost vmesnika ter direktno manipulacijo z gradniki vmesnika.

2. Reduciraj obremenitev uporabnikovega spomina

V tej skupini so razbremenitev uporabnikovega kratkoročnega spomina, zanašanje na prepoznavo in ne na priklic, zagotavljanje vizualnih namigov, bližnjic, privzetih vrednosti ter funkcij razveljavi in obnovi, spodbujanje uporabe ustrezne sintakse, metafor iz realnega sveta, vizualne razločnosti in jasnega napredka.

3. Zagotovi konsistentnost uporabniškega vmesnika

V tretjo skupino je Mandel uvrstil jasno povezavo posameznih akcij s celotnim kontekstom vmesnika, konsistenten estetski izgled vmesnika, ki je v koraku s časom, konsistentno delovanje s pričakovanimi rezultati ter spodbujanje raziskovanja funkcij vmesnika, brez strahu pred negativnimi posledicami.

2.5 Nielsenovi principi

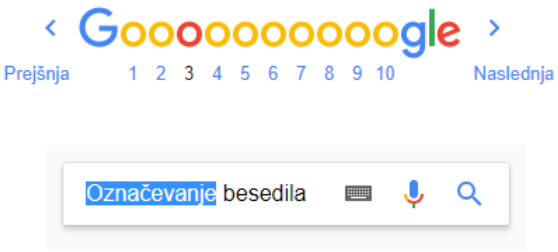
Nielsenovi principi so precej obširni in se nanašajo tako rekoč na vse vrste grafičnih uporabniških vmesnikov. Služijo kot smernice pri oblikovanju k uporabnikom usmerjenega uporabniškega vmesnika [20].

1. Vidljivost statusa sistema

Uporabnik mora biti ves čas obveščen o statusu sistema. Povratne informacije morajo biti podane v ustreznem odzivnem času. S tem ustvarjamo iluzijo napredka [19, 11].

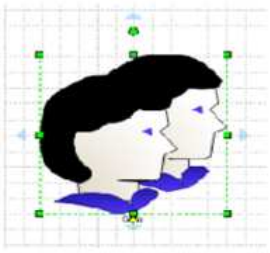
V ukazni vrstici lahko uporabnik nemudoma dobi odgovor na nek ukaz, medtem ko določene akcije ostanejo nevidne. Tekoče spremembe lahko poudarimo s statusno vrstico, v kateri se izpiše uspešnost oziroma neuspešnost neke akcije, ali pa s pojavnimi okni, spremembo barv ali teksture, določenimi kratkimi zvoki in ostalimi podobnimi vizualizacijami. Primera vizualizacije tekočih sprememb sta prikazana na sliki 2.10. Bistveno je, da smiselno pred-

vidimo kam bo uporabnikova pozornost usmerjena in da v povezavi z uporabnikovo nalogo prikažemo ustrezno vizualizacijo [11, 30].



Slika 2.10: Statusna vrstica brskalnika Google in primer označevanja besedila.

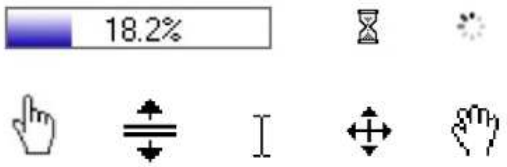
Izbor je še posebej pomemben. Ko uporabnik izbere grafični gradnik, na katerega želi delovati, ga označi. Za oblike in predmete lahko izbiro poudarimo z ročicami za izbor [30] kot prikazuje slika 2.11.



Slika 2.11: Primer ročic za izbor, ki nakazujejo možnost uporabe.

Med izvedbo akcije nad grafičnim gradnikom, uporabnik v zaznavnem intervalu (angl. perceptual fusion) skoraj neopazno prejme povratno informacijo o spremembi stanja gradnika, nad katerim je bilo dejanje izvedeno. Če sprememba stanja traja dlje od zaznavnega intervala, bo uporabnik opazil zamudo. V tem primeru se mora nekaj vidno spremeniti (na primer besedilo, kazalka, status sistema, barva ali oblika gradnika). Različne primere sprememb prikazuje slika 2.12. Miller [30] predlaga ukrepe le za določene odzivne čase:

- Sprememba v manj kot 0,1 sekunde je popolnoma zvezna.
- Spremembo, v času med 0,1 in 1 sekundo, uporabnik opazi, vendar povratna informacija ni potrebna.
- Pri spremembi, ki traja med 1 in 5 sekundami, naj kazalka spremeni obliko.
- Če sprememba traja več kot 5 sekund, poleg spremembe kazalke, prikažite tudi indikator napredka.



Slika 2.12: Primeri različnih indikatorjev napredka in oblik kazalcev, ki nakazujejo spremembo.

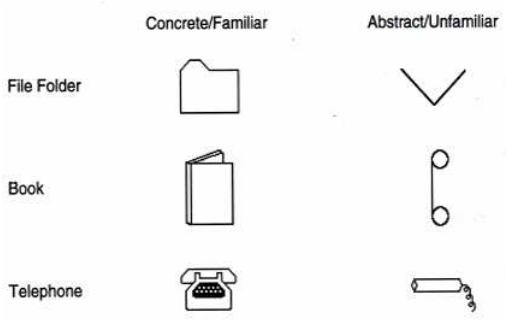
2. Prilagodi se realnemu svetu

Jezik sistema mora biti splošen. Uporabljajte besede, besedne zveze in pojme, ki so uporabnikom poznani in ne tehnični žargon. V primeru, ki ga prikazuje slika 2.13, bi neizkušen uporabnik lahko razumel, da mora odtipkati besedo "mismatch", medtem ko gre pravzaprav za opozorilo o neujemanju tipov. Ne omejujte imen, definiranih s strani uporabnikov ter dovolite okrajšave in sinonime v ukaznih jezikih. Specifične izraze v dani domeni lahko uporabite, če je to primerno. Sledite družbenim normam, pri čemer naj se informacije pojavijo v naravnem in logičnem vrstnem redu [19, 11, 20].

Z uporabo metafor lahko podobe realnega sveta prenesemo v svoj vmesnik. Dobro izbrana in izvedena metafora je zelo učinkovita, medtem ko je v nasprotnem primeru lahko zavajajoča. Primere dobrih in slabih metafor

Slika 2.13: Primer slabega opozorila o napaki.

prikazuje slika 2.14. Prednost metafore je, da si sposodimo konceptualni model, s katerim ima uporabnik že izkušnje. Metafore lahko naenkrat posredujejo veliko informacij o modelu vmesnika [30]. Prav tako razbremenjujejo uporabnikov spomin, saj so hitro naučljive, če so primerne [11].



Slika 2.14: Levo so prikazani primeri dobrih metafor in desno slabih.

Pomagljivost (angl. affordance) je namig o možni fizični akciji uporabnika nad nekim objektom. Nanaša se na zaznane in dejanske lastnosti objekta, predvsem na lastnosti, ki določajo, kako je ta objekt možno uporabiti [11]. Gumbi in hiperpovezave so najpreprostejša oblika pomagljivosti za izvedbo akcij. Npr. navzdol usmerjena puščica namiguje, da lahko vidite več možnosti, če kliknete puščico. Tekstura kaže, da je nekaj mogoče klikniti ali vleči [30]. Nekaj primerov prikazuje slika 2.15.

Slika 2.15: Primeri gumbov, hiperpovezav in tekstur, ki nakazujejo možnost uporabe.

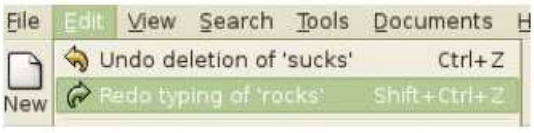
3. Uporabnikov nadzor in svoboda

Uporabnikov nadzor in svoboda sta pristopa, ki predstavljata idejo, da mora, pri dajanju in sprejemanju med uporabnikom in sistemom, uporabnik imeti končni nadzor. Najpreprostejša vrsta uporabnikovega nadzora je veto – možnost preklica operacije, čeprav jo je zahteval uporabnik sam [30]. Ti namreč pogosto izberejo sistemske funkcije po pomoti in potrebujejo jasno označeni "izhod v sili", ki jim bo omogočal, da zapustijo neželeno stanje takoj, brez nepotrebnega razširjenega dialoga [11, 20]. Dolgi postopki zato ne smejo imeti le vrstice napredka, ampak tudi gumb Prekliči (angl. Cancel) [19, 30]. Slika 2.16 prikazuje na levi pojavno okno, pri katerem ima uporabnik možnost nadzora in na desni pojavno okno brez možnosti nadzora.

Slika 2.16: Uporabnik mora imeti končni nadzor.

Če je gumb Prekliči najpogostejša možnost za uporabnikov nadzor nad pogovornim oknom, je gumb Razveljavi (angl. Undo) najpogostejša možnost za uporabnikov nadzor nad podatki. Ta razveljavi zadnjo akcijo, ki jo je uporabnik izvršil v tovrstni aplikaciji, kar pa ni nujno zadnji ukaz, ki ga je

uporabnik izdal sistemu kot celoti [30]. Za obnovitev zadnje razveljavljene akcije podpirajte funkcijo Obnovi (angl. Redo) [20]. Primer postavitve gumbov Razveljavi in Obnovi prikazuje slika 2.17.



Slika 2.17: Gumba Razveljavi in Obnovi.

4. Konsistentnost in standardi

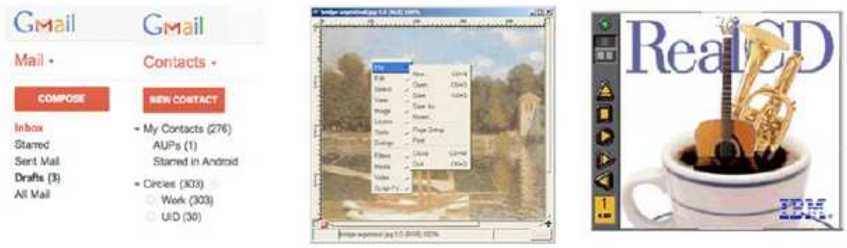
Uporabniki se ne bi smeli spraševati, ali različne besede, situacije in akcije pomenijo enako [19]. Upoštevajte princip najmanjšega presenečenja: podobne stvari naj izgledajo in se obnašajo podobno, različne stvari pa naj izgledajo in se obnašajo vidno drugače [11, 30]. To naj velja za vse lastnosti: velikost, položaj gradnikov, barvo, navodila, terminologijo, vrstni red ukazov in argumentov ter tudi ostale lastnosti [30].

Konsistentnost omogoča uporabniku, da svoje obstoječe znanje zlahka prenese na nov uporabniški vmesnik. Obstajajo tri vrste konsistentnosti: notranja (znotraj aplikacije), zunanja (skladnost z drugimi aplikacijami na isti platformi) in metaforična (metafora vmesnika v primerjavi z objekti realnega sveta) [11, 30]. Vse tri vrste konsistentnosti prikazuje slika 2.18.

Upoštevajte navodila platform [19, 11]: Java Look & Feel Design Guidelines, Apple Human Interface Guidelines, Windows User Experience Guidelines, GNOME Human Interface Guidelines, KDE User Interface Guidelines in druge.

5. Izogibanje napakam

Dobra sporočila o napakah so ključna, še pomembnejši pa je skrbno načrtovan vmesnik, ki preprečuje, da bi se težava sploh pojavila [19]. Onemogočite



Slika 2.18: Z leve proti desni: primer notranje, zunanje in metaforične konsistentnosti.

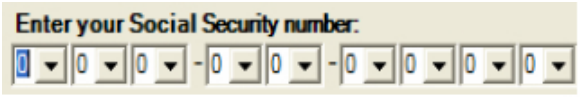
napačne (nemogoče pri dani akciji) ukaze ali vsaj preverite, kje se lahko pojavijo in uporabnikom vedno predložite možnost potrditve pred izvedbo tovrstne akcije [11].

Napake lahko klasificiramo kot spodrsljaje, pogreške in pomote, odvisno od tega, kako se pojavljajo. Spodrsljaj (angl. slip) je opustitev izvajanja ali nadzora - na primer zamenjava ene akcije z drugo v postopku. Pogrešek (angl. lapse) je napaka pomnilnika – npr. kadar pozabimo, kakšen je splošni cilj ali kje točno smo v postopku. Pomota (angl. mistake) pa je napaka, storjena pri načrtovanju ali uporabi določenega pravila [30].

Če želite zmanjšati število napak, ki nastanejo zaradi podobnih imen akcij, se izogibajte akcijam z enakimi predponami in dejanjem s podobnimi opisi.

Napakam v opisu se lahko izognete tako, da funkcije z destruktivnim učinkom (npr. izhod iz programa) dobro ločite od pogosto uporabljenih ukazov (npr. shrani, kopiraj, prilepi) [30]. Število napak se lahko zmanjša tudi z uporabo gradnikov za izbiro, npr. kombiniranih izvlečnih seznamov ali menijev in čim manj besedilnih območij, kjer je potrebno tipkati. Uporabnikovo delo lahko zaščitite s samodejnim shranjevanjem in funkcijo razveljavi [11]. Primer neustrezne postavitve gumbov je prikazan s sliko 2.19, slabo izbiro gradnikov pa prikazuje slika 2.20.

Slika 2.19: Pogosto uporabljeni gumb Odgovori se nahaja poleg gumba z destruktivnim učinkom Briši.



Slika 2.20: Včasih je polje za vnos besedila ustreznejša izbira.

6. Raje prepoznaj kot si zapomni

Zmanjšajte obremenitev uporabnikovega spomina, tako da so grafični gradniki, akcije in funkcije uporabniku jasno vidne. Uporabnik naj si ne bi potreboval zapomniti informacij iz enega dela dialoga do drugega, kot je prikazano z dialogom na sliki 2.21. Navodila za uporabo sistema morajo biti vidna ali zlahka dostopna, kadar je to primerno [19, 11]. Dosti lažje je namreč prepoznati nek vsesplošno znan vizualni namig, kot pa se samostojno spomniti nečesa iz glave [30].

Uporaba menijev in polj za vnos je veliko bolj naučljiva kot uporaba ukaznega jezika. Za grafične uporabniške vmesnike lahko uporabite direktno manipulacijo, ki vključuje naslednje lastnosti [30]:

- Neprekinjena vizualna predstavitev podatkovnih gradnikov sistema (na primer ikone, ki predstavljajo datoteke in mape, grafični objekti v urejevalniku slik ali besedilo v urejevalniku besedila), kot prikazuje slika 2.22.

- Uporabnik z vizualno predstavitvijo komunicira preko fizičnih akcij ali pritiskov na označene gumbe in simbole (na primer s klikom na gradnik ga izbere, če gradnik povleče, ga premakne, če pa povleče oprimek za

spremembo velikosti, pa ga spremeni).

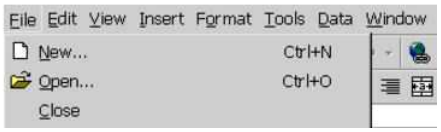
- Učinki posameznih akcij morajo biti hitro opazni (vidni čim hitreje, brez uporabnikovega dodatnega posredovanja), inkrementalni (vsaka, tudi najmanjša, sprememba na gradniku naj bo vidna) ter reverzibilni (omogočena naj bo razveljavitev katere koli operacije).

Slika 2.21: Primer neustreznega dialoga, ki obremenjuje uporabnikov spomin.

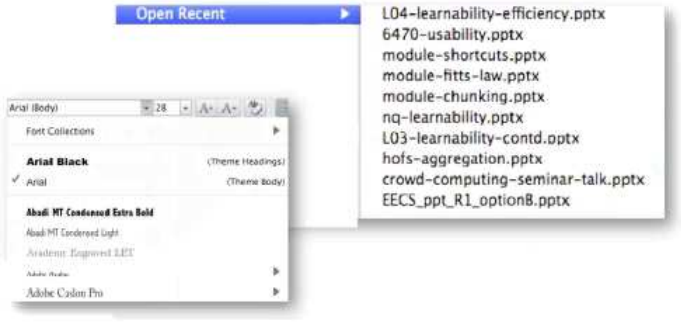
Slika 2.22: Vizualna predstavitev podatkovnih gradnikov sistema.

7. Fleksibilnost in učinkovitost uporabe

Pospeševalci, ki jih nov uporabnik niti ne vidi, lahko zelo pospešijo interakcijo za izkušenega uporabnika, tako da lahko sistem hkrati poskrbi tako za neizkušene kot tudi za izkušene uporabnike. Zagotovite uporabo pospeševalcev, mnemonikov in lahko naučljivih bližnjic preko tipkovnice za pogoste operacije [19], kot prikazuje slika 2.23. Prav tako omogočite uporabo privzetih nastavitev, zgodovine in okrajšav dolgih ukazov (angl. alias) [11]. Primer zgodovine zadnjih odprtih dokumentov in izbranih pisav prikazuje slika 2.24.



Slika 2.23: Uporaba pospeševalcev, mnemonikov in bližnjic preko tipkovnice.



Slika 2.24: Zgodovina zadnjih odprtih dokumentov in izbranih pisav.

Privzete vrednosti so pred izpolnjeni odgovori v poljih z besedilom, ki jih program predlaga na podlagi predvidevanja ali najpogostejših preteklih odgovorov. Privzete vrednosti morajo biti zasnovane tako, da se ob kliku na polje zapis v celoti označi, tako da ga lahko uporabnik nemudoma nadomesti z novo vrednostjo. Ta tehnika se imenuje brisanje v teku (angl. pending delete) [30].

Poljem za vnos besedila lahko povečate učinkovitost tudi z uporabo samodejnega dopolnjevanja (angl. autocomplete), ki minimizira tipkanje s pomočjo predvidevanja uporabnikovih potreb [30]. Slika 2.25 prikazuje na kakšen način je predvidevanje besedila zasnovano v brskalniku Google.

Slika 2.25: Predvidevanje besedila v brskalniku Google.

8. Estetika in minimalistično načrtovanje

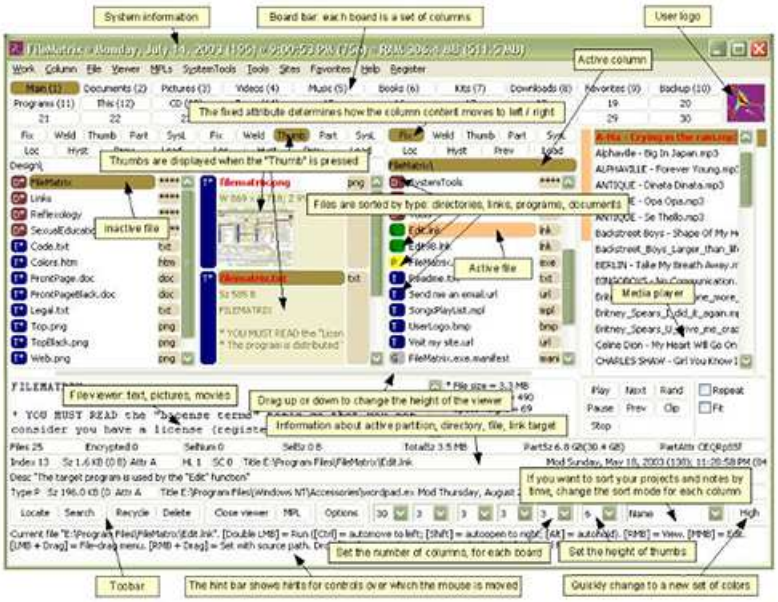
Upoštevajte pravilo preprostosti: "Manj je več" [11]. Preprosto načrtovanje s čim manj deli in elementi teži k boljšemu oblikovanju v vseh treh dimenzijah uporabnosti: učljivosti, učinkovitosti in varnosti. Preprostost lahko dosežete tako, da odstranite vse nepotrebne elemente - izogibajte se odvečnim informacijam, grafikam in lastnostim. Dober primer preprostega dizajna so ikone. Na sliki 2.26 je primer preprostega brskalnika.

Slika 2.26: Osnovna stran brskalnika Google.

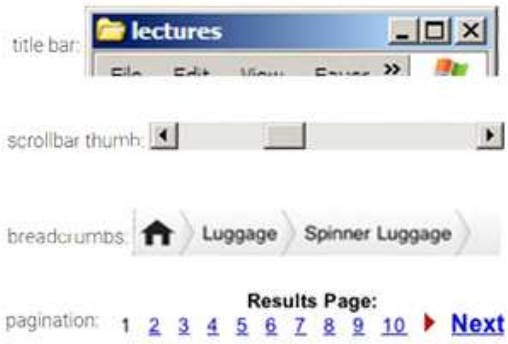
Kontrast se nanaša na zaznavne razlike vzdolž vizualne dimenzije, kot sta velikost ali barva. Če ste se odločili, da je v vašem vmesniku bistven kontrast, izberite ustrezno vizualno spremenljivko, ki ga bo predstavljala. Pri tem upoštevajte informacije, ki jih skušate sporočiti in nalogo, ki jo uporabniki potrebujejo s podatki.[30]. Preveč elementov in barv na kupu lahko naredi vmesnik nepregleden. Primer z barvami in gradniki zasičenega vmesnika prikazuje slika 2.27.

Ena izmed tehnik za dosego preprostosti je tudi večkratna vloga posameznih gradnikov. Slika 2.28 prikazuje primer naslovne vrstice, drsnika, nivojskega razčlenjevanja na podskupine in oštevilčenje spletnih strani. Slednje uporabniku prikazuje na kateri strani se trenutno nahaja, hkrati pa mu omogoča, da se z enim samim klikom premakne na neko drugo stran [30].

Svoj vmesnik dobro grafično načrtujte. Za poravnavo in izdelavo gradnikov uporabite čim bolj konsistentno oblikovanje, manjše število dobro izbranih pisav, barv in podrobnosti. Oznake izbirajte pazljivo, uporabljajte kratke izraze ter jedrnat jezik. Za ločevanje posameznih elementov namesto črt uporabljajte bele presledke, pri grupiranju vizualno podobnih gradnikov

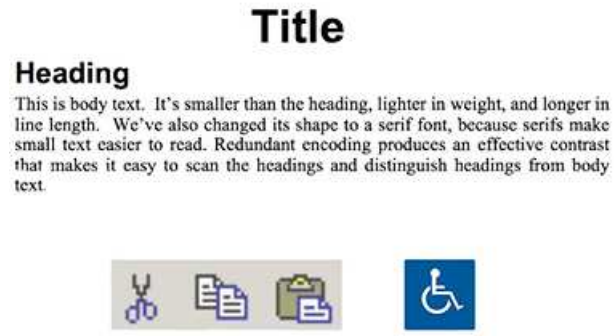


Slika 2.27: Popolnoma nepregleden vmesnik.



Slika 2.28: Večkratna vloga gradnikov.

pa si pomagajte s principi šole Gestalt [30, 11]. Ti opisujejo, kako ljudje navadno vidijo predmete z združevanjem podobnih elementov, prepoznavanjem vzorcev in poenostavitvijo kompleksnih slik. Upoštevajte predvsem zaključek (preoblikovanje), konveksnost, bližino, simetrijo in podobnost gradnikov[16]. Na sliki 2.29 so predstavljeni primeri enotnega oblikovanja teksta in ikon.



Slika 2.29: Enotno oblikovanje teksta in ikon.

Dialogi ne smejo vsebovati informacij, ki so nepomembne ali redkeje potrebne. Vsaka dodatna enota informacij v dialogu zmanjšuje relativno vidljivost ustreznih informacijskih enot [19].

9. Javljanje napak, diagnoza in reševanje

Sporočilo o napaki je dokaz omejitve ali pomanjkanja prilagodljivosti s strani sistema. Najboljša rešitev je preprečitev, da bi do napake sploh prišlo ali pa po drugi strani poskus zaobitve napake brez uporabnikove vednosti. Sistem naj bo prilagodljiv in strpen, nesmiselne vnose lahko pogosto prezrete brez nepotrebnih obvestil [30].

Sporočila o napakah izražajte v preprostem jeziku (brez kode in tehničnih podrobnosti), natančno navedite, zakaj je prišlo do napake in konstruktivno predlagajte rešitev [19], kot kaže slika 2.30. Sporočilo naj bo vljudno in oblikovano tako, da ne graja uporabnika, temveč prevzame "krivdo" sistem. Pri tem ne uporabljajte izrazov, kot so: prepovedan (angl. forbidden), usoden

Slika 2.30: Pojavno okno z opisom napake, razlago in ponujenimi rešitvami.

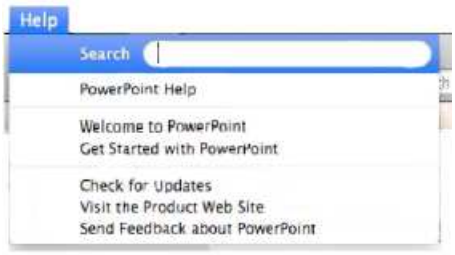
(angl. fatal), smrtonosen (angl. deadly), nezakonit (angl. illegal), prekinjen (angl. aborted) ali zaključen (angl. terminated), saj ti uporabnika lahko prestrašijo. Primer neprimernega pojavnega okna prikazuje slika 2.31. Raje uporabljajte nevtralen jezik [11, 30].

Slika 2.31: Pojavno okno z napako, ki lahko zaradi ikone z bombo prestraši uporabnika.

10. Pomoč in dokumentacija

Uporabniki običajno ne preberejo priročnikov, zato grafični vmesnik oblikujte tako, da bo prilagojen njihovim potrebam, hitro naučljiv in čim bolj interaktiven. Primer interaktivnega vmesnika prikazuje slika 2.32. Lastnosti naj bodo zasnovane tako, da jih uporabnik opazi hitro, zgolj s pogledom na vmesnik: zazna funkcije, prepozna ikone in kratice ali poudarjene besede [30].

Čeprav je bolje, če se sistem lahko uporablja brez navodil, je nujno potrebno zagotoviti pomoč in dokumentacijo. Dokumentacija ne sme biti preobsežna, temveč enostavna za preiskovanje, v kontekstu in usmerjena k upo-



Slika 2.32: Interaktivna pomoč uporabnikom.

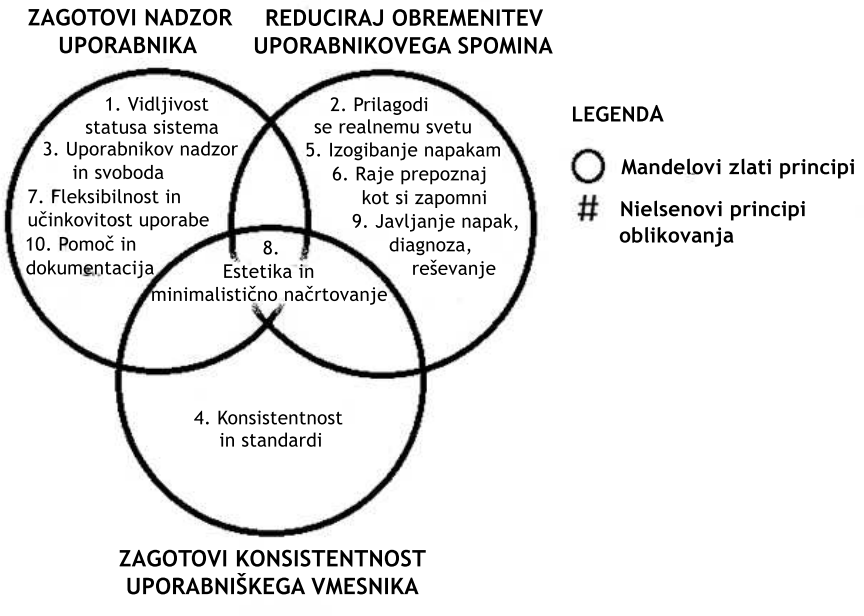
rabnikovi nalogi. Prav tako bi za rešitev problema morala navesti konkretne korake, ki jih je potrebno izvesti [19, 11]. Slika 2.33 prikazuje dobro organiziran sistem.

Slika 2.33: Dobro organiziran vmesnik, ki je enostaven za preiskovanje.

Klasifikacija Nielsenovih in Mandelovih principov

Ne glede na to, kaj bomo oblikovali, naj bo to domena, naprava ali uporabniški vmesnik, priporočljivo je, da pri tem upoštevamo čim več različnih navodil, principov in standardov oblikovanja [32, 33].

Glede na podobnosti lahko v Mandelove tri skupine principov klasificiramo tudi Nielsenove principe, kot kaže slika 2.34.



Slika 2.34: Klasifikacija Nielsenovih in Mandelovih principov.

2.6 Navodila načrtovanja

2.6.1 Pregled obstoječega stanja, iskanje in prebiranje literature

Pred vsakim večjim projektom moramo najprej preveriti obstoječe stanje doma in po svetu. Če se bomo lotili izdelave vmesnika svetovnega merila, moramo temeljito prečesati svetovni splet ter preveriti, kakšni izdelki že obstajajo in kaj nudijo. Ko zberemo dovolj povratnih informacij se lahko lotimo iskanja in pregledovanja gradiva. Med iskanjem si delamo izpiske. Nekatere teme so zelo dobro pokrite, o drugih pa je napisanega malo. Prav tako ni vsa literatura nam dostopna. Uporabimo lahko najrazličnejše gradivo (knjige, članke, spletne dokumente, laboratorijske ali terenske meritve in celo ustne vire). Če se odločimo, da imamo dovolj ustreznega materiala, oblikujemo seznam [12].

2.6.2 Zbiranje idej

Tudi ideje lahko razvrstimo v seznam. Sestavljanje seznamov je namreč eden izmed najbolj preprostih, a kljub temu izjemnih načinov za organiziranje naših idej. Na vrh seznama postavimo najbolj obetavne ideje, ki so lahko že dober približek temu, kar bomo izdelali. Ideje lahko organiziramo tudi grafično, z miselnim vzorcem. V izhodišče zapišemo besedo ali kratko frazo, iz katere rišemo povezave do drugih idej ali asociacij, ki se nam porodijo v miselnem toku. Povezave rišemo toliko časa, dokler nam ne zmanjka idej. Na ta način so ideje nesistematično razporejene, zato jih moramo kasneje skrbno pregledati in izluščiti najbolj obetavne [12].

2.6.3 Izdelava prototipov

Po izboru glavne ideje se lotimo izdelave prototipov. Ti omogočajo hitro in poceni povratno informacijo, preizkušanje različnih alternativ, poleg tega pa jih je lažje spremeniti ali zavreči. V začetni fazi je smiselna izdelava prototipov nizke natančnosti, ki ne vsebujejo detajlov, so izdelani iz cenejših materialov in vsebujejo drugačne tehnike interakcije. Šele ko smo zadovoljni s končnim izdelkom, lahko izdelamo prototip visoke natančnosti, ki pa je že zelo podoben končnemu izdelku [11, 30].

Glede na kompleksnost poznamo naslednje vrste prototipov [23]:

- verbalni prototip (tekstovni opis akcij in rezultatov),
- enostavni papirnati prototip (ročno narisane skice),
- dodelani papirnati prototip (skice, izdelane z urejevalniki slik),
- interaktivne skice (interaktivna sestava ročno narisanih skic) in
- delovni prototip (interaktiven žični model, izdelan s strokovnim orodjem za izdelavo prototipov).

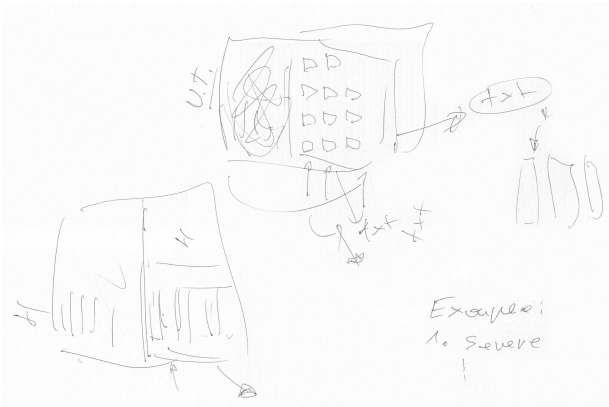
Verbalni prototip

Verbalni prototip precej spominja na skupek idej. Zapisan je kot seznam glavnih funkcij, ki bi jih vmesnik lahko podpiral. Sem lahko dodamo tudi

izbire, podizbire in rezultate posameznih akcij [23].

Enostavni papirnati prototip

Poleg verbalnega med prototipe nizke natančnosti spada tudi enostavni papirnati prototip. Kot kaže slika 2.35, so črno-bele skice narisane na roke, z njimi pa hitro dobimo dobro povratno informacijo. Na papirju lahko simuliramo ekran, dialoge in praktično vse gradnike vmesnika. Pri tem se nam ni potrebno obremenjevati z detajli, npr. tekst lahko simuliramo z vijugastimi črtami. Ob koncu ustvarjanja slabe primere zavržemo in dobre obdržimo [11, 23].



Slika 2.35: Primer enostavnega papirnatega prototipa.

Dodelani papirnati prototip

Prvi izmed prototipov visoke natančnosti je dodelani papirnati prototip. Izdelamo ga lahko z različnimi urejevalniki slik, kot sta npr. Adobe Illustrator ali Corel Draw. Po navadi so izdelani v barvah in vsebujejo večje število detajlov. Pogosta napaka ustvarjalcev je, da se pri izdelavi dodelanega prototipa preveč osredotočijo na zunanji izgled vmesnika, namesto da bi dodelali tok funkcionalnosti [23].

Interaktivne skice

Interaktivne skice so nadgradnja enostavnih ali dodelanih papirnatih prototipov. Izdelamo jih tako, da skeniramo skice in jih v nekem programu, npr. Macromedia Director ali Microsoft Expression SketchFlow, združimo tako, da lahko nanje klikamo. Na ta način simuliramo delovanje pravega vmesnika, kar nam ponudi še boljšo povratno informacijo o tem, ali je prototip razumljiv, ali manjka kakšna značilnost, se uporabniki znajdejo, ali razumejo oznake in kaj mora biti prikazano na določenem mestu. Na tem mestu lahko specificiramo tudi preostale detajle, kot so npr. barve, fonti in ikone. Teh elementov ne moremo integrirati v prototip, lahko pa jih izrišemo v ločeni datoteki, ki jo priložimo oz. pripnemo interaktivnim skicam [11, 23].

Delovni prototip

Delovni prototip je najvišja oblika prototipov visoke natančnosti. Vsebuje vse elemente prejšnjih prototipov skupaj z barvami, fonti in ikonami. Na njem lahko testiramo izgled, preverimo učinkovitost delovanja, odzivne čase in dinamiko z interaktivnimi povratnimi informacijami. Izdelamo ga lahko z različnimi generatorji vmesnikov, kot sta npr. Swing ali MFC. Za delovne prototipe je značilno, da so oblikovani horizontalno, kar pomeni, da omogočajo klikanje na posamezne elemente, ti pa ni nujno, da delujejo. Lahko omogočimo odpiranje določenih sekundarnih oken in menijev, kompleksnejših operacij pa v tej fazi še ne izvajamo. V tem trenutku lahko prototip preizkusimo in spremenimo ali dodamo še zadnje potencialne funkcionalnosti. Naš končni delovni prototip pa nato lahko služi kot odlična podlaga za izdelavo vmesnika [30, 11].

2.6.4 Izdelava programskega orodja

Preden se lotimo izdelave programskega orodja ali uporabniškega vmesnika, se moramo seznaniti s čim večjim številom principov, namigov in standardov oblikovanja. Univerzalnih principov oblikovanja ni, najbolj uspešni pa bomo,

če jih pri izdelavi programskega orodja upoštevamo čim več.

Schneiderman in Plaisant [5] navajata svojih 8 zlatih pravil oblikovanja:

- Stremite h konsistentnosti (doslednost čez celoten vmesnik) - npr. uporabljen je en font pisave za vmesnik in drug font za kodo.
- Oblikujte za vse uporabnike (vmesnik naj bo primeren tako za začetnike kot tudi za napredne uporabnike) - npr. do neke funkcije lahko dostopamo po dolgi poti skozi voden meni ali pa z uporabo pospeševalcev in mnemonikov.
- Nudite uporabno povratno informacijo (pri navodilih ali napakah) - npr. napake v kodi se obarvajo rdeče, prav tako se izpiše natančen opis težave.
- Oblikujte pojavna okna z dialogi, ki vodijo uporabnika od začetka do konca postopka - npr. nov projekt lahko ustvarimo le v nekaj korakih, pri čemer program jasno in razumljivo vodi uporabnika.
- Preprečite napake (sistem oblikujte tako, da do napak ne more priti) - npr. če neka bližnjica tipk ne obstaja, program ukaz preprosto ignorira.
- Dovolite enostavno razveljavitev akcij - npr. NetBeans podpira gumbe Razveljavi, Prekliči in Obnovi ter omogoča pregled zgodovine dela.
- Podpirajte notranjo kontrolo (uporabnik naj ima občutek, da sam nadzoruje vsa dejanja) - npr. uporabnik vedno sam nadzoruje postopke in poda ustrezna dovoljenja.
- Zmanjšajte obremenitev uporabnikovega spomina - npr. pri pisanju kode, glede na besedilo, ki ga uporabnik tipka, v spustnem seznamu ponudite možnosti ukazov, ki obstajajo.

2.6.5 Navodila načrtovanja

Vsa pravila, principi, namigi in standardi stremijo k istemu cilju. Oblikovati uporaben vmesnik, ki bo enostaven za uporabo in primeren za vse uporab-

nike, ki bo vseboval čim manj napak in bo nudil visoko stopnjo zadovoljstva za uporabnike. To pa lahko dosežemo le z upoštevanjem vsesplošnih navodil načrtovanja.

Izbor naprav za interakcijo

Vhodne naprave

Uporabnik komunicira z računalnikom preko vhodnih naprav. Mednje uvrščamo tipkovnico, nadzorno ploščo z gumbi, igralno konzolo, miško, kroglo za sledenje, krmilno palčko, zaslone na dotik, svetlobna pisala in sisteme za prepoznavanje glasu [10, 11].

Stone in ostali [10] so vhodne naprave razdelili v dve skupini: diskretne in kazalne naprave, slednje pa se delijo še na indirektne ali posredne in direktne ali neposredne naprave.

- *Diskretne naprave* so namenjene vnašanju posameznih elementov informacij, kot so npr. črke, številke in ukazi. Primeri takih naprav so tipkovnica, nadzorna plošča z gumbi in igralna konzola. Tovrstne naprave ponujajo veliko izbiro bližnjic in funkcij, s čimer povečajo učinkovitost, po drugi stani pa zaradi svoje velikosti in robustnosti zasedejo precej prostora [10].
- *Indirektne* ali *posredne naprave* prevedejo del človekovega gibanja v podatke. Primeri vključujejo računalniško miško, kroglo za sledenje ali krmilno palčko. Čeprav imajo te naprave različne fizične lastnosti, si delijo sposobnost kognitivnega prevajanja med človeškim telesom in strojem. Npr. premik miške naprej premakne kurzor navzgor po zaslonu. Zaradi tega posredne naprave omogočajo visoko natančnost za naloge, ki jih izvajamo na zaslonu [1].
- *Direktne* ali *neposredne naprave* nimajo posrednika. Gibanje telesa je enako vnosu naprave. Primeri neposrednih naprav so zasloni na dotik, svetlobna pisala in sistemi za prepoznavanje glasu. Neposredne naprave ne potrebujejo zavestnega kognitivnega prevajanja. Gibanje se preko dotika izenači s prikazano razdaljo na zaslonu, zato lahko funkcije

naprave z uporabo Fittsovega zakona predvidijo želen rezultat. Med drugim omogočajo tudi balistično gibanje [1].

Uspešnost izvedbe lahko določimo z ujemanjem med tipom vhodne naprave in zahtevami za vnos. Zahteve se namreč lahko zelo dobro ali pa slabo ujemajo z atributi vhodne naprave [1].

Izhodne naprave

Računalnik komunicira z uporabnikom preko izhodnih naprav. Najpogostejši je zaslon, vendar obstajajo še drugi, kot so 3D očala za virtualno resničnost (VR), Braillovi zasloni, zvočniki, slušalke, tiskalniki, lučke in diode [10, 41].

Računalniški zvoki so napredovali od preprostega zvoka do reprodukcije govora, glasbe in zvočnih učinkov. Za dosego pravega učinka mora kakovost zvočnika odražati kakovost predstavljenega zvoka [41].

Zasloni so zelo uporabni za predstavitev širokega nabora vizualnih elementov in zapletenih podatkov. Najbolj razširjena je uporaba tekočekristalnih (LCD) zaslonov, klasičnih in na dotik, saj so cenovno dostopni, majhni, tanki, lahki in imajo visoko ločljivost [41], na svetovnem trgu pa se pojavljajo tudi transparentni zasloni s svetlobo iz organsko zasnovanih svetlečih diod (OLED) in virtualni zasloni.

Galitz, Stone in ostali [10, 41] predlagajo, da pri izbiri zaslona upoštevamo naslednje:

- *Slika.* Kako podrobna mora biti slika? Za pretežno grafične aplikacije, ki vključujejo ikone, slike in fotografije, je zaželena uporaba zaslona z visoko ločljivostjo. Za delo z besedilom in uporabo večje velikosti črk je zadovoljiv zaslon z nižjo ločljivostjo.
- *Barve.* Koliko barv je potrebnih? Število barv se lahko giblje od črnobele ali ene same barve do milijonov različnih odtenkov. To izbiro lahko prepustimo aplikaciji.
- *Velikost.* Tako kot pri televizorjih se velikost računalniških zaslonov meri po diagonali. Večji kot je zaslon, več prednosti ima. Lahko se

prikaže več informacij, večje besedilo in slike. Velikost zaslona je odvisna od potreb aplikacije in njenih uporabnikov. Slabovidna oseba bo morda želela večje besedilo in slike, medtem ko je za ročne naprave primeren veliko manjši zaslon.

- *Prenosljivost.* Ali mora biti zaslon prenosljiv? LCD zasloni so lahki in prenosljivi.
- *Zavzemanje prostora.* Na splošno LCD zasloni zavzamejo zelo malo prostora na delovni mizi. Glede na možnosti in prostorske omejitve pa lahko izberemo manjši ali večji zaslon.

Načrtovanje oken in menijev

Značilnosti oken

Okno je območje na zaslonu, običajno pravokotne oblike in določeno z mejo, ki vsebuje pogled na neko določeno področje računalnika ali del pogovornega okna. Okno je mogoče premakniti in prikazati neodvisno od zaslona. Ta pa lahko vsebuje eno, dve ali več oken znotraj svojih meja. Vsako okno ima naslednje značilnosti [41]:

- Ime ali naslov, ki omogoča identifikacijo.
- Spremenljivo velikost po višini in širini.
- Stanje, aktivno ali neaktivno. Vsebino lahko spreminjamo samo aktivnim oknom.
- Vidnost dela, ki ga je mogoče videti. Okno je lahko delno ali popolnoma skrito za drugo okno ali pa informacije znotraj okna presegajo območje, ki ga okno prikazuje.
- Lokacija glede na omejitev zaslona.
- Ureditev glede na druga okna. Lahko se razprostira čez cel zaslon, prekriva drugo okno ali pa je postavitev kaskadna.
- Možnosti upravljanja in metode za manipulacijo okna.
- Poudarek oz. del, ki je izbran.
- Funkcijo, nalogo ali aplikacijo, kateri je namenjeno.

Vrste oken

Vrsta okna je odvisna od funkcije, ki jo opravlja [31, 41].

- *Primarno* ali *osnovno okno* je okno, ki se pojavi na zaslonu takoj, ko se dejavnost začne. Je nujno potrebno za vsako funkcijo ali aplikacijo, obvezno pa mora vsebovati menijsko vrstico in nekaj osnovnih nadzornih gradnikov. Predstavlja ogrodje za glavne ukaze in podatke, na podlagi katerega izhajajo vsa odvisna okna [41].
- *Sekundarna okna* so dodatna okna. Lahko so odvisna ali pa prikazana popolnoma neodvisno od primarnega okna. Sekundarna okna se uporabljajo za izvajanje dodatnih, podrejenih ali razširjenih nalog. Lahko so modalna ali nemodalna [41]. Modalna okna ne dopuščajo interakcije z drugimi okni, dokler se trenutni dialog ne zaključi, medtem ko nemodalna okna uporabniku omogočajo vzporedno interakcijo z različnimi okni oz. dialogi [31].
- *Pogovorna okna* (angl. dialog boxes) se uporabljajo za razširitev in zaključek interakcije v omejenem kontekstu. Prikažejo se vedno iz drugega okna - primarnega, sekundarnega ali drugega pogovornega okna. Lahko se prikažejo kot posledica aktivacije ukaznega gumba, izbire menija ali pa jih samodejno sproži sistem, če obstaja pogoj, ki zahteva uporabnikovo pozornost, potrditev ali dodaten vnos [41].

Postavitve oken

Dostop do dodatnih možnosti se lahko doseže z vključitvijo ukaznega gumba, ki odpre drugo sekundarno okno. Poiščite okno, tako da bo vidno. Če se okno obnovi, ga poiščite, kjer se je nazadnje pojavilo. Če je okno novo in lokacija še ni vzpostavljena, jo postavite na točko pozornosti gledalca. Prednostni položaji so v bistvu spodaj in desno [31, 41].

- *Kaskadno okno* ohranja izvorno okno, pri čemer je odvisno okno prikazano na vrhu, poravnano nekoliko v desno in pod predhodnim sekundarnim oknom. Kaskadna okna se običajno uporablja, ko želimo predstaviti napredne možnosti na nižji ravni v zapletenem pogovornem

oknu. Če s tovrstno postavitvijo prikrijemo podatke, ki jih želimo pregledovati hkrati, lahko postavimo okna vzporedno ali navpično, kakor ploščice [41].

- *Razširljivo sekundarno okno* se razširi, da razkrije dodatne možnosti. Razširjanje oken se običajno uporablja za zagotavljanje naprednih možnosti na enaki ravni v zapletenem pogovornem oknu [41].
- Če se *pogovorno okno* nanaša na celoten sistem, ga postavimo na sredino zaslona, pri čemer naj bodo ključne informacije spodaj ležečega okna še vedno vidne [41].

Vsebovalniki in ostali grafični gradniki

- *Harmonika* (angl. accordion) je navpično zložena zbirka naslovov, ki jih je mogoče klikniti, da prikažejo ali skrijejo z njimi povezano vsebino [13].
- *Zavihki* (angl. tabs) omogočajo uporabniku izbiro in pregled vsebine posamezne komponente, tako da izbere ustrezni zavihek [39].
- *Večplastna plošča* (angl. layered pane) je eden izmed Swingovih vsebovalnikov, ki omogoča tretjo dimenzijo za postavitev gradnikov - globino [37].

Načrtovanje menijev

Meniji omogočajo enostavno navigacijo skozi grafični uporabniški vmesnik, tako da na preprost način vizualizirajo konceptualni model vmesnika [31].

Za ustrezno organizacijo menijev je priporočena uporaba imen, ki odražajo funkcije menija. Imena naj bodo kratka in slovnično pravilna, vsak element vrstičnega menija pa naj bo zapisan z veliko začetnico. Elemente smiselno grupirajte. Izogibajte se dolgim menijem in upoštevajte velikost zaslona pri odločanju o številu elementov menija [10].

Vrste menijev

- *Vrstični meni* je običajno pozicioniran tik pod naslovom glavnega okna. Zasedati mora le eno vrstico in predstavljati kritične ali pogoste ukaze v smiselnem vrstnem redu. Vsak element vrstičnega menija naj bo opisan z eno besedo, klik nanj pa naj ne bi neposredno izvedel ukaza [10, 31].
- *Spustni meni* se prikaže, ko izberemo element vrstičnega menija ali ikono v orodni vrstici. Spustni meni mora vsebovati več kot en element, pri čemer mora biti vsak element edinstven. Uporabite ločevalne črte oz. separatorje za združevanje povezanih in ločevanje uničevalnih ukazov. Mnemonike in pospeševalce uporabite le za tiste elemente spustnega menija, za katere menite, da jih bodo uporabniki želeli uporabljati brez klikanja na meni [10, 31].
- *Kaskadni meni* je podmeni, ki se pojavi, ko se z miško premaknemo nad element spustnega menija, ki ima poleg imena na desni strani tudi trikotni simbol puščice. To lahko povzroči, da se na zaslonu istočasno prikažejo številni meniji. Ti od uporabnika zahtevajo še posebej dober nadzor nad miško, zato ne uporabljajte več kot dveh ravni kaskadiranja [10, 31].
- *Dvižni* oz. *izvlečni meni* je samostoječi premični meni. Če želimo prihraniti prostor na zaslonu, ga lahko dvignemo in ponovno izvlečemo. Dvižne menije lahko uporabljamo za ukaze, h katerim se uporabniki pogosto vračajo [10, 31].
- *Pojavni meni* je nepritrjen oz. lebdeč meni, ki ga po navadi pozovemo z desnim klikom miške. Položaj pojavnega menija na zaslonu je odvisen od položaja miške v trenutku, ko je pozvan. Lahko ga uporabite za podmnožico ukazov, ki so značilni za dano okolje [10, 31].
- *Orodna vrstica* služi kot bližnjica do elementov menija ali kot način predstavitve grafičnih gradnikov, ki jih je težko opisati z besedami. Orodna vrstica naj bo konsistentno oblikovana skozi celoten vmesnik. Prikazuje naj samo gradnike ukazov, ki so trenutno na voljo. Orodna

vrstica naj bo premična. Na ta način uporabnikov ne bo ovirala pri delu, po želji pa jo lahko prestavijo drugam. Uporabnikom prav tako omogočite, da lahko orodno vrstico sami vklopijo ali izklopijo preko pogovornega okna ali možnosti v spustnem meniju. To je še posebej pomembno, če imate več orodnih vrstic. Razmislite tudi o tem, da uporabnikom omogočite prilagajanje orodne vrstice, tako da sami določijo, kaj bo orodna vrstica prikazovala. Če je možno, lahko dodate tudi opise ali namige za uporabo [10, 31].

Izbor grafičnih gradnikov za interakcijo

Grafični gradniki so elementi zaslona, ki ga sestavljajo. Po definiciji so to grafični objekti, ki predstavljajo lastnosti ali delovanje drugih objektov. Že sam izgled grafičnega gradnika mora predstavljati, da je namenjen interakciji - da lahko vanj nekaj vnesemo ali pa ga kliknemo. Vsi gradniki morajo biti načrtovani konsistentno, predvsem pa tako, da služijo svojemu namenu [41].

Operativni grafični gradniki

Operativni grafični gradniki omogočajo vnos, izbiro, spreminjanje ali urejanje določene vrednosti ali pa povzročijo izvedbo ukaza. Operativni grafični gradniki so [41]:

- *Gumbi* so najprimernejša izbira za pogosto in hitro izvršljive akcije. Uporabimo jih lahko za shranjevanje dokumenta, izklop sistema ali brisanje besedila. Lahko prikazujejo razne možnosti, kot so barve ali pisave. Lahko pa se uporabljajo tudi za prikaz drugih sekundnih oken, razširitev pogovornega okna ali za prikaz nastavitev sistema ali aplikacije.

- *Polje za vnos* vsebuje besedilo, ki ga vnese ali spremeni izključno uporabnik z uporabo tipkovnice. *Polje, ki je samo za branje*, pa vsebuje besedilo ali vrednosti, ki so prikazane samo za namene branja ali prikazovanja in se jih ne da spreminjati.

- *Polja z besedilom* so najbolj uporabna pri neomejeni količini podatkov za vnos, ki jih težko kategoriziramo in so precej spremenljivi. Polja z besedilom so ponavadi edina možnost, ko ustvarjanje polja za izbiro ni mogoče.

Grafični gradniki za izbiro

Grafični gradnik za izbiro prikazuje vse možne alternative, pogoje ali izbire, ki lahko obstajajo za neko lastnost ali vrednost. Izmed prikazanih alternativ pa nato izberemo ključni element. Nekateri gradniki za izbiro lahko na zaslonu prikazujejo vse možnosti hkrati, drugi pa za prikaz vseh alternativ potrebujejo nek ukaz. Grafični gradniki za izbiro so [41]:

- *Radijski gumbi* so uporabni za nastavitev atributov, lastnosti ali vrednosti, če je na voljo dovolj prostora na zaslonu. V pomoč so pri manjšem številu težje zapomljivih alternativnih možnosti, pri katerih se vsebina ne spreminja. Radijske gumbe najpogosteje uporabimo takrat, ko je za neko vprašanje možen le en pravilen odgovor.
- *Potrditvena polja* so uporabna za nastavitev atributov, lastnosti ali vrednosti, če je na voljo dovolj prostora na zaslonu. Tudi ti so v pomoč pri manjšem številu težje zapomljivih alternativnih možnosti, pri katerih se vsebina ne spreminja. Potrditvena polja najpogosteje uporabimo takrat, ko je pri nekem vprašanju možnih več pravilnih odgovorov.
- *Palete* uporabljamo za nastavljanje atributov, lastnosti ali vrednosti medsebojno izključujočih izbir, če je na voljo dovolj prostora na zaslonu. Razmislite o uporabi palete, če imajo izbire lastnosti, ki jih lahko najlažje opišemo z ilustracijo. Uporabne so za diskretne in maloštevilne podatke in izbire, ki se poredko spreminjajo.
- *Izbirne sezname* uporabljamo za izbiro vrednosti ali pa nastavljanje atributov, ki se med seboj izključujejo ali ne, če je na voljo dovolj prostora na zaslonu, da prikaže šest do osem izbir. Najpogosteje jih uporabljamo za besedilne podatke, z večjim, fiksnim ali spremenljivim številom elementov, ki niso dobro znani in so pogosto nepredvidljivo

urejeni. Izbirne sezname lahko uporabimo takrat, ko je uporaba radijskih gumbov in potrditvenih polj zaradi količine podatkov in pomanjkanja prostora na zaslonu nepraktična.

- *Izvlečni* in *dvižni seznami* se uporabljajo podobno kot izbirni seznami, z izjemo tega, da pri izvlečnih in dvižnih seznamih izbire niso vedno vidne. Dodatno omogočajo še, da lahko seznam po potrebi izvlečemo in nazaj dvignemo, s čimer prihranimo prostor. Ne uporabljajte izvlečnega ali dvižnega seznama, če je pomembno, da so vse možnosti prikazane hkrati.

Kombinirana polja za vnos ali izbiro

Gradnik ima lahko značilnosti tako polja z besedilom kot tudi gradnika za izbiro. Pri tovrstnem grafičnem gradniku lahko informacije bodisi vnesete ali pa jih izberete in vstavite v polje. Kombinirana polja za vnos ali izbiro so [41]:

- *Krožna polja* uporabljamo za nastavitev atributov, lastnosti ali vrednosti, ki se med seboj izključujejo. Uporabni so, kadar naloga ali uporabnik zahteva možnost vnosa ali izbire ključa s seznama. Krožna polja so uporabna za diskretne in maloštevilne podatke in izbire. Sami elementi pa morajo biti dobro znani in smiselni, da lahko ljudje predvidijo naslednjo, še ne vidno izbiro.
- *Kombinirani seznami* so uporabni za vnos ali izbiro vrednosti ali pa nastavljanje atributov, ki se med seboj izključujejo. Najbolj so uporabni, kadar uporabnik meni, da je bolj praktično vnašati informacije, le občasno pa je potrebno zagotoviti razpoložljive alternative, ki jih lahko izbere iz seznama. Seznam lahko izvlečemo in spet dvignemo po potrebi, s čimer prihranimo prostor. Koristni so tudi, če so elementi seznama dinamični in spremenljivi, saj omogočajo samostojno dodajanje ključnih elementov.
- *Izvlečni* in *dvižni kombinirani seznami* imajo popolnoma enake lastnosti kot klasični kombinirani seznami. Dodatno omogočajo le to, da

lahko seznam po potrebi izvlečemo in nazaj dvignemo, s čimer prihranimo prostor.

Predstavitveni gradniki

Predstavitveni gradniki so zgolj informativni. Navajajo podrobnosti o drugih gradnikih ali elementih zaslona ali pa pomagajo pri prikazovanju same strukture zaslona. Predstavitveni gradniki so [41]:

- *Namige* uporabljamo za zagotavljanje natančnih opisnih informacij o gradniku, ko je potrebno njegove funkcije hitro prepoznati. Uporabljamo jih tudi za zagotavljanje dodatnih opisnih ali statusnih informacij o določenem elementu zaslona.
- *Obvestila v oblačkih* uporabljamo za prikazovanje ne nujnih informacij ali obvestil, za katere bi sicer potrebovali sporočilno okno. So zelo uporabna za obveščanje uporabnika o nepričakovanem vedenju sistema, nikoli pa na ta način ne prikazujte ključnih informacij.

Ostali specializirani grafični gradniki

Obstajajo tudi drugi bolj specializirani grafični gradniki. Med njimi so [41]:

- *Drsnike z nastavitveno funkcijo* (angl. sliders) uporabljamo za nastavitev atributov, če obstaja neko omejeno število prefinjenih nastavitev. Omogočati morajo enostavno in elegantno povečanje ali zmanjšanje nekega atributa (npr. glasnosti ali svetlosti). Vizualna predstavitev atributa pa mora biti natančna in razumljiva.
- *Drsniki za okna* (angl. scroll bars) so zasnovani za pomikanje po informacijah, zato jih lahko uporabimo, če je na voljo več informacij, kot je prostora za prikazovanje v oknu.
- *Zavihke* lahko uporabljamo za prikaz med seboj neodvisnih podatkov, ki jih je mogoče logično strukturirati v diskretne in pomensko različne strani ali razdelke. Najbolj uporabni so za predstavitev izbir, ki se lahko uporabijo na nekem objektu. Zavihki so uporabni tudi, če vrstni red podatkov ni pomemben.

- *Izbirnike datuma* uporabljamo za izbiro in prikaz enega samega datuma privzetega meseca, prikazanega na spustnem seznamu.
- *Drevesni prikaz* je posebna vrsta seznama, ki prikazuje hierarhično razporejen nabor predmetov. Gradnik uporabljamo za prikaz razmerja med hierarhičnimi elementi in po potrebi vključuje gumbe za razširitev ali strnitev hierarhije.

Aranžiranje grafičnih gradnikov za interakcijo

Vedno skušajte upoštevati Nielsenov princip uporabnosti za estetsko in minimalistično zasnovo. Ta predlaga, da ohranjanje enostavnega dizajna izboljša načrtovanje v vseh dimenzijah uporabnosti [20].

Konflikt med vizualno in funkcionalno preprostostjo

V nekaterih primerih je visok nivo kompleksnosti neizogiben, ker to zahteva naloga ali raven izkušenj uporabnikov. Načrtovanje zaporedja ukazov je za uporabnika lažje, če je način, kako to storiti, dobro viden. V realnosti imajo lahko le najpreprostejši vmesniki po en gradnik na akcijo [10, 11].

Tehnike za doseganje preprostosti so zmanjševanje (manj je več), večkratna vloga gradnikov, dobro grafično načrtovanje in regularnost [30].

Če želite ustvariti uporaben uporabniški vmesnik, morate združiti različne komponente, tako da lahko uporabniki hitro in enostavno dosežejo svoje cilje. To zahteva dobro postavitev. Obstaja več vidikov, ki jih je potrebno upoštevati [30]:

- Smiselno grupirajte gradnike.
- Ločite trenutno aktivne komponente.
- Poudarite pomembne gradnike.
- Zmanjšajte količino nepotrebnih informacij.
- Učinkovito uporabite bele presledke.
- Gradniki naj bodo vidni.
- Uravnotežite estetiko in uporabnost.

Smernice za aranžiranje grafičnih gradnikov

- *Ravnovesje in simetrija.* Enostaven način za doseganje ravnovesja je, da izberete os (po navadi navpično) in enakomerno razdelite elemente okoli osi. Izenačite maso in obseg [2].
- *Grupiranje in hierarhija.* Beli presledki imajo bistveno vlogo pri načrtovanju. Uporabite jih za združevanje namesto vrstic. Okoli vsebine postavite robove. Ko je objekt obdan z belimi presledki, ohranjajte občutek razmerja med objektom in okolico. Ne nizajte gradnikov preblizu skupaj. Prenatrpanost ustvarja prostorsko napetost in znižuje preglednost [10, 30].
- *Poravnava in mreže.* Poravnava prispeva k preprostosti zasnove. Elemente lahko razporedite horizontalno in vertikalno. Slednja je za grafične uporabniške vmesnike primernejša. Oznake lahko konsistentno poravnate z levimi ali desnimi robovi. Ukazni stolpec naj bo poravnan tako na levi kot na desni strani. Učinkovit način za doseganje usklajenosti in ravnotežja skoraj samodejno je mreža [10, 30].

Principi grupiranja šole Gestalt

Obstajajo številna načela oblikovanja, ki jih je razvila šola zaznavanja Gestalt [8]:

- *Bližina.* Eno izmed načel zaznavanja je opažanje, da se predmeti, ki se pojavljajo blizu skupaj v prostoru ali času, pogosto zaznavajo skupaj.
- *Podobnost.* Drugi princip pravi, da so elementi s podobnimi atributi verjetneje grupirani.
- *Zveznost.* Tretji princip navaja, da se nepovezani elementi lahko pogosto štejejo kot del neprekinjene celote.
- *Simetrija.* Naslednji princip namiguje, da so ljudem bolj všeč simetrične oblike. Grafični uporabniški vmesniki, ki ne uporabljajo simetrije, pogosto izgledajo neuravnoteženo oz. kot da se bodo prevrnili na eno stran.

- *Zaprtje.* Ta princip se nanaša na dejstvo, da je zaprte objekte lažje zaznati od tistih, ki so odprti. Ljudje pogosto nezavedno dodajo manjkajoče podatke, zato da bi zaključili podobo, da bi jo lažje zaznali kot celoto.
- *Področje.* Zadnji princip pravi, da ko se dva elementa prekrivata, po interpretaciji manjši element stoji pred večjim elementom.

Izbor teksta, barv, slik in animacije

Tekst

Grafični uporabniški vmesniki nudijo več možnosti za prikaz besedila [31].

- *Čitljivost.* Prepričajte se, da so stavki neposredni in kratki. Odpravite vse nepotrebne besede ali besedne zveze in prehode med odstavki. Pišite za hitro razumevanje, v krajših odsekih po največ štiri ali pet stavkov. Pišite v drugi osebi in uporabite trenutni čas – ta je krajši in bolj neposreden. Izogibajte se pisanju v trpniku. Tvornik ustvari krajše stavke, lažje ga je razumeti in spodbuja odgovornost. Vsa sporočila zapišite pritrdilno. Ne obsojajte in se prepričajte, da sporočilo vsebuje konstruktivne informacije o tem, kaj mora uporabnik narediti [30].
- *Slog.* Izogibajte se "sindromu besedilnega procesorja" (besedilo do skrajnega roba okna) in pustite dovolj prostora. Oblikovna mreža zagotavlja konsistentno postavitev in obseg besedila, stolpcev, naslovov in ilustracij [23].
- *Tipografija.* Izberite različne pisave za prikaz pomembnih kontrastov, sicer pa ohranite preprostost. Ne uporabite več kot dveh ali treh pisav. V okviru tipov, ki ste jih izbrali, lahko uporabite spremembo velikosti, sloga ali barve, da določite potrebne kontraste. Z velikostjo je še posebej enostavno vzpostaviti hierarhijo, kot so naslovi in podnaslovi. Štiri do pet različic pisave bi moralo zadoščati [2].

Vizualne spremenljivke in kontrast

Kontrast kodira informacije vzdolž vizualnih dimenzij. Nanaša se na zaznavne razlike vzdolž vizualnih razsežnosti, kot sta velikost ali barva. Kontrast je oblikovna neznačilnost, ki sporoča informacije ali povzroči izstopanje elementov [30].

- *Značilnosti.* Izbira vizualnih spremenljivk lahko vpliva na uporabnike na dva načina. Prvi, selektivnost, je stopnja, do katere lahko posamezno spremenljivko izberete iz celotnega vidnega polja. Selektivne spremenljivke so vrednost, odtenek, položaj, orientacija in velikost. Drugi, asociativnost, pa se nanaša na to, kako enostavno je prezreti spremenljivko, tako da vse razlike v tej dimenziji izginejo. Asociativne spremenljivke so odtenek, oblika, položaj in orientacija [30].
- *Tehnike za dosego dobrega kontrasta.* Če ste se odločili, da bo v vašem vmesniku bistven kontrast, izberite pravo vizualno spremenljivko, da ga zastopa. Uporabite čim večjo dolžino in izostrite razlike za lažje zaznavanje. Uporabite preizkus s priprtimi očmi, da preverite, ali so kontrasti, ki ste jih poskušali doseči, očitni [30].

Barve

Barve imajo potencial za izboljšanje ali poslabšanje uporabniške izkušnje [41].

- *Barvni modeli.* RGB model je barvni model v obliki kocke, pri kateri (0,0,0) pripada črni barvi, (1, 1, 1) beli barvi, tri dimenzije pa merijo ravni rdeče, zelene in modre barve. Model RGB uporabljajo LCD zasloni. CMYK model je podoben RGB modelu, vendar se uporablja za tiskalne barve, kjer pigmenti ne ustvarjajo, temveč absorbirajo valovne dolžine. Podpira cian modro, magenta rožnato, rumeno in včasih črno barvo. K pomeni ključ. HSV model (odtenek, zasičenost) na najbolj enostaven način prikazuje, kako ljudje dojemajo barvo in je zato najbolj uporaben pri izbiri barv za uporabniški vmesnik. HLS model (odtenek, svetloba, nasičenost) pa je v bistvu simetričen in eleganten dvakratnik

HSV modela, v obliki dvojnega stožca, z enim vrhom bele in drugim vrhom črne barve [30].

- *Barvne kombinacije in vsebovana intenziteta.* Barve lahko razdelimo v več skupin: temne barve, svetle barve, tople barve, hladne barve in druge. Pomemben dejavnik med kombiniranjem barv je njihova vsebovana intenziteta (angl. intrinsic brightness). Za čitljivo besedilo mora obstajati zadosten kontrast med svetlostjo ozadja in barvami v ospredju. Bodite previdni pri uporabi svetle barve, kot je bela ali svetlo rumena, za večje površine zaslona, ker jih je utrujajoče gledati. Dobre alternative so lahko bledo siva barva ter barvi smetane ali magnolije [10].
- *Navodila za uporabo barv.* Ker barvna slepota vpliva na kar nekaj ljudi, je bistveno, da jo upoštevate, ko se odločate o izbiri barv za uporabniški vmesnik. Posredovanje informacij naj ne bo odvisno zgolj od barvnih razlik, še posebej v rdeče-zeleni kombinaciji [30, 31]. Izogibajte se močno nasičenih barv (npr. barv okoli zunanjega roba HSV stožca). Nasičene barve lahko povzročijo utrujenost, ker se morajo oči stalno prilagajati različnim valovnim dolžinam. Namesto tega uporabite manj nasičene (pastelne) barve [10]. Na splošno je priporočljiva zmerna uporaba barv. Vmesnik z veliko barvami deluje zapleten, nastlan in nepregleden. Uporabite manjše število različnih odtenkov ali celo le en odtenek v kombinaciji s črno, belo in odtenki sive barve [3, 8].

- *Barve ozadja* morajo vzpostaviti dober kontrast s tekstom in grafičnimi gradniki v ospredju. Visoko intenzivne barve, kot so rdeča, cian modra, magenta rožnata in živo zelena, povzročajo izčrpanost oči, če jih je potrebno gledati v večjih razsežnostih daljše časovno obdobje. Črno besedilo na belem ozadju je najbolj kontrastno in tudi najlažje berljivo. Izogibajte se tudi ozadju z vzorci [18].
- *Učinkovita uporaba barv.* Najprej oblikujte vmesnik v črno-beli barvi. Na ta način se lahko osredinite na ureditev uporabniškega vmesnika.

Šele nato postopoma dodajte barve. Nekatere smernice ne priporočajo uporabe več kot šestih barv, manj pa je pogosto boljše. Barve lahko uporabite tudi za predstavitev nekaterih informacij [10]:

- Barve za poudarek. Lahko se uporabljajo za poudarjanje pomembnih področij zaslona ali ključnih delov diagrama (tople barve: rdeča, oranžna, rumena ali barvni kontrast: rdeča in zelena).
- Barve za združevanje. Lahko se uporabljajo za organizacijo zaslona (podobne barve: zelena, turkizna, modra).
- Barvno kodiranje. Lahko se uporablja za predstavitev določenega predmeta ali stanja (rdeča za napake in hladne barve za stanje sistema).
- Perspektiva. Barva se lahko uporablja za poudarjanje perspektive na zaslonu (za ozadje temne nenasičene barve in za ospredje svetle nasičene barve).
- Plasti. V povezavi z uporabo perspektive se barva lahko uporabi za predstavitev različnih slojev v diagramu (barvni odtenki).

Slike

Slike lahko uporabimo za motivacijo, pritegnitev uporabnikove pozornosti, zabavo in prepričevanje. Uporabimo jih lahko za posredovanje informacij, zlasti prostorskih predstav ali za premagovanje jezikovnih ovir. Prav tako jih lahko uporabimo kot podporo interakciji (z metaforami in ikonami) [10].

Poznamo naslednje vrste slik [10]:

- Slike. Sem spadajo fotografije, risbe in stripi.
- Diagrami. Ti vključujejo zemljevide in druge predstavitve odnosov med objekti.
- Grafi in grafikoni. To so vizualne predstavitve številk.

Pri načrtovanju in uporabi slik upoštevajte naslednje [10]:

- Izberite najustreznejšo vrsto slike.
- Sliko oblikujete tako, da bo čim bližje zahtevam naloge.
- Upoštevajte vse ustrezne konvencije, da zagotovite konsistentnost z ostalimi slikami te vrste.

- Še posebej učinkovito je združevanje besedila in slik.
- Upoštevajte ločljivost uporabnikovega zaslona, sicer se lahko podrobnosti slike izgubijo.
- Slike, zlasti fotografije, lahko privedejo do zelo velikih datotek in dolgih časov prenosa.

Animacije

Animacijo lahko uporabimo za naslednje namene [10]:

- Za ponazoritev gibanja,
- za zagotovitev dinamične povratne informacije,
- da bi pritegnili pozornost ali
- da bi pokazali delovanje računalniškega sistema.

Čeprav je animacija pomembna in uporabna, jo uporabite le, kadar je to nujno potrebno. Npr. videoposnetek se lahko uporabi za prikaz človekovega vedenja in čustev, dogodkov, za motiviranje ali zagotavljanje dodatnih informacij [10].

Izbor in načrtovanje ikon

Ikone so grafične podobe, ki jih najpogosteje uporabljamo za predstavitev objektov in akcij, s katerimi so uporabniki v interakciji ali pa z njimi manipulirajo. Ikone so lahko prosto stoječe na namizju ali v oknu, lahko pa so grupirane skupaj v orodni vrstici [41]. Razdelimo jih lahko v naslednje kategorije [14]:

- *Ikona.* Nekaj, kar zgleda tako kot tisto, kar predstavja.
- *Indeks.* Znak, ki je bil povzročen s stvarjo, na katero se nanaša.
- *Simbol.* Znak, ki je lahko popolnoma poljuben v podobi.

Značilnosti ikon

Dobro načrtovane ikone [23, 41]:

- zgledajo drugače od ostalih ikon,
- so lahko razpoznavne in razumljive,

- enako dobro izgledajo črno-bele ali v barvah,
- prihranijo prostor zaslona,
- so hitro prepoznavne v "zaposlenem" okolju,
- si jih z lahkoto zapomnimo in
- pomagajo vmesnikom k internacionalizaciji.

Ustvarite razumljive, poznane in konkretne oblike, ki naj bodo vizualno in konceptualno ločljive. Vključite izrazite lastnosti posameznega objekta. Ne prikazujte znotraj meje in jasno odražajte predstavljene objekte, pri čemer se izogibajte pretiranim podrobnostim. Pri načrtovanju skupine ikon ohranjajte konsistentnost strukture, oblike in metodologije načrtovanja. Enako velja za načrtovanje enake ikone v različnih velikostih. Kadar je smiselno, lahko ikoni dodate oznako in s tem zagotovite njen pravi pomen. Razmisliti je potrebno tudi o kontekstu v katerem bodo ikone uporabljene, o pričakovanju uporabnikov in kompleksnosti naloge [41].

Principi načrtovanja ikon

- *Skladnost* oz. *koherentnost.* Skupino ikon načrtujte kot celoto. Ta mora biti dosledna v velikosti, barvah, mataforah in nivoju realizma. Skupinske ikone naj bojo vizualno uravnotežene, vizualne razlike v drobnih dekoracijah pa naj bodo zanemarljive za pomen ikon [11, 23]. Ikone so lahko predstavljene z različnimi nivoji realizma: kot fotografija, risba, karikatura, skica ali silhueta. Držite se samo enega nivoja [23].
- *Čitljivost.* V mejah zmožnosti pri načrtovanju uporabite velike objekte, debele črte in preprost dizajn. Pri tem upoštevajte tudi ločljivost zaslona in razdaljo gledanja. Uporabite dober kontrast med ospredjem in ozadjem. Izogibajte se lokom in poševnim črtam, zunanja oblika pa naj sporoča največ informacij [23].
- *Raje prepoznavanje kot priklic.* Kjer je le mogoče, izberite metaforo, ki je uporabniku znana. Uporabite konkretne objekte, saj se abstraktne koncepte in akcije težko vizualizira. Ikone opremite z besedilnimi oznakami [23].

- *Zmerna uporaba barv.* Najprej načrtujte črno-bele ikone in šele nato barvne. Pretirana uporaba barv lahko uporabnika preobremeni, zato uporabite sive tone ter eno ali dve barvi [23].

Drugi vidiki načrtovanja ikon

- *Kulturna vprašanja in internacionalizacija.* Pazljivo pri uporabi besedila ali črk znotraj ikon (v nasprotju z oznakami), ker bo verjetno potrebno ustvariti ikone za različne jezike. Kretnje, čustveni simboli, videzi obraza ipd. se neznansko razlikujejo od kulture do kulture, zato jih na ikonah ne uporabljajte. Pazite tudi na metafore, ki se nanašajo na posamezne kulture [23].
- *Ikone niso vedno primerne.* Pri visoko abstraktnih konceptih in drobnih razlikah lahko verbalne predstavitve delujejo bolje kot predstavitev z ikonami [23].
- *Jezik ikon.* Za večje skupine ikon je pogosto uporabno razviti jezik ikon. Ta jezik je sistematičen način združitve primitivnih oz. elementarnih simbolov v bolj kompleksne ikone [23].
- *Življenjski cikel načrtovanja uporabnosti ikon*: oblikovanje, testiranje, preoblikovanje. Začnite s preprostimi črno belimi, na roko narisanimi skicami na papirju (silhuete posredujejo največ informacij). Testirajte in ponovno načrtujte dokler osnovni simboli ne delujejo. Dodajte sivine in morda barve. Načrtujte in oblikujte z računalnikom. Natisnite barvne različice v približno realni velikosti. Testirajte in preoblikujte dokler ikone ne delujejo [23].

Povratne informacije in interakcije

Da bi bila povratna informacija za določeno akcijo učinkovita, mora biti uporabniku posredovana znotraj določenega časovnega okvirja [41].

Odzivni čas

Odzivnost sistema naj bo enaka hitrosti in toku človeškega miselnega procesa [41]:

- Če je zahtevano neprekinjeno razmišljanje in si mora uporabnik informacijo zapomniti preko večjega števila odzivov, potem naj bo odzivni čas manjši od ene ali dveh sekund.
- Če se deli nalog sproti zaključujejo, visoka raven koncentracije ni potrebna in so odrejene zmerne kratkoročne spominske zahteve, je sprejemljiv odzivni čas od dveh do štirih sekund.
- Če se naloge v celoti zaključujejo in so odrejene le minimalne kratkoročne spominske zahteve, je sprejemljiv odzivni čas med štirimi in 15 sekundami.
- Če uporabnik medtem lahko počne druge stvari in se vrne, ko mu ustreza, pa je odzivni čas lahko večji od 15 sekund.

Konstantne zamude so boljše od zamud spremenljivk [41].

Delo s časovnimi zakasnitvami

Klik na gumb za potrditev [41]:

- Vse klike gumba potrdite z vidno ali slušno povratno informacijo znotraj ene desetine sekunde.

Čakanje do deset sekund [41]:

- Če operacija potrebuje deset sekund ali manj za zaključek, prikažite znak za zasedenost ali animirano peščeno uro, dokler se operacija ne zaključi.

Čakanje od desetih sekund do ene minute [41]:

- Če operacija potrebuje več kot deset sekund do zaključka, prikažite vrteč krog ali kakšen drug večji animiran objekt.

Čakanje več kot eno minuto [41]:

- Prikažite približno oceno trajanja čakanja.
- Med čakanjem prikažite indikator napredka, odstotek opravljene akcije ali sporočilo o trajanju akcije.
- Ko je operacija končana, prikažite potrditev o zaključku operacije.

Če je operacija zelo časovno zahtevna [41]:

- Razmislite o razdelitvi operacije na več krajših opravil in prikažite indikator napredka za vsako izmed njih.
- Dovolite uporabnikom, da med čakanjem opravljajo druge aktivnosti.

Pritegnitev pozornosti

Utripanje elementa na zaslonu običajno pritegne uporabnikovo pozornost. Če je okno prikazano na zaslonu, naj utripa njegova naslovna vrstica. Če je okno minimizirano, naj utripa njegova ikona. Vendar pazljivo, utripanje je za marsikoga lahko nadležno, zato z njim ne smemo pretiravati [41].

Če želite priskrbeti dodatno indikacijo, da uporabnika čaka neko sporočilo, lahko dodate slušni signal (en ali dva piska). To je uporabno predvsem v primeru, če je okno ali ikona skrita ali pa je uporabnikova pozornost pogosto usmerjena proč od zaslona [41].

Sporočilo prikažite, ko je aplikacija aktivirana ali ko to zahteva uporabnik. Prikazovanje sporočila, ko ga zahteva uporabnik ohrani njegov nadzor nad delovnim okoljem in poskrbi, da se sporočilo po nesreči ne zapre z nenamernim pritiskom gumba [41].

Uporaba zvoka

Zvoki bi morali biti vedno uporabljeni v povezavi z vidnim prikazom. Ne uporabite več kot šest različnih tonov in poskrbite, da uporabniki med njimi razločijo. Tone uporabljajte dosledno, podobne tone za podobne situacije. Ne uporabljajte zvonjenja, melodij in glasnih signalov. Privzeta frekvenca signalov naj se giblje med 500 in 1000 Hz, s tem da uporabniku omogočite, da

glasnost nastavi sam ali pa da ton popolnoma izključi. Zaradi nezanesljivosti je priporočena zmerna uporaba zvokov [41].

Uporaba pogovornih oken

Glede na obvestilo, ki ga pogovorno okno nosi, obstaja več različnih vrst pogovornih oken, kot so npr. okno z vprašanjem, informacijo, potrditvijo, izbiro, opozorilom, napako ali kritično akcijo in ostala prilagojena pogovorna okna, kot je npr. okno za vnos. Pomembno je, da uporabnik po pojavitvi sporočila dobi vse potrebne informacije za nadaljevanje [36].

Pogovorno okno je lahko modalno ali nemodalno. Modalno pogovorno okno zablokira vsa okna istega dokumenta ali aplikacije, razen oken, ki jih ustvari kot lastnik. Modalno pogovorno okno ujame in zadrži fokus okna, dokler je odprto. Zapremo ga običajno s pritiskom gumba kot odgovor na neko stanje. Po drugi strani pa nemodalno pogovorno okno omogoča sočasno upravljanje z drugimi okni [38].

Čarovnik je svetovno znano zaporedje modalnih dialogov, ki se uporablja za izboljšanje naučljivosti zahtevne interakcije. Strukturiran je kot proces koraka za korakom, pri čemer je vsak korak prikazan v pogovornem oknu. Pri čarovniku dialog nadzoruje sistem – narekuje korake, zaporedje korakov in navaja zahteve pri vsakem koraku. Kljub temu, da ima nadzor nad podrobnostimi sistem, pa je še vedno uporabnik tisti, ki proces nadzoruje in potrdi vsak korak posebej [30].

Veliko vmesnikov uporabnike večkrat prekine z vprašanjem, pri katerem je odgovor vedno enak. Ponavljajoče postavljanje enakih vprašanj je neučinkovito, zato je smiselno v pogovorno okno dodati možnost “Ne vprašaj več” [30].

Obvestila o napakah

Preden skušate napisati obvestilo o napaki, najprej poskusite napako odpraviti. Pogosta strategija za preprečevanje napak je potrditveno pogovorno

okno z obvestilom, še boljša izbira pa je reverzibilnost (npr. ukaza razveljavi in obnovi) [30]. Microsoftovi [26] in javini razvijalci [36] navajajo nekaj vodil za oblikovanje dobrih obvestil o napakah:

- Uporabljajte popolne, a preproste stavke.
- Za opis pogojev, ki so privedli do težave ali stanja, ki še obstaja, uporabljajte sedanjik.
- Uporabljajte zvočne signale, kadarkoli je mogoče.
- Izogibajte se uporabi grozečih ali zaskrbljujočih besed (npr. slab, moten, smrten, prekinjen, katastrofalen) ali besed, ki se pričnejo s predpono ne- (npr. nepravilen, neprimeren, nezakonit, nepredvidljiv).
- Ne uporabljajte dvojnih negativov ali avtoritativnega tona.
- Uporabljajte primerne ukazne gumbe, kot so *OK*, *Prekini*, *Da*, *Ne* in *Poskusi znova*. Lahko uporabite tudi kombinacijo teh gumbov. Gumba *Da* in *Ne* morata biti vedno uporabljena skupaj, pred tem mora biti zastavljeno vprašanje.
- Odločite se kdo nadzira pogovor med uporabnikom in sistemom.
- Ne bodite antropomorfni in pokroviteljski. Ne implicirajte, da lahko program oz. strojna oprema razmišlja ali čuti.
- Ne uporabljajte tehničnega žargona, okrajšav, pogovornih besed ali fraz, temveč terminologijo, ki jo uporabniki razumejo. Prav tako ne uporabljajte izrazov, ki bi bili v določenih kulturah lahko žaljivi.
- Izogibajte se uporabi velikih črk za pisanje celih besed in klicajev.
- Uporabnik se ne sme čutiti krivega za napako, čeprav je do težave prišlo zaradi njegove napake.
- Uporabniku ponudite rešitev za nastali problem. Če ima rešitev več kot en korak, ga usmerite na priročnik ali menijski element Pomoč, kjer so koraki podrobneje razloženi.

2.7 Testiranje delovanja in uporabnosti programskega orodja

Testiranje delovanja je priporočljivo že v fazi izdelave prototipov. Dosti lažje je namreč popravljati napake na listu papirja kot kasneje na pravem orodju [20]. Testiranja prototipa se lahko lotimo tako, da določimo vloge vsaj štirim sodelujočim. "Računalnik" je oseba, ki premika delčke, zapisuje odzive in v splošnem počne vse, kar bi počel računalnik. "Posrednik" je glas ekipe razvijalcev, ki vodi testiranje. Uporabniku predstavi vmesnik, razloži navodila ter ga med testiranjem usmerja in spodbuja, da na glas komentira. "Uporabnik" je tisti, ki prototip testira. Vsi ostali sodelujoči so "opazovalci". Njihova naloga je, da so tiho in opazujejo. Spodbudno je, da si pišejo zapiske, ki jih predstavijo šele po zaključku testiranja, ko se lahko razvije debata [30].

Seveda pa je ključnega pomena testiranje končnega izdelka. To lahko izvedemo na več načinov. Vmesnik lahko ocenijo strokovni ocenjevalci, ki izvedejo hevristično vrednotenje, lahko ga preveri razvijalec sam tako, da preveri navodila oblikovanja, lahko ga z uporabniškega vidika pregledajo strokovnjaki, ki se skušajo vživeti v vlogo tipičnega uporabnika, lahko pa celotno osebje organizira sestanek, na katerem skupaj korak za korakom pregledajo vmesnik. Drug način ocenjevanja vmesnika pa je testiranje uporabnosti. To lahko izvedemo s testiranjem pravih uporabnikov, ki rešijo v naprej pripravljene teste uporabnosti, lahko izvedemo primerjavo stare verzije vmesnika z novo, lahko izvedemo študijo primera ali pa kar testiranje na daljavo. Z uporabo ene ali več tehnik testiranja bomo zagotovo dobili dobro povratno informacijo glede uporabnosti programskega orodja, kar pa je ključnega pomena za nadaljnje delo [4, 5].

2.8 Dimenzije uporabnosti

Andrews [23] je mnenja, da s kombiniranjem definicije po ISO standardu in Nielsenove definicije uporabnosti dobimo šest merljivih lastnosti, po katerih lahko ocenjujemo uporabnost nekega produkta. Te lastnosti so: uporabnost, naučljivost, učinkovitost, zapomljivost, varnost in zadovoljstvo.

Uporabnost

Uporabnost meri, kako dobro lahko uporabniki uporabljajo funkcionalnost sistema [11, 30]. Izmerimo jo lahko tako, da določimo temeljne funkcionalnosti, ki uporabnika na najbolj ustrezen način privedejo do uresničitve cilja, nato pa seštejemo (lahko v obliki točk), do kakšne mere je uporabnik te funkcionalnosti izkoristil [23]. Primer: Uporabnik A je dosegel 82 od 100 točk, kar znaša 82 %.

Naučljivost

Naučljivost meri, kako lahko se je naučiti uporabljati sistem [11, 30]. Najlažje jo izmerimo tako, da merimo čas, ki ga nek uporabnik začetnik potrebuje, da dokonča posamezno nalogo. Rezultat bo najbolj pravilen, če v skupino testnih uporabnikov vključimo tako popolne začetnike brez kakršnega koli računalniškega znanja kot tudi začetnike z nekaj osnovnega računalniškega znanja [23]. Primer: Uporabnik A je za 1. nalogo potreboval 45 minut.

Učinkovitost

Učinkovitost meri, kako učinkovit je sistem, ko ga uporabnik že obvlada [11, 30]. Izmerimo jo tako, da določimo nivo strokovnega znanja, nato pa merimo čas, ki ga nek strokovni uporabnik potrebuje, da dokonča posamezno nalogo [23]. Primer: Uporabnik C je za 1. nalogo potreboval 20 minut.

Pomnenje

Pomnenje meri, kako kompleksna je uporaba sistema – ali si je uporabnik, ki že nekaj časa ni uporabljal sistema, zapomnil navodila od zadnje uporabe ali je potreboval osvežitev znanja. Najlažje jo izmerimo tako, da merimo čas, ki ga nek tipičen uporabnik potrebuje, da dokonča posamezno nalogo [23]. Primer: Uporabnik B je za 1. nalogo potreboval 30 minut.

Varnost

Varnost meri, kako pogosto se pojavljajo manjše in hujše napake s strani uporabnikov ali sistema [11, 30]. Najbolj primeren je sistem brez napak. Varnost izmerimo tako, da preštejemo število manjših in hujših napak, do katerih pride med reševanjem določene naloge [23]. Primer: Pri reševanju 1. naloge je 10 uporabnikov skupaj povzročilo 6 večjih in 4 manjše napake.

Zadovoljstvo

Zadovoljstvo meri, kako so uporabniki zadovoljni s samim produktom – ali je orodje prijetno za uporabo, praktično, estetsko privlačno in kakšna je njihova splošna ocena produkta [11, 30]. Najhitreje ga izmerimo tako, da uporabnike po nekajkratni uporabi s kratkim vprašalnikom vprašamo po njihovi subjektivni oceni posameznih lastnosti produkta [23]. Primer: Ali je orodje prijetno za uporabo (5 – zelo prijetno, 1 – popolnoma neprijetno)?

Dimenzije uporabnosti se po pomembnosti razlikujejo od uporabnika do uporabnika, odvisne pa so tudi od samih nalog, zahtev, cene, standardov in zmogljivosti sistema. Razvijalci morajo pri razvoju upoštevati vse atribute, za dosego končnega cilja pa je med njimi pogosto potrebno sklepati kompromise [11, 30].

2.9 Hevristično vrednotenje

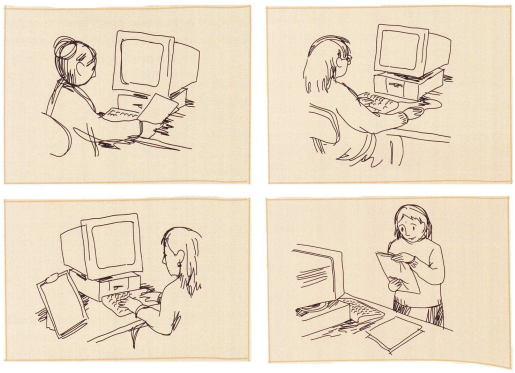
Hevristično vrednotenje je ena izmed raziskovalnih metod pri kateri eden ali več strokovnih ocenjevalcev pregleda in oceni uporabniški vmesnik. Ocenjujejo ga na podlagi vnaprej določenih pravil oblikovanja, npr. po Normanovih namigih, Mandelovih zlatih pravilih, Nielsenovih principih ali katerih drugih [10, 30].

Koraki za izvedbo hevrističnega vrednotenja

S kombinacijo nasvetov, ki so jih navedli Miller, Stone in ostali [10, 30], hevristično vrednotenje izvedemo po naslednjih korakih:

- Izberi pravila oblikovanja.
- Določi strokovne ocenjevalce.
- Pripravi seznam nalog.
- Podrobno preglej uporabniški vmesnik.
- Primerjaj skladnost uporabniškega vmesnika s pravili oblikovanja.
- Oblikuj seznam nepravilnosti in težav.
- Vsak problem podpri z razlago oz. priporočilom za izboljšavo.

Pri vrednotenju in pisanju priporočil se zelo pozna, če so strokovni ocenjevalci seznanjeni s pravili oblikovanja in jih znajo pravilno interpretirati ter uporabljati [4, 5]. Iz tega razloga so Stone in ostali [10] mnenja, da je inšpektor lahko kdorkoli: strokovnjak na področju uporabnosti, lahko je strokovnjak domene, ki odlično pozna program (sem uvrščamo uporabnike in predstavnike uporabnikov), lahko so to izkušeni oblikovalci, razvijalci, ki jih zanima uporaba vmesnika z vidika uporabnikov, v redkih primerih pa tudi nestrokovnjaki (med te spadajo prijatelji, znanci, sodelavci ali člani družine). Kot prikazuje slika 2.36, pa je vsekakor priporočljivo uporabiti več različnih ocenjevalcev, saj bodo našli različne probleme [11, 30]. Nielsen, ki je hevristično vrednotenje tudi zasnoval, predlaga uporabo od treh do petih različnih ocenjevalcev [30].



Slika 2.36: Več različnih ocenjevalcev najde več različnih problemov.

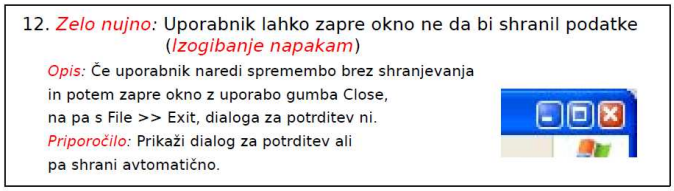
Nasveti za oblikovanje hevrističnega poročila

Vsak problem je potrebno povezati z ustrezno hevristiko. Ni dovolj, da le opišemo problem, potrebno ga je tudi utemeljiti z razlogom. Posamezen element vmesnika ima lahko več neskladnosti, vendar je vsakega izmed problemov treba opisati v ločenem razdelku. Problemi se namreč med seboj razlikujejo po stopnji nujnosti. Prav tako je vmesnik smiselno pregledati najmanj dvakrat. Enkrat za pridobitev občutka o sistemu in ponovno za osredotočenje na posamezne gradnike vmesnika [11, 30].

Kot prikazuje slika 2.37, je predlagani format poročila sestavljen iz [11, 30]:

- Problema uporabnosti,
- pripadajoče hevristike,
- opisa problema,
- prioritete problema,
- priporočila (če je) in
- slike (če pomaga).

Stopnjo nujnosti lahko določimo s pomočjo dejavnikov, ki vplivajo na prioriteto problema uporabnosti. Vprašati se moramo, kako pogosto se problem pojavlja in kako težko je problem zaobiti. Glede na resnost napake ločimo



Slika 2.37: Predlagani format poročila.

štiri kategorije: zelo nujno, nujno, pomembno in manj pomembno. Zelo nujne probleme moramo odpraviti nemudoma, po možnosti še preden je produkt izdan. Nujne napake je potrebno odpraviti čim prej, saj imajo visoko stopnjo prioritete. Pomembne napake se lahko odpravijo nekoliko kasneje, saj imajo nižjo stopnjo prioritete. Manj pomembnih napak ni potrebno odpraviti, saj so to po večini le kozmetični popravki in manjši detajli, ki bistveno ne vplivajo na delovanje vmesnika [11, 30].

Poročilo naj vključuje tako pozitivne komentarje kot tudi kritike. Imejte v mislih, da razvijalci niso nujno strokovnjaki uporabnosti in lahko kritiko vzamejo zelo osebno. Komentarji naj bodo zapisani taktno, natančno in konstruktivno [11, 30].

Analiza in interpretacija rezultatov hevrističnega vrednotenja

Za analizo in interpretacijo rezultatov je potrebno zbrati vsa poročila strokovnih ocenjevalcev. Ker se njihova mnenja med seboj pogosto razlikujejo, bi bilo dobro, če bi vsa poročila pregledali skupaj, jih pokomentirali in uskladili mnenja. Enako velja za predlagana priporočila [10].

2.10 Testiranje uporabnikov

Test uporabnosti je po ISO standardu postopek testiranja uporabnikov, nalog in okolja z namenom vrednotenja zmogljivosti, iskanja pomanjkljivosti in napak uporabniškega vmesnika z ozirom na uporabnikovo učinkovitost in zadovoljstvo. Lahko ga izvedemo na papirnem prototipu ali pa, ko že imamo

končno aplikacijo[11, 31].

Testiranje uporabnikov s pravimi uporabniki pa je temeljna in nenadomestljiva metoda, ki priskrbi direktno informacijo o tem, kako ljudje uporabljajo računalnik in kakšne težave imajo z uporabo konkretnega uporabniškega vmesnika [20]. Med testiranjem testiranec rešuje dane naloge, opazovalec oz. preizkuševalec pa ga med tem diskretno opazuje ter zapisuje testirančeve komentarje, opazke in morebitne težave z reševanjem [10].

Koraki za izvedbo testiranja uporabnikov

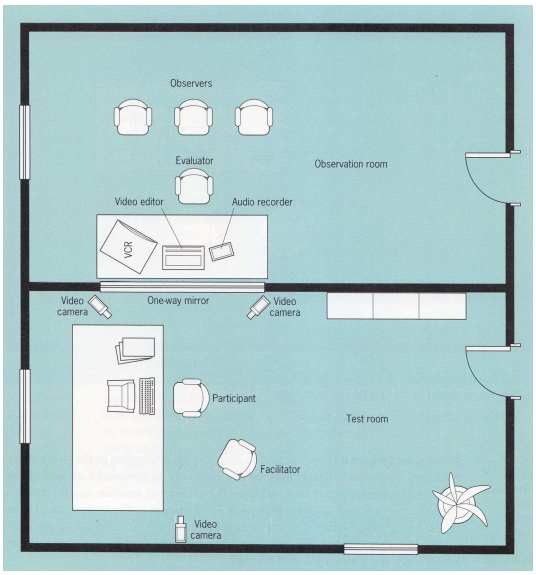
Da bi izvedli test, ki nas bo preskrbel s pomembnimi in uporabnimi podatki, moramo upoštevati naslednje korake[23]:

- Naredi načrt.
- Določi sodelujoče.
- Pripravi vse potrebne materiale in prostor.
- Izvedi poskusni test.
- Izvedi pravo testiranje.
- Analiziraj rezultate.

Izdelava načrta

Pri načrtovanju testa je potrebno določiti posamezne enote: cilje, ki bi jih želeli doseči, sredstva, ki jih imate na voljo, morebitne materiale za potrebe testa, vrsto interakcije s subjekti ter čas in okolje v katerem bo potekal test. Cilje in sredstva po navadi določi naročnik testiranja, z njimi pa so pogojeni tudi materiali, ki jih bomo uporabili. Za obsežnejše testiranje lahko najamemo studio ali laboratorij, ki ga opremimo z najrazličnejšo tehnologijo in kamerami za izčrpno spremljanje uporabnikovega dela in obrazne mimike, kot prikazuje slika 2.38. Manjša oz. skromnejša testiranja ne zahtevajo nič več kot računalnik in list papirja. Glede na izbrano nato določimo čas in lokacijo testiranja. Kot že omenjeno, lahko test izvedemo v laboratoriju ali pa v delovnem okolju, kjer se testiranec verjetno počuti bolj sproščeno [20, 31]. Na koncu določimo še vrsto interakcije s testiranci. Reševanje posameznih

nalog je lahko v celoti prepuščeno testirancem (nizka interaktivna tehnika), lahko pa jih preizkuševalec nekoliko usmerja, jih ob reševanju sprašuje ali jim svetuje do te mere, da ne vpliva na rezultat testiranja (visoka interaktivna tehnika) [31].



Slika 2.38: Nadzorovano testiranje uporabnikov v laboratoriju za testiranje uporabnosti.

Določitev sodelujočih

Landauer in Nielsen sta v raziskavi dokazala, da lahko samo pet skrbno izbranih uporabnikov najde 85 % težav uporabnosti. Za natančnejše rezultate pa lahko uporabimo 15 testirancev, ki naj bi našli kar 99 % vseh problemov v eni sami iteraciji. S sredstvi, ki jih imamo na voljo, lahko torej izvedemo enkratno testiranje s 15 udeleženci ali pa trikratno testiranje s petimi [21]. Vsekakor pa je dobro, če so karakteristike testirancev čim bolj raznolike. Pomembne so predvsem razlike v starosti, spolu, narodnosti, fizičnih zmožnostih in nezmožnostih, izobrazbi, izkušnjami z računalnikom in tehnologijo, motivaciji in odnosu do dela [10, 31].

Uporabniki med testiranjem doživljajo stres, čeprav testiranje predstavlja minimalno psihično ali fizično tveganje. Počutijo se, kot da testiramo njihovo inteligenco, da jih primerjamo z drugimi uporabniki in da bodo izpadli slabo. Glavno pravilo je, da testirance spoštujemo kot individualne osebe s svobodno voljo in čustvi. Poskrbeti moramo, da bo testirancu udobno in da ne tratimo njegovega časa. Vse potrebno za izvršitev testa pripravimo že prej. Spoštovati moramo njegovo zasebnost in večkrat poudariti, da lahko test prekinemo, kadarkoli bo uporabnik sam želel [11, 30].

Priprava materialov in prostora

Sestavljanje vprašalnika naj bo premišljeno in pregledano s strani sodelavcev. V začetku naj bo jasno zapisan namen testiranja skupaj z navodili za reševanje. Sledi naj rubrika s karakteristikami, ki jo izpolnijo testiranci sami (npr. spol, starost, izobrazba, delovno mesto, delovna doba, funkcije, izkušnje z računalniki in seznanjenost z dotičnim programom). Jasno naj bo izpostavljeno tudi, da je reševanje anonimno, da se je uporabnik prostovoljno odločil za testiranje in da lahko test kadarkoli prekine. Za tem naj sledi sklop nalog z razumljivimi navodili [4, 5]. Vsaka naloga naj vsebuje čim bolj realistično situacijo, da bo imel uporabnik občutek, kot da je običajen delovni dan. Naloge naj bodo različno težke in naj usmerjajo uporabnika k iskanju rešitev [31]. Po zaključku vsake naloge lahko sledi nekaj vprašanj glede težavnosti naloge, morebitnih težav, opazk in predlogov. Na koncu vprašalnika pa naj bo poleg zahvale za sodelovanje še nekaj dodatnih vprašanj glede vmesnika na splošno, kakšen se jim zdi, ali je prijeten za uporabo, kaj bi na njem pohvalili ali pokritizirali, kakšni so njihovi predlogi in njihova splošna ocena [23].

Preizkuševalec naj si pripravi tudi spremljevalni obrazec, ki ga bo izpolnjeval med testiranjem. Ta naj vsebuje vse elemente, ki jih pri testirancu želi opazovati (npr. čas izvedbe posamezne naloge, opazke, akcije in reakcije uporabnika, rabo pomoči, komentarje in druge). Preizkuševalec si lahko določi tudi posebne znake, ki jih testiranec ne bi prepoznal, v primeru, da bi

pokukal na njegov obrazec [23].

V fazi priprave na test, mora preizkuševalec poskrbeti, da je prostor pripravljen, da so računalniki prižgani in v delujočem stanju ter da so vsi materiali, navodila in vprašalniki dostopni. Po potrebi preveri delovanje kamer in drugih naprav za snemanje ter odstrani kakršne koli znake predhodnega testiranja [20].

Izvedba poskusnega testa in pravega testiranja

Izvedba poskusnega testa je zelo pomembna. Poskusni test mora biti pravi test, opravljen mora biti v istem okolju, kot bo potekalo pravo testiranje, izvaja pa naj ga resnični uporabnik (ne preizkuševalec sam). Med reševanjem lahko preizkuševalec preveri vso opremo, različne scenarije in izmeri čas. Tako bo dobil dobro povratno informacijo, ali je na izvedbo testiranja resnično pripravljen [31].

Pravo testiranje uporabnikov je intenzivno. Če se testiranje snema, naj se preizkuševalec osredotoči predvsem na opazovanje uporabnika, če pa snemanja ni, so potrebni izčrpni zapiski vseh opažanj. Lažje je, če testiranje spremljata dve osebi in je ena zadolžena za pisanje zapiskov, druga pa ima kontrolo nad testirancem, opremo in vsem drugim. Weinschenk [31] predlaga, da naj testiranje uporabnikov poteka po naslednjem postopku:

- Pozdravi udeleženca.
- Predstavi se.
- Zakrij besedilo spremljevalega obrazca.
- Udeleženec naj izpolni podatke o svojih karakteristikah.
- Izvedi testiranje.
- Udeleženec naj odgovori na vsa preostala vprašanja na koncu vprašalnika.
- Z udeležencem izvedi kratek intervju.
- Zahvali se udeležencu.

Analiza in interpretacija rezultatov testiranja uporabnikov

Pri testiranju uporabnikov lahko nastane ogromno dokumentacije, kot prikazuje slika 2.39.

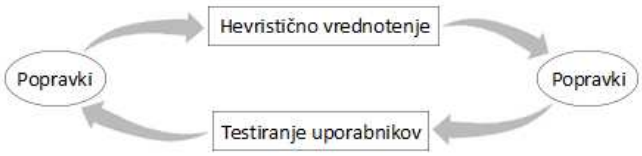
Slika 2.39: Pri testiranju uporabnikov lahko nastane ogromno dokumentacije.

Vse posnetke, vprašalnike in obrazce je potrebno zbrati in jih ustrezno označiti. Najlažje je, če si oblikujemo nek sistem shranjevanja (npr. po ločenih in označenih mapah ali škatlah). Nato je potrebno določiti kriterije točkovanja, vse vprašalnike pregledati in oceniti. Uporabnost vmesnika lahko ocenimo s pomočjo dimenzij uporabnosti. Z njimi ocenjujemo merljive lastnosti, kot so uporabnost funkcij, naučljivost uporabe, učinkovitost delovanja, zapomljivost postopkov, občutek varnosti in pojavljanje napak ter najpomembnejše - zadovoljstvo uporabnikov. Na podlagi dokumentacije lahko napišemo izčrpno poročilo, v katerega vključimo tabele in grafe, ki predstavljajo naše ugotovitve. V poročilu ne smejo manjkati tudi komentarji in predlogi uporabnikov [10].

2.10.1 Primerjava hevrističnega vrednotenja in testiranja uporabnikov

Najbolj priporočljivo in tudi učinkovito je iterativno hevristično vrednotenje in testiranje uporabnikov, kot kaže slika 2.40. Tako lahko dobimo najboljše

rezultate, saj vsaka metoda najde različne probleme [11].



Slika 2.40: Najbolj učinkovito je iterativno hevristično vrednotenje in testiranje uporabnikov.

Pri obeh testiranjih je lahko prisoten opazovalec. Če se ocenjevalcu hevrističnega vrednotenja nekje zalomi, mu opazovalec lahko pomaga, vendar le v primeru, da je ocenjevalec že opazil dotični problem uporabnosti. Z nasvetom lahko prihranimo veliko časa in ocenjevalec lahko hitro nadaljuje z delom. Pri testiranju uporabnikov pa bi bila tovrstna pomoč popolnoma neustrezna. Opazovalec ali preizkuševalec, kot smo ga poimenovali v našem primeru, želi izvedeti, kako se bo testiranec soočil s težavo. Na ta način poustvarimo resnično situacijo, v kateri bi se testiranec znašel sam, brez pomoči. Kadar testiranec obtiči do te mere, da sam ne zna poiskati rešitve, je po navadi potrebno nalogo opustiti in se lotiti naslednje [20, 30].

Hevristično vrednotenje se koncentrira na pregledovanje uporabniškega vmesnika navzven. Gre za ocenjevanje razporeditve in konsistentnosti posameznih gradnikov, pojavnih oken, jezika, barv, pisav, dostopne pomoči, morebitnih bližnjic, vidljivosti statusa sistema, reševanje napak in na splošno estetski vidik uporabniškega vmesnika. S hevrističnim vrednotenjem pogosteje odkrijemo tudi probleme s Fittsovim zakonom. Po drugi strani pa testiranje uporabnikov temeljiteje razišče programerski oz. notranji vidik vmesnika. Ocenjujejo se merljive lastnosti, kot so uporabnost, naučljivost, učinkovitost, zapomljivost, varnost in zadovoljstvo. Na ta način dobimo vredne informacije o tem, ali je vmesnik enostaven in prijeten za uporabo, ali zadošča uporabnikovim potrebam ter kaj bi lahko spremenili, da bo delo z njim še lažje, učinkovitejše in prijetnejše [11, 30].

Poleg želje po izboljšavi in težnje po boljših rezultatih, pa na izvedbo testiranja po navadi vplivajo tudi dejavniki kot so čas ali sredstva, zaradi česar se naročniki velikokrat odločijo le za enkratno testiranje uporabnikov in morda nekajkratno hevristično vrednotenje. Slednje je namreč občutno cenejše in časovno manj potratno [11].

2.11 Razvoj programskega orodja za asistenco pri načrtovanju in vrednotenju grafičnih uporabniških vmesnikov

2.11.1 Pregled obstoječega stanja, iskanje in prebiranje literature

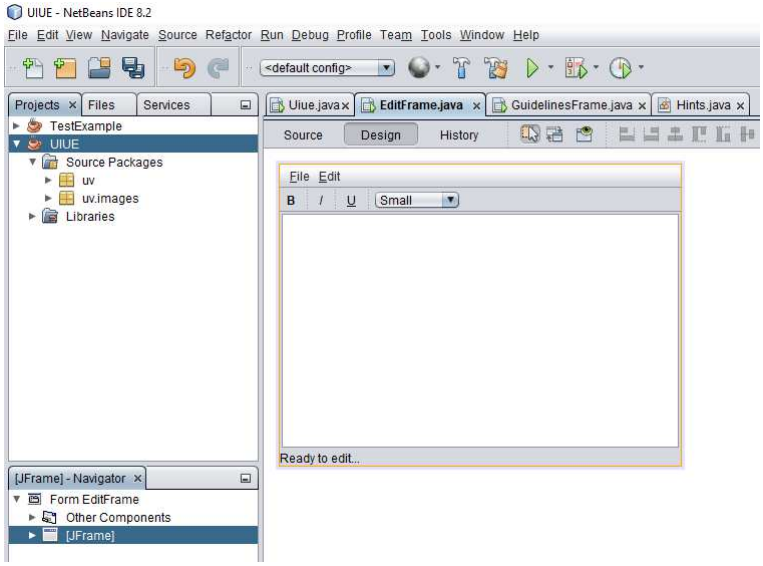
Med dosegljivo literaturo je na voljo ogromno navodil, principov in standardov, po katerih naj bi bili oblikovani dobri uporabniški vmesniki. Na svetovnem spletu lahko najdemo tudi nekaj spletnih strani, kjer ponujajo svoje usluge ali organizirajo tečaje samoocenjevanja uporabniških vmesnikov. Nikjer pa nismo zasledili programskega orodja ali aplikacije, ki bi združevala vsa ta navodila, principe in standarde na enem mestu. Orodja, s katerim bi si uporabnik lahko pomagal pri samem oblikovanju ter ocenjevanju končnega uporabniškega vmesnika.

2.11.2 Zbiranje idej

Po pregledu obstoječega stanja na trgu in obisku knjižnice smo prišli na idejo, da bi takšno programsko orodje lahko oblikovali sami. Tovrstno orodje bi zajemalo večje število navodil, principov in standardov oblikovanja, hkrati pa ocenjevalcu služilo v pomoč pri ocenjevanju uporabnosti nekega grafičnega uporabniškega vmesnika.

Kot primer dobre prakse smo vzeli NetBeans, integrirano razvojno okolje za programski jezik Java [28], ki ga prikazuje slika 2.41. NetBeans je

oblikovan zelo premišljeno in ima veliko karakteristik dobro oblikovanega uporabniškega vmesnika.



Slika 2.41: NetBeans, integrirano razvojno okolje za programski jezik Java.

Z obetavno idejo v mislih, primerom dobre prakse ter vso dostopno literaturo glede principov, navodil in standardov oblikovanja grafičnih uporabniških vmesnikov, smo se lotili izdelave prototipov.

Poglavje 3

Rezultati

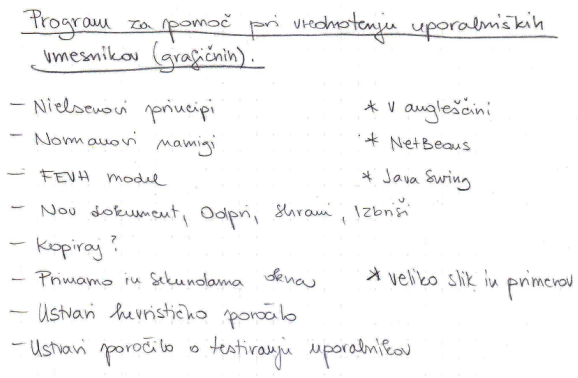
3.1 Programsko orodje za asistenco pri načrtovanju in vrednotenju grafičnih uporabniških vmesnikov

3.1.1 Izdelava prototipov

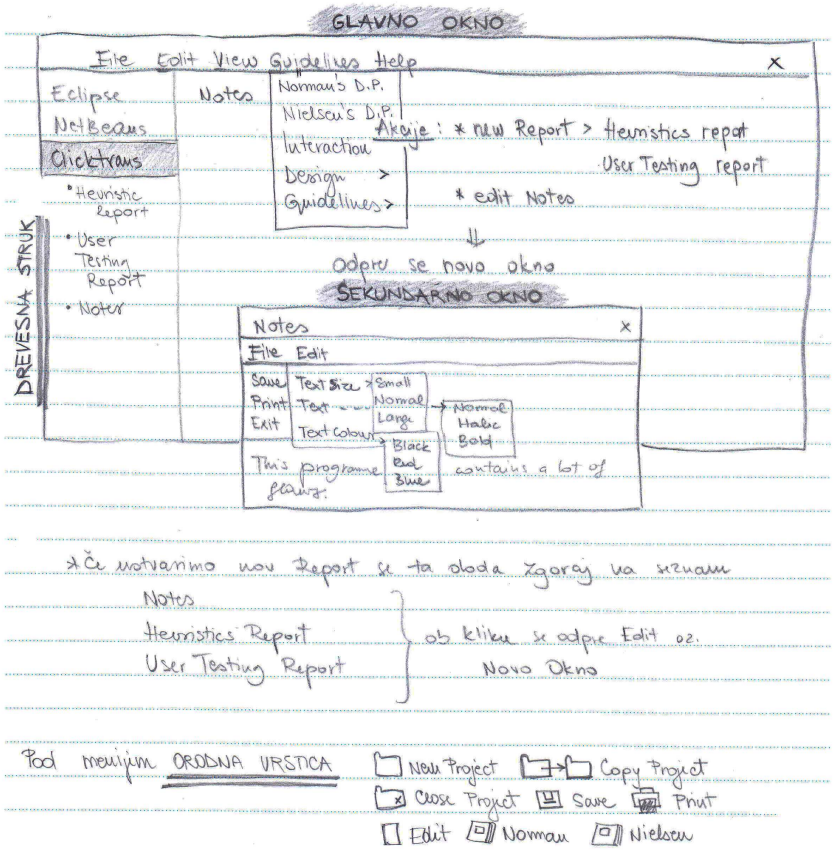
Med zbiranjem idej smo določene izbire in akcije vmesnika določili verbalno. Kot prikazuje slika 3.1 smo sestavili seznam funkcij, ki bi jih vmesnik lahko podpiral. Naše osnovne ideje so zajemale: pomoč uporabniku z naštetimi Nielsenovimi principi in Normanovimi namigi; osnovne funkcije, kot so nov dokument, odpri, shrani in izbriši; možnost oblikovanja hevrističnega poročila in poročila o testiranju uporabnikov.

V fazi izdelave enostavnega papirnatega prototipa, smo skicirali glavno okno, sekundarna okna, menije in orodno vrstico ter predvideno vsebino posameznih elementov. Dodali smo tudi drevesno strukturo odprtih dokumentov na levi in prikazovalno okno na desni ter nekaj dodatnih navodil oblikovanja, ki so se nam zdela ključna za prikaz. Papirnati prototip, ki smo ga skicirali prikazuje slika 3.2.

Nazadnje smo pri izdelavi interaktivnih skic prejšnje ideje še nadgradili

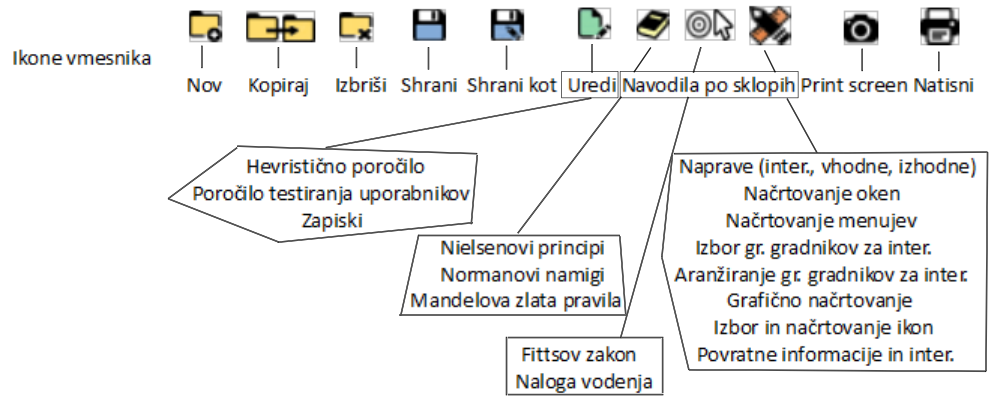


Slika 3.1: Verbalni prototip.



Slika 3.2: Enostaven papirnati prototip.

in dodali še barve, opombe in vodila, ki so nam bila v oporo pri izdelavi končnega produkta. Del interaktivnih skic prikazuje slika 3.3.



Slika 3.3: Interaktivne skice: ikone z barvami in opisi ter povezave na elemente, ki se odprejo ob kliku nanje.

3.1.2 Izdelava programskega orodja

"User Interface Usability Evaluator" (v nadaljevanju UIUE), kot smo ga poimenovali, je programsko orodje za asistenco pri načrtovanju in vrednotenju grafičnih uporabniških vmesnikov. Izdelali smo ga v razvojnem okolju NetBeans, saj poleg Jave podpira tudi standardni označevalni jezik za izdelavo spletnih strani - HTML [42] in orodje za oblikovanje vizualnih gradnikov grafičnih uporabniških vmesnikov za programski jezik Java - Swing [43].

Funkcionalnost

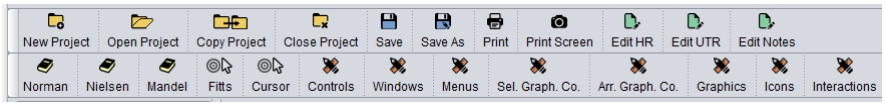
UIUE je sestavljen iz štirih razredov in združuje več kot 6000 vrstic programske kode. Jezik programa je angleščina, saj smo ocenili, da bi program lahko uporabljali v različnih državah po vsem svetu. Vsa okna, glavno in sekundarna, so oblikovana nemodalno, tako da ima uporabnik lahko odprtih

več oken hkrati, npr. na eni strani hevristično poročilo, na drugi strani pa navodila za pisanje hevrističnega poročila.

Razred UIUE, ki vsebuje glavno (angl. main) metodo, je napisan v Javi in oblikovan s Swingom. Glavno okno sestavljajo različni gradniki, ki so po mreži smiselno razporejeni v razmerju zlatega reza, ki je očesu prijazen in ga pogosto najdemo v naravi [2]. Da bi bil vmesnik čim bolj enostaven za uporabo in uporabniku razumljiv, so posamezni gradniki postavljeni drug ob drugega oz. po skupinah po vsesplošnem konceptu združevanja, kakor priporoča Raskin [22]. Spustni meniji vrstičnega menija so oblikovani po modelu FEVH (angl. File, Edit, View, Help), vsak meni pa ima smiselno nanizane elemente z mnemoniki in pospeševalci, kjer so primerni.

Skoraj vsi elementi menijev so v obliki gumbov z ikonami smiselno razporejeni v dve dinamični orodni vrstici, ki ju prikazuje slika 3.4. Gumbi v zgornji orodni vrstici omogočajo izdelavo novega projekta (gumb New Project), odpiranje obstoječega projekta (gumb Open Project), kopiranje projekta (gumb Copy Project), zapiranje projekta (gumb Close Project), shranjevanje (gumb Save), shranjevanje pod drugim imenom (gumb Save As), tiskanje (gumb Print), posnetek zaslona (gumb Print Screen), urejanje hevrističnega poročila (gumb Edit HR), urejanje poročila o testiranju uporabnikov (gumb Edit UTR) in urejanje zapiskov (gumb Edit Notes). V spodnji orodni vrstici pa so nanizana navodila za oblikovanje grafičnih uporabniških vmesnikov. Po vrsti si sledijo: Normanovi namigi (gumb Norman), Nielsenovi principi (gumb Nielsen), Mandelova zlata pravila (gumb Mandel), Fittsov zakon (gumb Fitts), naloga pomikanja oz. vodenja (gumb Cursor), izbor naprav za interakcijo (gumb Controls), načrtovanje oken (gumb Windows), načrtovanje menijev (gumb Menus), izbor grafičnih gradnikov za interakcijo (gumb Sel. Graph. Co.), aranžiranje grafičnih gradnikov za interakcijo (gumb Arr. Graph. Co.), izbor teksta, barv, slik in animacije (gumb Graphics), izbor in načrtovanje ikon (gumb Icons) ter povratne informacije in interakcije (gumb Interactions).

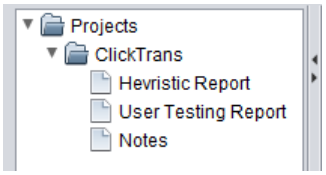
Ikone smo oblikovali tako, da smo upoštevali Hortonov seznam, ki ga pri-



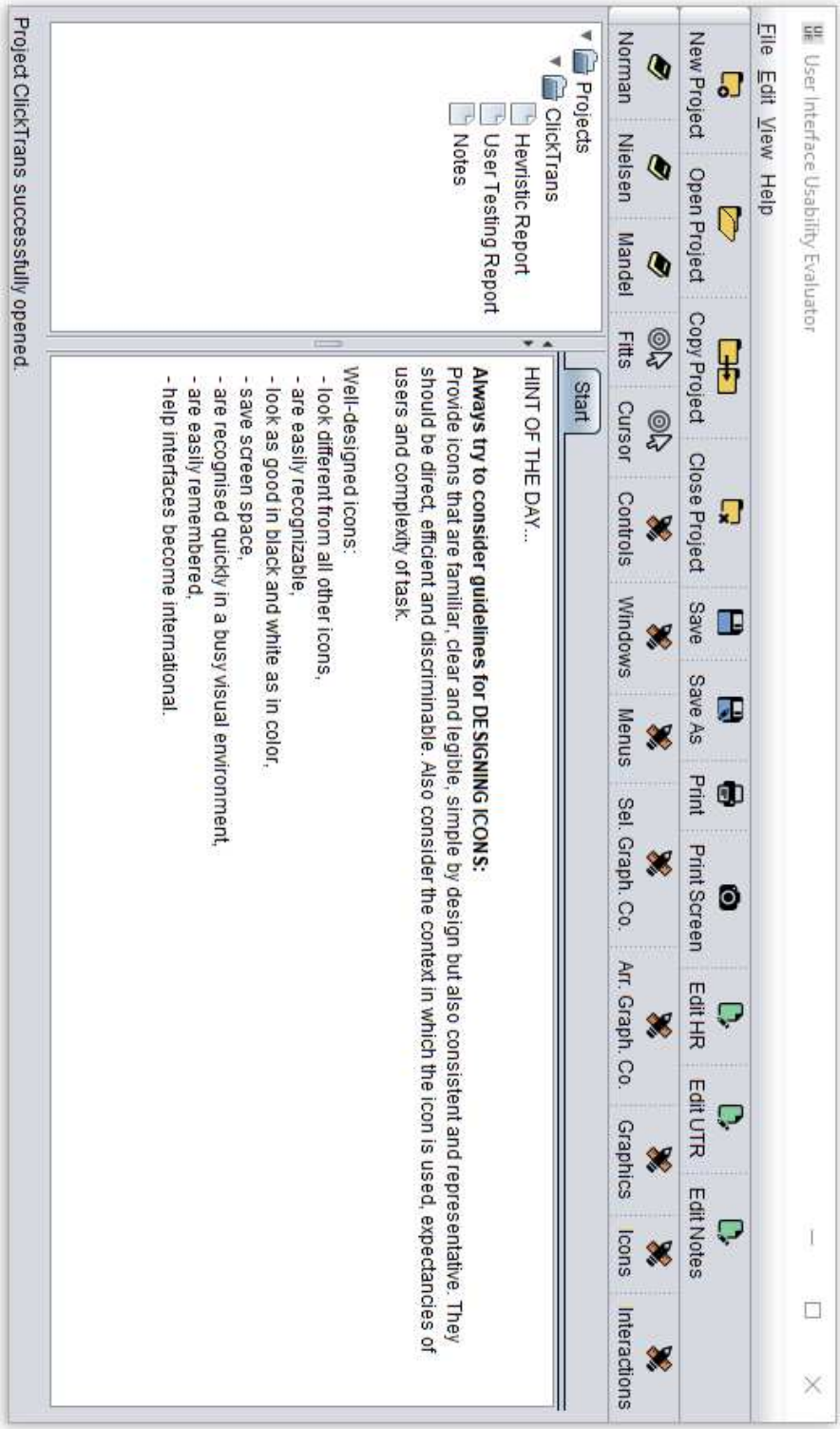
Slika 3.4: Orodni vrstici z gumbi in edinstveno oblikovanimi ikonami.

poroča Benyon [8]. Slike so razumljive in uporabniku znane, opremljene z oznako, zapomljive in estetsko oblikovane. Posamezne akcije, ki se med seboj popolnoma razlikujejo nosijo edinstvene slike, med tem ko si skupine navodil za oblikovanje in gumbi za urejanje datotek delijo enake slike. Uporabnik lahko jasno razlikuje med posameznimi gumbi in skupinami gumbov tudi s pomočjo skrbno izbranih barv, ki se med seboj lepo dopolnjujejo. Po navodilih različnih priznanih avtorjev [10, 31, 41] smo v celotnem orodju uporabili le pet različnih barv v različnih odtenkih, ki smo jih kombinirali s črno pisavo ter sivo-belim ozadjem.

Pod orodnima vrsticama je okno razdeljeno na polje z drevesnim prikazom (angl. tree view), ki hierarhično prikazuje trenutno odprte datoteke, kot kaže slika 3.5 in polje, ki s pomočjo zavihkov prikazuje vsebino izbrane odprte datoteke. Ob vsakem prvem zagonu programa se avtomatsko odpre zavihek Start, na katerem je izpisan naključno izbran namig dneva. Polji sta ločeni s premičnim separatorjem, tako da si uporabnik sam lahko nastavi ustrezno velikost. Na dnu okna je še statusna vrstica, v kateri se izpiše vsaka uporabnikova akcija. Kakor svetuje Galitz [40], je povratna informacija hitra in jasna, napisana v kratkih, jedrnatih in uporabniku razumljivih stavkih. Slika 3.5 prikazuje glavno okno z vsemi omenjenimi gradniki.

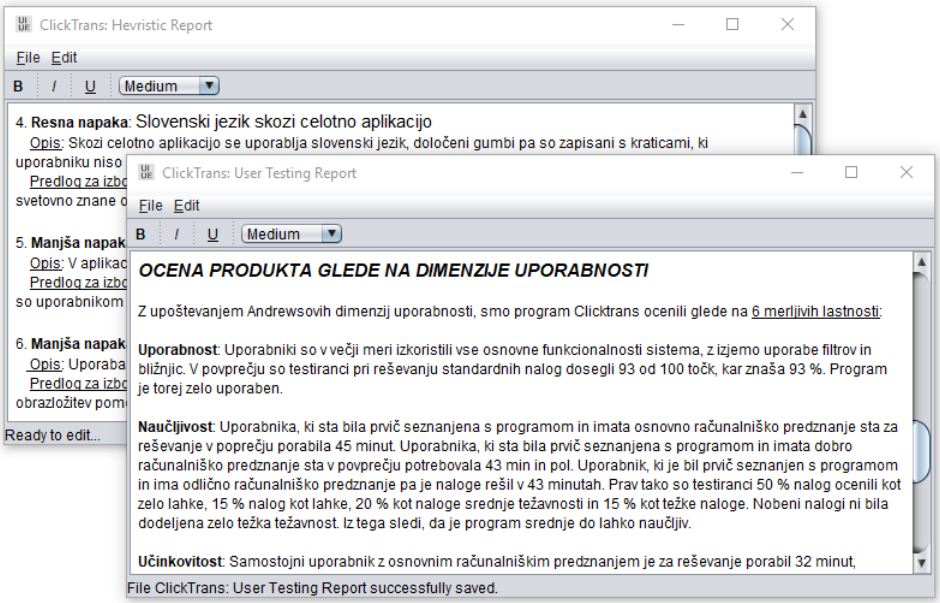


Slika 3.5: Polje s hierarhičnim prikazom trenutno odprtih datotek.



Slika 3.6: Glavno okno z odprtim začetnim zavihkom.

Ob kliku na Edit Hevristic Report (gumb Edit HR), Edit User Testing Report (gumb Edit UTR) ali Edit Notes se odpre novo sekundarno okno, v katerem lahko uporabnik ureja in oblikuje vsebino dokumenta. Razred EditFrame je tako kot glavni razred napisan v Javi in oblikovan s Swingom. Kot kaže slika 3.7, okno sestavlja vrstični meni s spustnima menijema File in Edit, pod njim pa je orodna vrstica z gumbi, ki jih predstavljajo vsesplošno znane ikone za krepko (gumb B), ležeče (gumb I) in podčrtano besedilo (gumb U) ter spustni seznam, ki omogoča izbiro treh velikosti pisave: majhno, srednjo in veliko. Pod orodno vrstico je polje z besedilom in pod njim statusna vrstica.

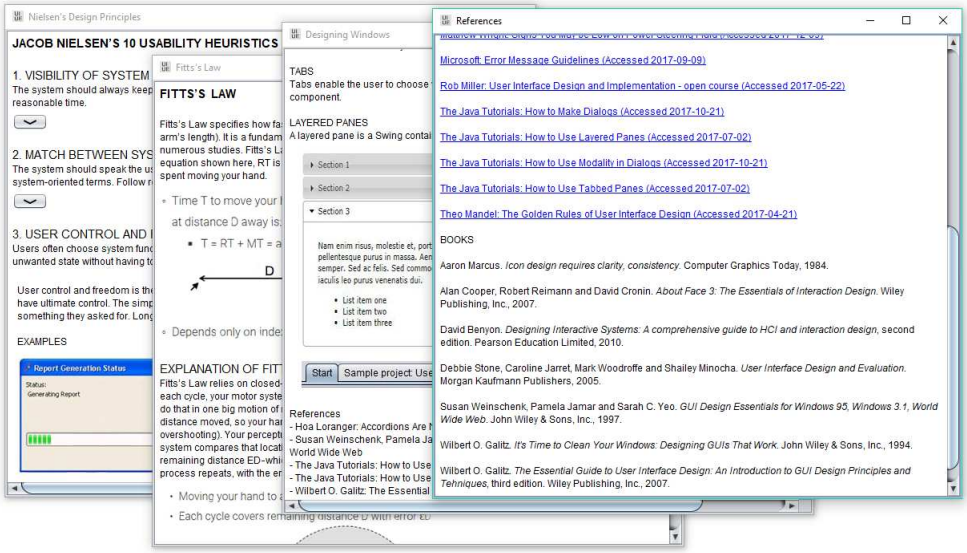


Slika 3.7: Sekundarni okni za urejanje dokumentov.

Razred GuideslinesFrame je napisan v Javi in oblikovan s kombinacijo Swinga in Oraclove odprte kode [29], ki smo jo preoblikovali za lastne potrebe. Ob kliku na katerikoli element spustnega menija View, ki se nahaja na menijski vrstici glavnega okna programa ali pa na katerikoli gumb spodnje orodne vrstice, se odpre novo sekundarno okno z navodili za oblikovanje. Ta so razdeljena v štiri sklope:

- navodila za pisanje poročil hevrističnega vrednotenja in testiranja uporabnikov,
- Normanovi namigi, Nielsenovi principi in Mandelova zlata pravila oblikovanja,
- interakcije – Fittsov zakon in naloga pomikanja oz. vodenja ter
- vsesplošna navodila načrtovanja grafičnih uporabniških vmesnikov.

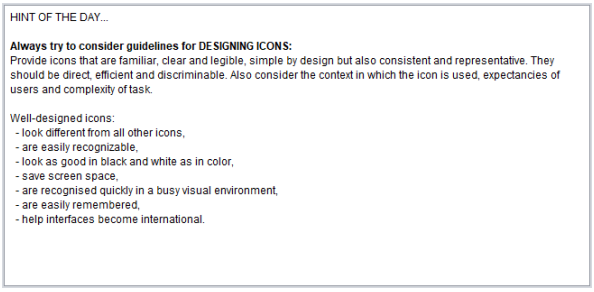
Navodila se dotikajo različnih področij, kot so načrtovanje menijev in oken, izbor in aranžiranje grafičnih gradnikov, oblikovanje ikon in drugih. Vsaka navodila vsebujejo strnjene nasvete za oblikovanje dobrega grafičnega vmesnika, opremljena pa so tudi s primeri in slikami. Vsi uporabljeni viri so v skrajšani obliki zapisani na koncu vsakega sestavka, celostne zapise pa dobimo s klikom na References, element spustnega menija Help, ki je na menijski vrstici glavnega okna programa. Nekaj primerov navodil in uporabljeni viri so prikazani na sliki 3.8.



Slika 3.8: Navodila za oblikovanje grafičnih uporabniških vmesnikov in seznam z vsemi uporabljenimi viri.

Že omenjeni namigi dneva so kratki povzetki navodil oblikovanja grafičnih

uporabniških vmesnikov in navodil pisanja poročil hevrističnega vrednotenja ter testiranja uporabnikov. Zapisani so v razredu Hints, oblikovani pa so s kombinacijo Jave in HTML. Ob vsakem zagonu program naključno izbere in prikaže enega izmed 15 namigov. Na sliki 3.9 je prikazan eden izmed namigov.



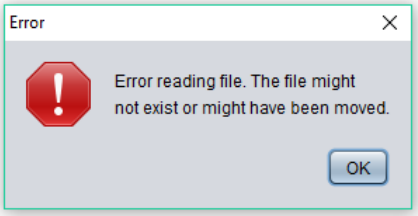
Slika 3.9: Namig dneva.

Obvestila in sporočila o napakah

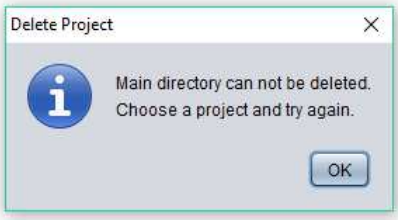
Poleg statusne vrstice, ki uporabnika konstantno obvešča o statusu sistema in akcijah, ki jih uporabnik izvaja, za takojšnjo povratno informacijo poskrbijo tudi pojavna okna. Ustvarili smo odpuščajoč sistem, ki tolerira oz. sprejme nepopolne ali nemogoče akcije brez negativnih posledic za uporabnika [3]. V primeru sistemske napake se pojavi okno, ki uporabnika opozori na napako in namigne, zakaj je do napake lahko prišlo. Sistemska napaka je prikazana na sliki 3.10.

V primeru nemogoče akcije se pojavi okno, ki uporabnika obvesti, da ta akcija ni izvedljiva in mu predlaga drugo izvedljivo opcijo. Okno z obvestilom prikazuje slika 3.11.

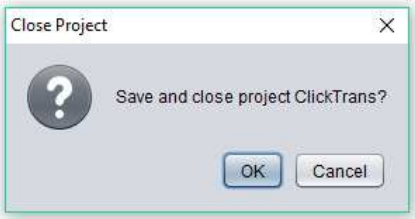
Tretja različica pa so okna, ki se pojavijo pri akcijah, ki vplivajo na vsebino datotek, npr. shranjevanje, zapiranje in brisanje projektov, kot kaže slika 3.12. Ta uporabnika pozovejo k potrditvi želenega dejanja, tako da ima uporabnik večji občutek kontrole nad dejanji [6].



Slika 3.10: Obvestilo o sistemski napaki.



Slika 3.11: Obvestilo o neizvedljivi akciji.



Slika 3.12: Obvestilo o akciji, ki vpliva na vsebino datotek.

3.1.3 Testiranje delovanja in uporabnosti programskega orodja

Orodje smo želeli preizkusiti na resničnem primeru, zato smo izkoristili priložnost ter zadovoljili kar dve strani hkrati. Transportno-logistično podjetje Kobal Transporti d.o.o. se ukvarja s specializiranim transportom živil po Sloveniji, ki zagotovlja sledljivost in temperaturni režim tovora [25]. Potrebovali so program, ki bi združeval funkcije treh programov, ki so jih v tistem trenutku uporabljali za svoje delo. Po daljšem razmisleku so se odločili za ClickTrans, program podjetja Infotrans d.o.o., ki transportno-logističnim podjetjem pomaga pri vodenju in izvajanju različnih aktivnosti [15]. Ker so pred nakupom in celovito prenovo želeli program najprej preizkusiti, smo se ponudili, da za njih ovrednotimo uporabnost aplikacije ClickTrans, prikazane na sliki 3.13.

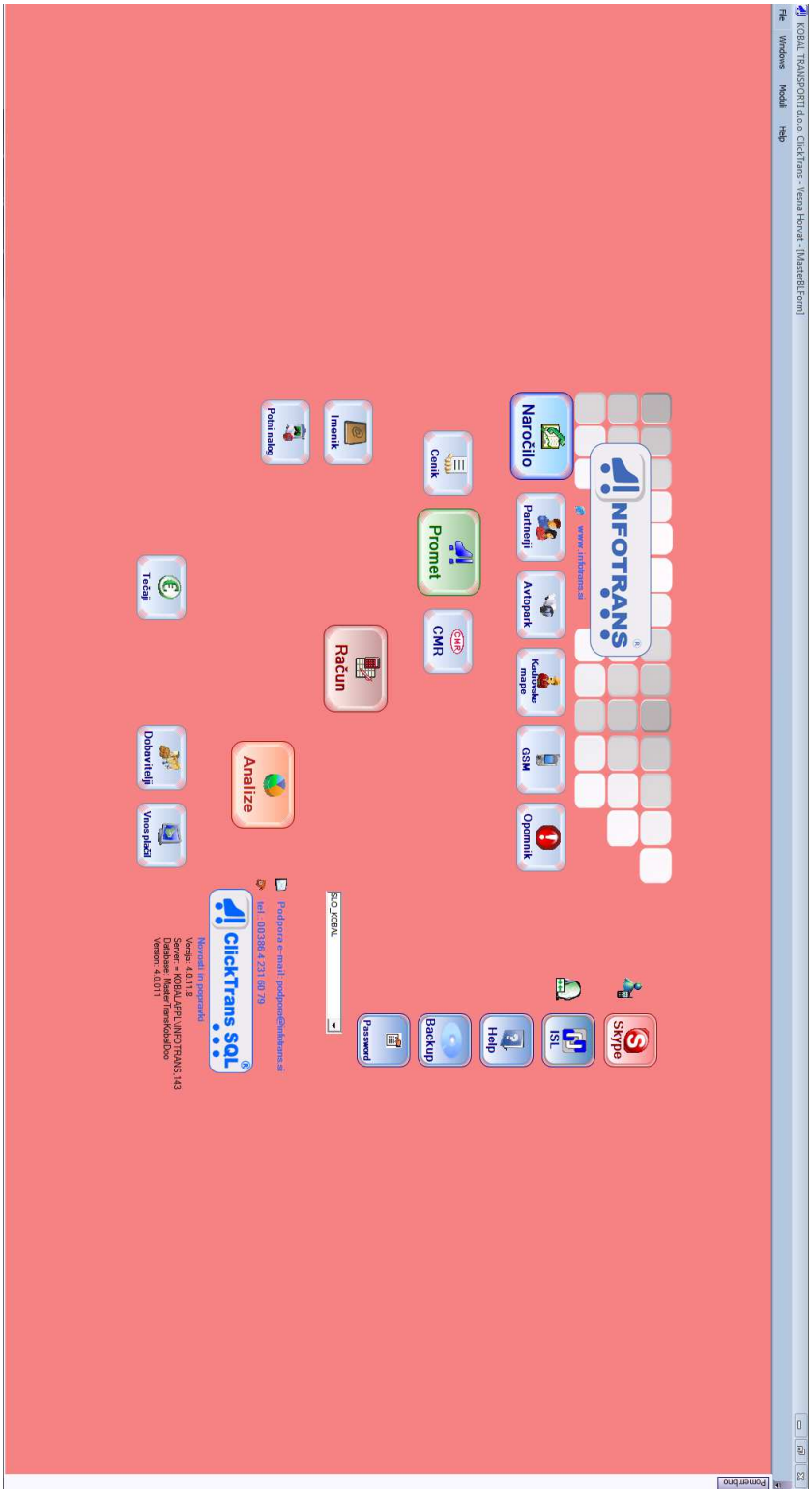
V naslednjih dveh poglavjih sledita podrobna primera hevrističnega vrednotenja in testiranja uporabnikov na podlagi aplikacije ClickTrans, ki smo ju izvedli s pomočjo UIUE.

3.2 Primer hevrističnega vrednotenja

Pri vrednotenju uporabniškega vmesnika in same aplikacije smo skušali upoštevati čim več principov, namigov in zlatih pravil oblikovanja, pa tudi želje in potrebe tipičnih uporabnikov.

Odločili smo se, da bo hevristično vrednotenje izvedel strokovnjak domene - izkušen uporabnik, z odličnim poznavanjem delovanja programa. To vlogo smo prevzeli sami in inšpekcijski pregled izvedli v svojem delovnem okolju, pri čemer smo poskrbeli, da med delom ni prihajalo do nepotrebnih motenj.

Zaradi preglednosti vrednotenja, smo za glavno vodilo ocenjevanja določili 10 Nielsenovih principov oblikovanja uporabniških vmesnikov. Ocenjevalca smo prosili, da naj svoje opazke skrbno oštevilči, zabeleži in priloži slike, kjer so primerne.

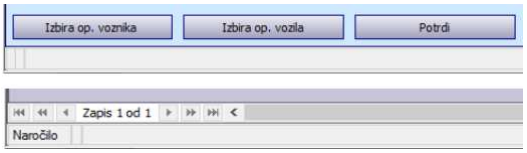


Slika 3.13: Osnovni meni aplikacije Clicktrans.

3.2.1 Vrednotenje po Nielsenovih principih

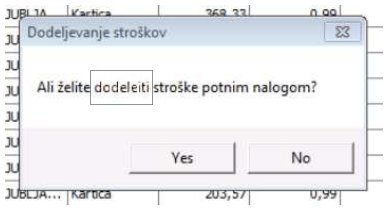
1. Vidljivost statusa sistema

- **Zelo nujno**: Odsotnost statusne vrstice.
  Opis: Statusna vrstica, ki uporabnika obvešča o vseh izvedenih akcijah ni prisotna v vseh modulih aplikacije (npr. manjka v modulu Opomnik), kot prikazuje slika 3.14.
  Priporočilo: Statusna vrstica naj bo del vsakega modula, podmodula in na splošno vsakega elementa, ki se lahko posamezno odpre v novem oknu ali zavihku.



Slika 3.14: Odsotnost statusne vrstice v modulu Opomnik.

- **Nujno**: Uporabnik mora biti ustrezno obveščen o statusu sistema.
  Opis: Pojavna okna uporabnika obveščajo o statusu sistema, vendar včasih ne zagotavljajo ustrezne povratne informacije. Prisotne so tudi slovnične napake, kot prikazuje slika 3.15.
  Priporočilo: Ustrezno usposobljena oseba naj pregleda in odpravi vse pravopisne napake. Pojavna okna naj vsebujejo kratke in ustrezne povratne informacije, ki bodo uporabniku razumljive.



Slika 3.15: Pojavno okno s slovnično napako.

- **Pomembno**: Tekoče spremembe, poudarjanje in vidljivost izbora.
  Opis: Po celotni aplikaciji klik na neko vrstico izbrano vrstico obarva belo-modro, kot prikazuje slika 3.16. Na ta način se zakrije dejanska barva opomnika (bela ali rdeča).
  Priporočilo: Ob kliku na vrstico naj se spremeni font besedila vrstice (npr. v krepko, ležeče ali podčrtano), ozadje besedila pa naj ostane brez polnila.

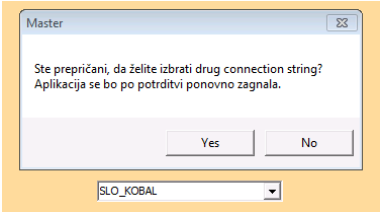
Slika 3.16: Belo-modro obarvana izbrana vrstica opomnika.

2. Prilagodi se realnemu svetu

- **Zelo nujno**: Slovenski jezik skozi celotno aplikacijo.
  Opis: Skozi celotno aplikacijo je uporabljen slovenski jezik, določeni gumbi pa so zapisani s kraticami, ki uporabniku niso blizu ali pa so zapisani v angleščini, kot prikazuje slika 3.17.
  Priporočilo: Jezik aplikacije naj bo slovenščina. Vsi zapisi naj bojo v slovenščini, lahko so uporabljene svetovno znane okrajšave ali prevzete besede (npr. Skype).

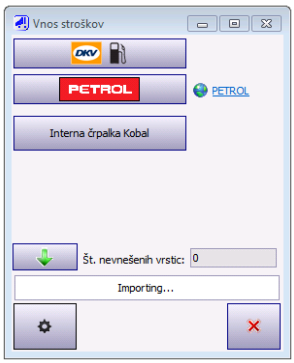
Slika 3.17: Nekaj gumbov z neprimernim zapisom.

- **Nujno**: Uporaba tujih izrazov v obvestilih.
  Opis: Pri menjavi podjetja na osnovni strani aplikacije, se v pojavnem oknu izpiše vprašanje, ki vsebuje tuj izraz, kot prikazuje slika 3.18. Če tujke uporabniki ne razumejo, potrjujejo nekaj na pamet, kar lahko privede do napake.
  Priporočilo: Skozi celotno aplikacijo naj se namesto tujk uporabljajo slovenske besede in splošni izrazi, ki so uporabnikom lažje razumljivi.



Slika 3.18: Neprimerna uporaba tujega izraza.

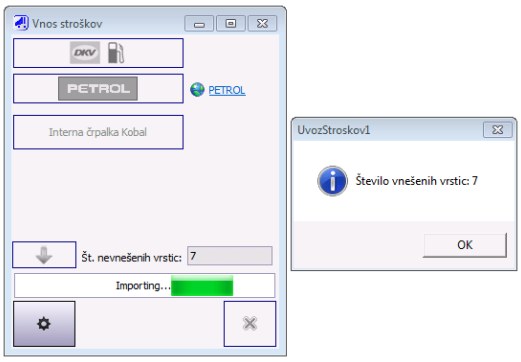
- **Manj pomembno**: Neustrezna uporaba metafor.
  Opis: Uporaba metafore puščice pri uvozu stroškov, kot prikazuje slika 3.19, je uporabniku nekoliko nerazumljiva.
  Priporočilo: Uporaba lažje razumljive metafore ali pa dodan pisni zapis, za obrazložitev pomena ikone (npr. Uvoz stroškov).



Slika 3.19: Neustrezna uporaba metafore na gumbu.

3. Uporabnikov nadzor in svoboda

- **Zelo nujno**: Prekinljivost akcij.
  Opis: Nekatere večje operacije nimajo gumba za preklic. Npr. ko se pri vnosu stroškov enkrat klikne na puščico, se akcije ne da več prekiniti, kot prikazuje slika 3.20.
  Priporočilo: Dodati je potrebno gumb Prekliči, ki bo omogočal prekinitev izvajanja daljših operacij.

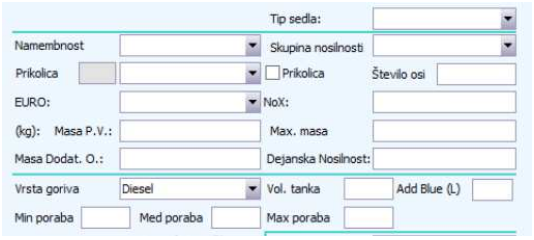


Slika 3.20: Nezmožnost prekinitve uvoza stroškov.

- **Nujno**: Izbris napačnega vnosa.
  Opis: Če se pri vpisu novega vozila v modulu Avtopark uporabnik zmoti pri vrsti vozila (npr. vozilo vnese med vlačilce namesto med kombinirana vozila), te napake sam ne more odpraviti. Potreben je klic na podjetje Infotrans d.o.o., kjer napačen vnos odstranijo direktno iz baze podatkov.
  Priporočilo: V vsaki kartici vozila naj bo možnost razveljavitve vnosa, s čimer preprečimo kreiranje vnosa še pred shranjevanjem. Če je vnos že shranjen, bi lahko dodali možnost premika vozila (izpolnjena polja se prenesejo), tako uporabnik ne rabi znova vnašati istih podatkov, ali pa omogočili dejanski izbris vozila, ki bi ga zaradi destruktivnega učinka spremljalo potrditveno okno.

4. Konsistentnost in standardi

- **Zelo nujno**: Konsistentnost razvrstitve gradnikov.
  Opis: V kartici vozila katerekoli vrste vozila znotraj modula Avtopark so gradniki slabo poravnani (nekateri gradniki so poravnani levo, drugi desno, prisotna je tudi neustrezna raba belih presledkov), kar daje zgled neorganiziranosti. Kartico vozila prikazuje slika 3.21.
  Priporočilo: Enotna poravnanost znotraj vsake kartice vozila bo zagotovila večjo preglednost nad podatki in poskrbela za lažjo uporabo aplikacije.

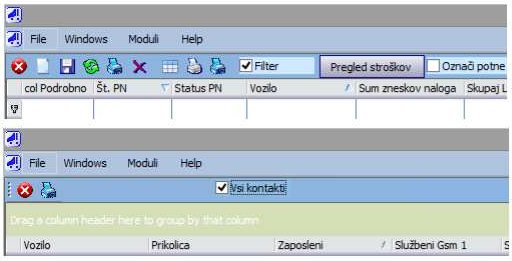


Slika 3.21: Nekonsistentna razvrstitev gradnikov.

- **Nujno**: Nepremišljena uporaba barv.
  Opis: V aplikaciji so barve uporabljene v prevelikem obsegu, zato izgubijo na pomenu. Pogosto se zgodi celo, da je na istem mestu uporabljenih več različnih odtenkov iste barve, ki se med seboj tepejo, kar vzbuja občutek nelagodja. Primer neustrezne uporabe barv prikazuje slika 3.22.
  Priporočilo: Potrebno je zmanjšati število barv in z njimi povezati tematsko sorodne sklope aplikacije. Nekaj več pozornosti je potrebno nameniti oznakam, ki naj imajo ozadje brez polnila. Prav tako naj bo rdeča barva uporabljena samo pri opozorilih in opomnikih. S tem se bo drastično izboljšala preglednost nad celotno aplikacijo in njenimi sklopi.

Slika 3.22: Nesmotrna uporaba barv.

- **Pomembno**: Neenoten izgled posameznih modulov.
  Opis: Nekateri moduli aplikacije nimajo statusne vrstice, ponekod manjka tudi indikator napredka. Prav tako imajo nekateri moduli ukazne gumbe v orodni vrstici, medtem ko jih imajo drugi nad statusno vrstico ali na levem robu modula. Slika 3.23 prikazuje primerjavo dveh modulov.
  Priporočilo: Module v aplikaciji je potrebno poenotiti, vsi moduli naj bodo zasnovani na enak način. Orodna vrstica je lahko nekoliko večja (večji gumbi z ikonami), da izpostavljene funkcije bolj pridejo do izraza.



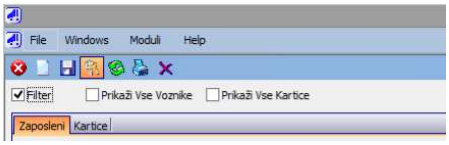
Slika 3.23: Primerjava dveh različnih modulov.

5. Izogibanje napakam

- **Nujno**: Postavitev gumbov v orodni vrstici.
  Opis: V orodni vrstici se nahajata dva različna gumba z oznako X,

kar povzroči nemalo zmede. Težavo predstavljata tudi gumba za nov dokument in shranjevanje, ki sta dve popolnoma raznoliki operaciji. Stojita namreč neposredno skupaj, zaradi česar lahko pride do pritiska na napačen gumb. Na sliki 3.24 je orodna vrstica modula Kadrovske mape.

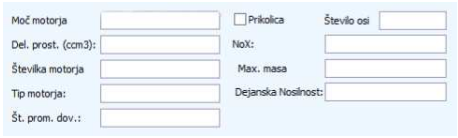
Priporočilo: Oznaka X naj predstavlja samo gumb za zapiranje ali izhod iz programa. Prav tako se ta po vsesplošnih standardih oblikovanja ne nahaja v orodni vrstici, temveč v desnem zgornjem kotu okna ali menijske vrstice. Drugi X je potrebno zamenjati z drugačno ikono, ki uporabnika ne bo zmedla. Med gumba za nov dokument in shranjevanje bi lahko postavili gumb za zaklep dokumenta.



Slika 3.24: Neprimerna postavitev gumbov v orodni vrstici.

- **Nujno**: Zaščita uporabnikovega dela.
  Opis: Čeprav obstaja možnost varnostnega kopiranja podatkov (celotne baze), aplikacija ne nudi sprotnega samodejnega shranjevanja. Prav tako ne podpira funkcij Razeljavi in Obnovi.
  Priporočilo: Dobro bi bilo dodati vsaj razveljavitev in obnavljanje zadnjih akcij, saj bi tako uporabnik dobil občutek, da njegova dejanja niso nepopravljiva.

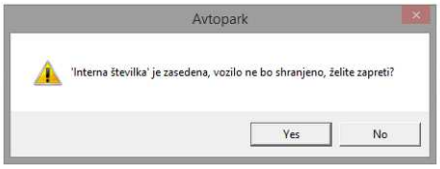
- **Pomembno**: Napake zaradi tipkanja.
  Opis: Na nekaterih mestih je po nepotrebnem uporabljeno polje za vnos besedila, ki je ranljivo za napake pri tipkanju. Slika 3.25 prikazuje polje za vnos v kartici vozila.
  Priporočilo: V primerih, kjer se odgovor ponavlja, bi bilo bolj smiselno uporabiti polje za izbiro ali kombiniran seznam.



Slika 3.25: Pretirana uporaba polj za vnos besedila.

6. Raje prepoznaj kot si zapomni

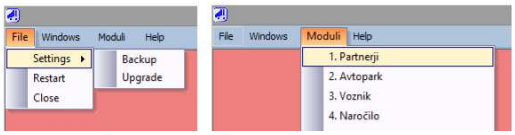
- **Manj pomembno**: Zaporedna številka novega vnosa.
  Opis: Kot prikazuje slika 3.26, si mora uporabnik pri vnosu novega vozila zapomniti zadnjo uporabljeno zaporedno številko in določiti naslednjo, ali pa uganiti neko še nezasedeno številko. To lahko povzroči, da si številke ne sledijo po vrsti, ampak skačejo (npr. en uporabnik dodeli številke od 1 do 10, drugi pa začne z 20, namesto z 11).
  Priporočilo: Aplikacija naj ob novem vnosu avtomatično dodeli naslednjo prosto zaporedno številko.



Slika 3.26: Slaba rešitev dodeljevanja zaporedne številke.

7. Fleksibilnost in učinkovitost

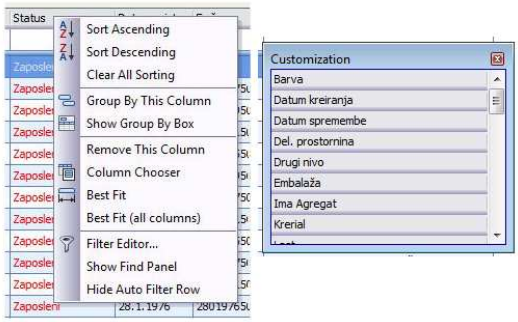
- **Pomembno**: Lahko naučljive bližnjice za pogoste operacije.
  Opis: V aplikaciji pospeševalcev, mnemonikov in lahko naučljivih bližnjic preko tipkovnice ni, kot je jasno razvidno iz slike 3.27. Napredni uporabniki z dobrim računalniškim predznanjem zato svoje delo opravljajo počasneje, kot bi ga lahko sicer.
  Priporočilo: Za lažjo in hitrejšo uporabo aplikacije je potrebno zagotoviti uporabo bližnjic.



Slika 3.27: Pospeševalcev, mnemonikov in bližnjic preko tipkovnice ni.

- **Manj pomembno**: Beleženje zgodovine zadnjih sprememb.
  Opis: V aplikaciji ni beleženja zgodovine zadnjih sprememb, ali pa ta uporabnikom ni vidna. V primeru človeške napake, krivec ni znan.
  Priporočilo: Beleženje zgodovine bi bilo smiselno dodati, vendar ne za vsako akcijo. Ob beleženju vseh dejanj, bi bil zapisnik zgodovine hitro lahko neobvladljivo dolg, pri čemer bi bilo nemogoče poiskati posamezne zapise. Najbolj učinkovito bi bilo beleženje glavnih akcij, kot so npr. novi vnosi, spreminjanje podatkov, shranjevanje in brisanje.

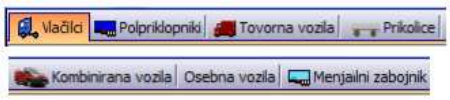
- **Manj pomembno**: Povrnitev na privzete nastavitve.
  Opis: Aplikacija sicer omogoča personaliziranje stolpcev v posameznih modulih, kot kaže slika 3.28, ne omogoča pa povrnitve na privzete nastavitve.
  Priporočilo: V glavnem meniju in v vsakem posameznem modulu naj obstaja možnost povrnitve na privzete nastavitve.



Slika 3.28: Povrnitev na privzete nastavitve ni možna.

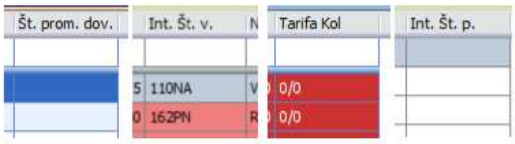
8. Estetika in minimalistično načrtovanje

- **Manj pomembno**: Prekompleksno grafično načrtovanje.
  Opis: Nekatere ikone so preveč grafično dodelane, kot kaže slika 3.29. Poleg tega je uporabljenih preveč barv.
  Priporočilo: Čeprav je dobro, da vrste vozil spremljajo ikone, ki razbremenijo uporabnikov spomin, so sličice nekoliko prekompleksne. Priporočljiva je uporaba le nekaj dobro izbranih barv in detajlov.



Slika 3.29: Kompleksna oblika in pretirana uporaba barv ikon.

- **Manj pomembno**: Uporaba kratkega in jedrnatega jezika.
  Opis: Skozi celotno aplikacijo je uporabljen minimalistično načrtovan jezik. Oznake so kratke, ponekod celo prekratke. Kot kaže slika 3.30, se v imenih stolpcev pojavljajo okrajšave, ki jih uporabnik ne razume.
  Priporočilo: Ohranitev kratkega in jedrnatega jezika v oznakah z uporabo celih besed. Kjer je smiselno, naj bodo uporabljene le znane okrajšave (npr. število in merske enote).



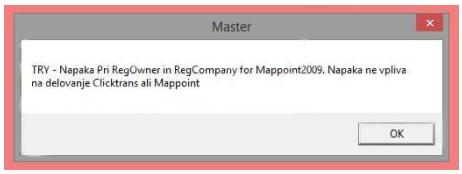
Slika 3.30: Neustrezne okrajšave besed v imenih stolpcev.

9. Javljanje napak, diagnoza, reševanje

- **Zelo nujno**: Obvestilo o napaki je v tehničnem žargonu.
  Opis: Nekatere napake so zapisane v tehničnem žargonu, kot prikazuje

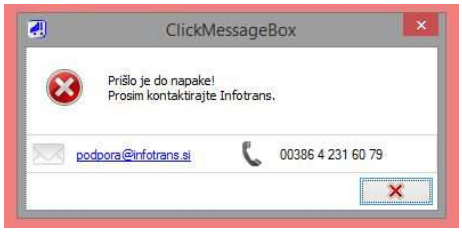
slika 3.31. Tehničnega žargona uporabniki v večini primerov ne razumejo.

Priporočilo: Skozi celotno aplikacijo naj se namesto tehničnega žargona uporabljajo splošne, uporabnikom razumljive besede. Specifični izrazi v domeni naj bodo uporabljeni samo, kadar je to primerno.



Slika 3.31: Neprimerna uporaba tehničnega žargona.

- **Nujno**: Pomanjkljivo sporočilo o napaki.
  Opis: Pojavljajo se napake, kakor ta na sliki 3.32, ki ne nudijo nikakršne povratne informacije (zakaj je do napake prišlo in kako jo odpraviti), temveč uporabnika zgolj pozovejo, da naj kontaktira podjetje Infotrans d.o.o.
  Priporočilo: V primeru napake naj se v pojavnem oknu izpiše do kakšne napake je prišlo ter kako naj uporabnik napako odpravi ali vsaj prepreči v prihodnje. V primeru ponavljanja težave ali pojavitve resnejše napake pa naj se uporabnik obrne na podjetje Infotrans d.o.o.



Slika 3.32: Razlog za napako ni jasen.

- **Pomembno**: Bodi prijazen in ne grajaj.
  Opis: Pojavna okna, ki so namenjena uporabniku aplikacije so na tre-

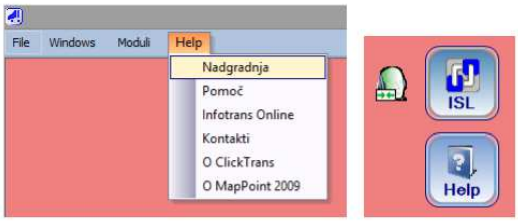
nutke preveč agresivna in zaradi tega lahko delujejo neprijazno do uporabnika. Opozorilo na sliki 3.33 ima ikono z znakom stop, ki deluje, kot da bo šlo nekaj narobe, če uporabnik ne bo pregledal opomnika ta trenutek.
Priporočilo: Pojavna okna z opozorili naj bodo prijaznejša uporabniku, da ne dobi občutka, da dela nekaj kaznivega, prepovedanega.

Slika 3.33: Preveč neposredna in napadalna opozorila.

10. Pomoč in dokumentacija

- **Pomembno**: Za pomoč uporabnikom ni poskrbljeno.
  Opis: Pri dejanski uporabi aplikacije za pomoč ni poskrbljeno. Pripravljeni so gumbi za pomoč in spletni pogovor, vendar ne delujejo. Na sliki 3.34 lahko vidimo pripravljene gumbe. Strokovni uporabnik se ob neki nejasnosti verjetno še nekako znajde, uporabnik začetnik pa mora pomoč poiskati neposredno pri podjetju Infotrans d.o.o. Kontaktirati jih mora preko telefona ali e-pošte, kar pa lahko za uporabnika postane zelo neprijetno v fazi, ko se še uči in se mu zatakne večkrat dnevno.
  Priporočilo: Smiselno bi bilo dodati namige za uporabo, ki bi se jih dalo izklopiti. Spustni meni Pomoč bi bil lahko sestavni del menijske vrstice vsakega modula, podmodula in elementa, prikazoval pa bi najpogostejša vprašanja ali napake z ustrezno razlago ali koraki reševanja.

- **Manj pomembno**: Dokumentacija je preobsežna.
  Opis: Sama aplikacija je zelo dobro dokumentirana, saj jo spremlja obširen in nazoren priročnik, ki zajema vse module in funkcije apli-



Slika 3.34: Za pomoč uporabnikom zaenkrat še ni poskrbljeno.

kacije. Poleg tega je skripta podkrepljena z veliko slikami in primeri. Problem nastane pri učenju uporabe aplikacije, saj uporabniku vzame absolutno preveč časa. Na sliki 3.35 je prva stran skripte, ki jo je prejel vsak uporabnik.

Priporočilo: Priročnik z navodili za uporabo aplikacije je potrebno poenostaviti, strniti navodila in tako skrajšati čas, ki je potreben za učenje uporabe aplikacije.

Slika 3.35: Priročnik z navodili za uporabo aplikacije.

3.2.2 Analiza in interpretacija rezultatov hevrističnega vrednotenja

Vse ocenjevalčeve ugotovitve smo zapisali na način, kot ga predlaga Miller [30]. Zapis vsebuje stopnjo nujnosti, kratek naziv opazke, opis težave in priporočilo.

Stopnje nujnosti smo razdelili v štiri kategorije: zelo nujno, nujno, pomembno in manj pomembno. S tabelo, ki jo prikazuje slika 3.36, smo prikazali število in delež težav glede na stopnjo nujnosti.

Stopnja nujnosti	Število opaženih težav	Delež
Zelo nujno	5	20 %
Nujno	7	28 %
Pomembno	6	24 %
Manj pomembno	7	28 %

Slika 3.36: Tabela števila in deležev težav glede na stopnjo nujnosti.

Čeprav je hevristično vrednotenje izvedel izkušen uporabnik, z odličnim poznavanjem domene, Nielsen [20] namiguje, da en ocenjevalec spregleda večino problemov uporabnosti, ki jih ima vmesnik. Posamezen ocenjevalec naj bi po njegovih raziskavah odkril le 35 % težav. Z njegovo ugotovitvijo se popolnoma strinjamo, saj gre za obsežen program, ki mu en sam inšpektor ne more biti kos.

Upoštevajoč rezultate smo ugotovili, da program ClickTrans potrebuje še nekaj manjših in večjih popravkov in še kakšno dodatno hevristično vrednotenje, sicer pa gre za dober produkt. Program je zelo kompleksen, saj ga sestavlja veliko število modulov in podmodulov, z desetinami malenkostnih podrobnosti, ki jih je potrebno prilagoditi potrebam podjetja, ki program uporablja. Ker pa mnenje enega strokovnjaka ne zadošča za popolno oceno produkta, nas je zanimalo še mnenje tistih, ki se s programom soočajo vsakodnevno. Zato smo izvedli še testiranje uporabnikov.

3.3 Primer testiranja uporabnikov

Za testiranje uporabnosti s pomočjo testov uporabnosti je bilo najbolj primerno testranje uslužbencev podjetja, ki imajo vsakodnevno stik s tem programom. Odločili smo se, da bomo testiranje izvedli izven službenega časa, da delo ni ovirano, v pisarni posameznega uslužbenca, z njihovim računalnikom in sistemom nadzorovanja preko oddaljenega dostopa.

3.3.1 Testiranje po korakih

Načrt

Pri načrtovanju testa smo določili posamezne enote: cilje, ki smo jih želeli doseči, sredstva, ki smo jih imeli na voljo, vrsto interakcije s subjekti ter čas in okolje v katerem bo potekal test.

Cilj izvedbe testa uporabnosti je bil odkriti, kakšno mnenje imajo uporabniki o aplikaciji oz. uporabniškem vmesniku aplikacije Clicktrans. Ali se jim zdi lahka/težka za uporabo? Ali so barve, ikone, besedila primerna? Ali vse funkcije najdejo takoj ali za to porabijo preveč časa? Ali se jim polja za vnos in drugi elementi zdijo primerni? Želeli smo dobiti odgovore na vsa ta vprašanja, predvsem pa nas je zanimalo, kako bi lahko aplikacijo še izboljšali.

Sredstvo, ki smo ga imeli na voljo, je bila dokončno izdelana aplikacija, zato je testiranje na papirnem prototipu odpadlo. Odločili smo se za testiranje na računalniku, direktno z aplikacijo oz. testno različico aplikacije, ki je imela popolnoma enake funkcije, vendar testno bazo podatkov.

Testi uporabnosti se med seboj lahko razlikujejo po količini interakcije med preizkuševalcem in testirancem. Interakcija med njimi je lahko visoka ali nizka [31]. Izbrali smo visoko interaktivno tehniko, pri kateri je interakcija med preizkuševalcem in testirancem dovoljena v veliki meri.

Za okolje testiranja se nam je zdelo najbolj primerno službeno okolje, bolj specifično pisarna posameznega testiranca. Domnevali smo, da bo mirno, domače okolje pripomoglo k temu, da bodo testiranci bolj sproščeni, pod manjšim pritiskom in s tem tudi bolj odprti. Odločili smo se namreč, da bo

reševanje testa potekalo na računalniku, ki bo z drugim računalnikom povezan preko oddaljenega dostopa, tako da bo preizkuševalec lahko v realnem času spremljal delo testiranca brez tega, da bi mu moral stati za hrbtom.

Določili smo še, da se bo test reševal izven delovnega časa zaradi naslednjih razlogov: da sodelujoči v službenem času lahko opravi svoje službene dolžnosti, da lahko reševanje poteka nemoteno in da se lahko testiranec in preizkuševalec med testiranjem posvetita izključno samo testu.

Sodelujoči

Izbira testirancev ni bila enostavna. Tipično je dovolj, če test opravi osem do deset oseb, saj ti v večini primerov zajamejo rezultate, ki so značilni za 95% vseh, ki aplikacijo uporabljajo [31]. Upoštevajoč etiko in raznolikost smo za test izbrali pet primarnih in tri sekundarne uporabnike [10].

Materiali in prostor

Pripravili smo vprašalnike, ki so jih testiranci dobili pred reševanjem. Na vsakem vprašalniku so na začetku vprašanja o njihovih karakteristikah - spol, starost, izobrazba, delovno mesto, delovna doba, funkcije, izkušnje z računalniki in s programom, jasno pa je izpostavljeno tudi, da je reševanje anonimno. Nato sledi deset nalog, ki se na vsakem vprašalniku razlikujejo v podrobnostih (vnos ali spreminjanje različnih podatkov).

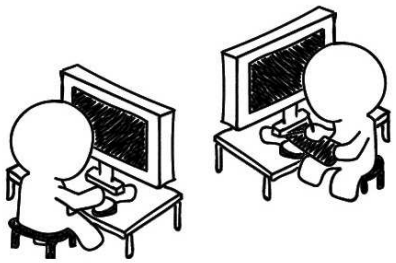
Po vzoru različnih obrazcev za zbiranje podatkov[23, 31] smo sestavili tudi obrazec, ki ga je preizkuševalec izpolnjeval med testiranjem. Nekatere elemente, kot so številka testiranca, datum, ura, naloga, čas izvedbe in sprotne opazke, je lahko vpisal sproti, medtem ko je napake preveril naknadno po koncu reševanja. Temu primerno smo oblikovali tudi ocenjevalno lestvico.

Ker bi bila pisarna vsakega testiranca prehrupna in bi lahko prihajalo do večjega števila motenj, smo okolje testiranja prestavili v prazno pisarno. Tam sta bila na voljo le dva računalnika, nekaj pisal in praznih listov papirja.

Izvedba testov

Poskusnega testa nismo izvajali, saj se nam zaradi dobre organiziranosti in ustreznih predpriprav, to ni zdelo potrebno. Preskočili smo kar na izvedbo pravega testa.

Testiranje je potekalo v mirnem in tihem okolju, v pisarni vsakega testiranca. Testiranec in preizkuševalec sta sedela pravokotno nekoliko odmaknjena drug od drugega, kot kaže slika 3.37. Vsak je imel svoj namizni računalnik, ki sta bila med seboj povezana preko programa Teamviewer [35], ki omogoča oddaljen dostop do računalnika. Na ta način smo zagotovili, da lahko preizkuševalec spremlja subjektovo delo, pri tem pa ne vdira v njegov zasebni prostor, kar je za testiranca manj stresno.



Slika 3.37: Prikaz položaja testiranca in preizkuševalca med testiranjem.

Med testiranjem smo sodelujoče prosili, da naj svoje izbire, dvome in druge opombe glasno komentirajo ali vprašajo, če česa ne razumejo. Svoje ugotovitve glede posamezne naloge so lahko napisali tudi na vprašalnik, kjer nas je zanimalo še, kako težka se jim je zdela naloga, kaj bi na aplikaciji spremenili, da bi bilo reševanje lažje, kaj jim je pri reševanju pomagalo itd. Primer vprašalnika prikazuje slika 3.38. Prav tako je preizkuševalec po vsaki izvršeni nalogi testiranca povprašal o tem, kakšno je njihovo mnenje o določenih elementih, funkcijah in ostalih podrobnostih aplikacije. Vse njihove komentarje, preedloge in opazke je zapisoval sproti v poseben obrazec, ki ga prikazuje slika 3.39.

PROGRAM CLICKTRANS

Pred vami je kratek vprašalnik z nalogami. Besedilo naloge dobro preberite, jo izvedite in odgovorite na vprašanja. Če imate med reševanjem kakršnokoli vprašanje, se obrnite na izvajalca ankete. Vprašalnik je anonimen, odgovori pa se bodo uporabili izključno za namen diplomske naloge.

Preden začnemo z reševanjem nalog, prosim, izpolnite naslednje podatke:
Spol: _____ Starost: _____
Najvišja dokončana izobrazba: _____
Delovno mesto: _____
Okviren čas delovanja na tem delovnem mestu: _____
Opis funkcije, nalog, zadolžitev: _____
_____
_____

Kako dobro poznate program Clictrans, obkrožite:

1 – danes sem se prvič seznanil s tem programom
2 – program poznam, ampak se ga še učim uporabljati
3 – program uporabljam samostojno, občasno za nasvet vprašam sodelavca ali podporo
4 – program obvladam, ne potrebujem več pomoči
5 – sem avtor programa/član IT skupine, ki je izdelala program in ga tudi posodablja

Naloge

1. Na glavnem menuju izberite in odprite modul avtopark. Med osebna vozila dodajte novo vozilo. Vnesite naslednje podatke:

Registrska številka: LJ AB-123
Interna številka: AB123
Osebno vozilo, kategorija A, Mercedes, C/250T/CDI
Letnik: 2012
Datum prve registracije: 6.3.2012
Tip karoserije: karavan
Številka šasije: WDD2042031F375935
Barva: S0M
Vrsta goriva: Diesel
Podjetje: Tovornjak d.o.o.
Registracija: 5.5.2016, opomnik 30 dni pred iztekom
Redni servis: 4.5.2016, 35700 km, perioda 20000 km

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: 1 2 3 4 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj? _____
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?
_____
_____

Slika 3.38: Primer vprašalnika: prva stran.

Z vprašanji in nalogami smo zajeli večino situacij, s katerimi se povprečni uporabnik sooči pri uporabi aplikacije. Poleg tega smo dodali tudi nekaj robnih primerov, pri katerih se izrazi poznavanje svojega področja (npr. ve, da osebno vozilo nima tahografa, loči vlačilca od tovornega vozila ipd.) in rokovanja z aplikacijo (npr. ve, da se glavni moduli nahajajo na glavnem meniju, zna uporabiti bližnjice ipd.).

Testiranec:			
Datum in ura:			
Naloga	Čas izvedbe	Napake	Opazke

Slika 3.39: Primer spremljevalnega obrazca preizkuševalca.

Analiza rezultatov

Kakor svetujejo Stone in ostali [10] smo po koncu testiranja zbrali vse vprašalnike in preizkuševalčeve zapiske ter jih ustrezno označili. Določili smo kriterije za točkovanje, vprašalnike pregledali in ocenili. Oblikovali smo tabelo in graf, ki predstavljata čas reševanja posameznih uporabnikov. Vse njihove komentarje, predloge in pripombe smo zbrali in povzeli v poročilu.

3.3.2 Analiza in interpretacija rezultatov testiranja uporabnikov

Kriterije za točkovanje smo določili za vsako nalogo posebej glede na dejavnike, ki so se nam zdeli pomembni. Upoštevali smo pravilnost rešitev, uporabo bližnjic ter samostojnost reševanja oz. potrebo po dodatni razlagi ali pomoči glede reševanja naloge. Naloge so različno težke in zato tudi različno točkovane. Vsaka naloga prispeva določeno število k skupni vrednosti točk, ki je 100. Primer točkovanja prve naloge prikazuje slika 3.40.

TOČKOVANJE

1. Naloga

Modul avtopark, zavihek Osebna vozila	3 točke
Registrska številka	3 točke
Interna številka	1 točka
Osebno vozilo, kategorija, znamka, tip	1 točka
Letnik	1 točka
Datum prve registracije	1 točka
Tip karoserije	1 točka
Številka šasije	3 točke
Barva	1 točka
Vrsta goriva	1 točka
Podjetje	1 točka
Registracija	2 točki
Redni servis	2 točki
Samostojno reševanje brez pomoči	1 točka
Uporaba bližnjic	1 točka
Skupaj	23 točk

Slika 3.40: Primer točkovanja: prva naloga.

Za testiranje smo izbrali osem reprezentativnih uporabnikov. Pet takih, ki naj bi bili pogosteje v stiku s programom (primarni uporabniki) in tri take, ki delajo v istem podjetju, a naj bi program uporabljali redkeje (sekundarni uporabniki). Nekateri so se na testiranju prvič seznanili s programom, drugi pa so ga različno dolgo že uporabljali. Prav tako imajo nekateri zgolj osnovno računalniško znanje, drugi so bolj vešči, tretji pa imajo odlično predznanje in obvladajo različno programsko opremo. S tabelo, ki jo prikazuje slika 3.41, smo prikazali koliko točk je dosegel posamezen testiranec in v kolikšnem času je dokončal vse naloge. V čas reševanja smo šteli le reševanje na računalniku,

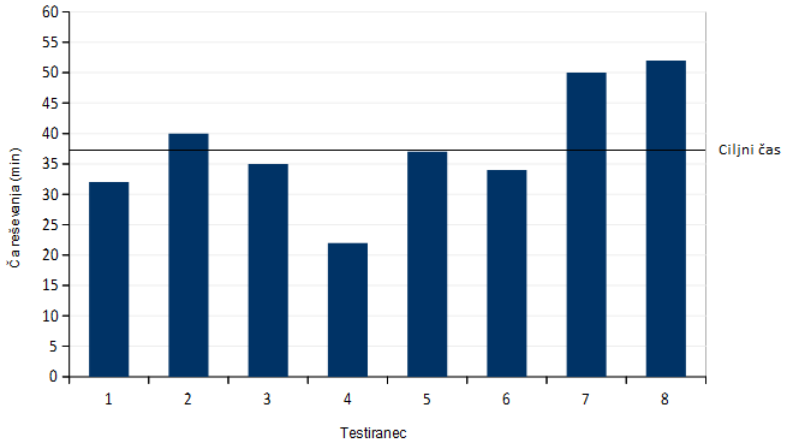
brez začetnega branja navodil, odgovarjanja na vprašanja ter končnega pogovora s preizkuševalcem.

Testiranec		Seznanjenost s programom	Računalniško predznanje	Doseženih točk	Čas reševanja
Primarni uporabniki	1 (LS)	Samostojni uporabnik	Osnovno predznanje	93	32 min
	2 (DN)	Prvič seznanjen	Osnovno predznanje	91	40 min
	3 (ES)	Prvič seznanjen	Dobro predznanje	88	35 min
	4 (JU)	Napredni uporabnik	Odlično predznanje	98	22 min
	5 (AP)	Uporabnik začetnik	Dobro predznanje	95	37 min
Sekundarni uporabniki	6 (GH)	Prvič seznanjen	Odlično predznanje	97	34 min
	7 (BK)	Prvič seznanjen	Osnovno predznanje	90	50 min
	8 (NT)	Prvič seznanjen	Dobro predznanje	92	52 min

Slika 3.41: Tabelaričen prikaz rezultatov testiranja.

Za lažjo predstavo uspešnosti smo oblikovali še graf na sliki 3.42, ki prikazuje, koliko časa je porabil posamezen testiranec za dokončanje vseh nalog. Čas reševanja se giblje med 22 in 52 minutami. Ciljni čas smo dobili tako, da smo poiskali sredino med najpočasnejšim in najhitrejšim časom. V povprečju so testiranci dosegli visoko število točk. Najslabši jih je dosegel 88, najboljši pa 98.

Primerjali smo tudi število napak, ki so jih testiranci storili pri posamezni nalogi. Na podlagi njihovih in preizkuševalčevih komentarjev in opomb smo skušali posplošiti zaradi česa je do teh napak prišlo. Ali je šlo zgolj za slabo poznavanje programa ali pa bi se v programu dalo kaj popraviti tako, da bi prihajalo do manj napak. Pri tem moramo poudariti, da so bili vsi testi izvedeni na podvojeni različici programa, ki smo jo imeli na voljo junija 2016, tako da je verjetno marsikatera napaka ali opazka do danes že odpravljena. Naša različica je bila postavljena na ločenem strežniku in je imela interno bazo s kopijo pravih podatkov, ni pa se posodabljala v realnem času tako kot program, ki so ga uporabljali v podjetju Kobal Transporti d.o.o. Na ta način so testiranci lahko vnašali podatke brez negativnih posledic za podjetje,



Slika 3.42: Grafičen prikaz časa reševanja testirancev z označenim ciljnim časom.

rezultati testa pa so bili pristni.

Ugotovili smo, da je do napak povečini prihajalo zaradi sistemskih pomanjkljivosti ali pa pomanjkanja znanja in izkušenj z uporabo programa. V splošnem so uporabniki naloge rešili zelo uspešno, z minimalnim številom kritičnih napak. Razloge, zaradi katerih je najpogosteje prišlo do napake in predloge za odpravo le-teh smo ponazorili s tabelo na sliki 3.43.

Testiranci so med reševanjem opozorili še na nekaj dodatnih pomanjkljivosti programa, ki smo jih skupaj s predlogi za izboljšavo predstavili s tabelo, ki jo prikazuje slika 3.44.

Ocena produkta glede na dimenzije uporabnosti

Z upoštevanjem Andrewsovih [23] dimenzij uporabnosti, smo program Clicktrans ocenili glede na šest merljivih lastnosti:

Uporabnost: Uporabniki so v večji meri izkoristili vse osnovne funkcionalnosti sistema, z izjemo uporabe filtrov in bližnjic. V povprečju so testiranci pri reševanju standardnih nalog dosegli 93 od 100 točk, kar znaša 93 %. Program je torej zelo uporaben.

Vrsta in opis napake		Primer	Predlog za odpravo napake
Sistemska napaka	Izbira datuma preko koledarja ni možna, potreben je ročni vnos datuma, zapis št. let veljavnosti in nato ponovno ročni vnos datuma veljavnosti.	Podatki opomnika v kartici vozila, npr. vnos opomnika za registracijo ali tahograf.	Izbira datuma preko koledarja in samodejni izračun datuma veljavnosti na podlagi vnešenega št. let veljavnosti.
	Nekatera imena gumbov in oznak so napisana v angleščini, čeprav je primarni jezik programa slovenščina.	Šifrant gesel je poimenovan s Password, varnostna kopija podatkov se imenuje Backup, pomoč pa je označena s Help.	Čez celoten sistem naj bo uporabljen le slovenski jezik. Vse oznake, gumbi in pojavna okna naj bodo prevedena v slovenščino.
	Na nekaterih mestih so uporabljene nejasne kratice, ki niso izražene ne v žargonu ne po svetovnih standardih.	Lastnosti vozila v kartici vozila, npr. Masa P. V., št. pal. mest ali istov. rampa.	Kratice naj bodo smiselne. Okrajšane naj bodo le besede, ki jih razumejo tako laiki, kot tudi strokovni uporabniki.
	Vnos nekaterih podatkov je nesmiselno zasnovan. Marsikje je potrebno podatke vpisati v polje z besedilom, kar podaljša čas vnosa.	Št. osi v kartici vozila ali pa vnos kraja v kartici partnerja.	Namesto polja za vnos besedila bi bil lahko marsikje spustni seznam z logičnimi izbirami. Za vnos vrednosti, ki je še ni na seznamu, bi lahko dodali gumbek z oznako +.
Sistemska in človeška napaka	Registrsko št. prikolice vozila, ki jo vozi, si je potrebno zapomniti na pamet, da lahko pri spremembah na vozilu popravimo tudi lastnosti prikolice.	Registrska št. prikolice je zapisana v kartici vozila, ki jo vozi. Prikolice se nahajajo v ločenem zavihku.	Znotraj kartice vozila bi lahko obstajal gumb z direktno povezavo na kartico prikolice.
	Klik na vrstico (označevanje) z modro barvo zakrije barvo vrstice, ki lahko ponazarja pretek nekega podatka.	Posamezna rdeče ali belo obarvana vrstica modula Opomnik.	Klik na vrstico bi lahko ohranil barvo vrstice, tekst pa bi se izpisal krepko.
Človeška napaka	Neuporaba bližnjic in filtrov, ki olajšajo in pohitrijo iskanje podatkov.	Filtri so locirani v prvi vrstici posameznega modula ali zavihka.	Lahko bi bili bolj izpostavljeni, odebeljeni, mogoče malo ločeni od tabele, da bi jih uporabnik prej opazil.

Slika 3.43: Tabela najpogostejših napak in predlogi za odpravo.

Opis pomanjkljivosti	Primer	Predlog za izboljšavo
Neenakomerna poravnava gradnikov v kartici vozila.	Nekateri gradniki so poravnani levo, drugi pa desno. Zapisi so nesistematično razporejeni.	Smiselna uporaba belih presledkov in mrežna razporeditev gradnikov s poravnavo na isto stran.
Podrobnosti elementa se odprejo s klikom na gumb, ki je označen s tremi pikami.	Npr. kartica vozila v modulu avtopark ali kartica osebe v modulu kadrovske mape.	Podrobnosti bi se lahko odprle tudi z dvojnim klikom na vrstico.
Metafore niso povsod razumljive. Gumbi z ikonami bi ponekod morali imeti še ustrezen zapis za lažje razumevanje.	Pri vnosu stroškov je zelena puščica, za katero ni jasno, kaj naredi (prenese vrstice, odpre podrobnosti, ...).	Vsak gumb z ikono naj ima smiselno oznako ali opis.

Slika 3.44: Tabela opaženih pomanjkljivosti in predlogi za izboljšavo.

Naučljivost: Uporabnika, ki sta bila prvič seznanjena s programom in imata osnovno računalniško predznanje sta za reševanje v poprečju porabila 45 minut. Uporabnika, z dobrim računalniškim predznanjem sta v povprečju potrebovala 43 minut in pol. Uporabnik, ki ima odlično računalniško predznanje pa je naloge rešil v 43 minutah. Prav tako so testiranci 50 % nalog ocenili kot zelo lahke, 15 % nalog kot lahke, 20 % kot naloge srednje težavnosti in 15 % kot težke naloge. Nobeni nalogi ni bila dodeljena zelo težka težavnost. Iz tega sledi, da je program srednje do lahko naučljiv.

Učinkovitost: Samostojni uporabnik z osnovnim računalniškim predznanjem je za reševanje porabil 32 minut, medtem ko je napredni uporabnik z odličnim računalniškim predznanjem naloge rešil v samo 22 minutah. Testiranca sta kar 90 % nalog ocenila kot zelo lahke in preostalih 10 % kot lahke naloge. Glede na dobljene rezultate bi sistem označili kot zelo učinkovit.

Pomnenje: Samostojni uporabnik je v povprečju rešil naloge pol minute hitreje kot uporabnik začetnik. Naloge, ki jih je opravljal redkeje, so mu vzele več časa kot naloge, ki jih je opravljal vsakodnevno. Iz tega lahko sklepamo, da si je uporabnik zapomnil navodila od zadnje uporabe in potreboval le minimalno količino časa za osvežitev znanja. Program bi torej lahko definirali kot sorazmerno zapomljiv.

Varnost: Do resnejših napak med reševanjem ni prišlo, smo pa zabeležili nekaj manjših napak. Več kot polovica jih je bila povzročena zaradi sistemskih pomanjkljivosti (npr. testiranec je vnesel napačen datum, ker je bil potreben dvojni ročni vnos ali pa ni našel gumba, ker je bil ta poimenovan v angleščini oz. ni bil jasno označen), ostale napake so bile zgolj posledica hitrega reševanja nalog. Ker bi program v tej fazi potreboval še nekaj izboljšav, bi označili sistem za zmerno varen.

Zadovoljstvo: Testiranci so s programom na splošno zadovoljni. Program so označili kot praktičen in srednje prijeten do prijeten za uporabo, vendar estetsko nekoliko neprivlačen. Še bolj bi bili zadovoljni, če bi razvijalci programa dodelali estetsko podobo programa, odpravili pomanjkljivosti in razmislili o njihovih predlogih za izboljšavo.

Poglavje 4

Sklepne ugotovitve

4.1 Diskusija

Razvito programsko orodje je preprosto za uporabo in oblikovano po principih in navodilih, ki jih predstavlja. Da bi testirali njegovo funkcionalnost, smo ga preizkusili na praktičnem primeru. Ovrednotili smo uporabnost aplikacije Clicktrans, ki jo je transportno logistično podjetje Kobal Transporti d.o.o. uporabljalo za vsesplošno kontrolo nad potekom dela, vodenjem evidence o zaposlenih in vozilih, nad porabo goriva in ostalih zadev. Z uporabo razvitega orodja smo pripravili hevristično poročilo s praktičnimi primeri pomanjkljivosti, ki smo jih našli v aplikaciji. Prav tako smo izvedli testiranje uporabnikov z osmimi uporabniki, ki delajo v podjetju. Tudi te ugotovitve smo predstavili v poročilu, pri čemer smo si pomagali z našim orodjem.

Orodje se je izkazalo za učinkovito, izpostavljenih pa je bilo manjše število pomanjkljivosti, ki jim bomo v prihodnosti še posvetili pozornost. Podrobnejše povratne informacije glede uporabnosti programskega orodja bomo pridobili v naslednjih letih, ko bo orodje praktično preizkusilo večje število ljudi.

4.2 Nadaljnje delo

Kakor humorno prikazuje slika 4.1, bo naše nadaljnje delo obsegalo odpravo hroščev in tiskarskih napak, dodati nameravamo še funkcije Razveljavi, Obnovi ter samodejno shranjevanje, dodelati pa je potrebno tudi stilsko oblikovanje besedila v urejevalnikih besedil. Sčasoma bo potrebno posodobiti tudi zunanjo grafično podobo orodja, da bo v koraku s sodobnimi grafičnimi orodji tega časa. Razmišljali smo tudi o prevodu orodja v slovenščino, če bi prišlo do povpraševanja, potencialno še v kateri drugi jezik, npr. nemščino, francoščino, španščino, ruščino, kitajščino ali druge. Glede na želje in potrebe uporabnikov bomo orodje sproti posodabljali in nadgrajevali, sledili razvoju principov in navodil ter po potrebi dodali tista, ki se bodo izkazala za uporabna.

Slika 4.1: Odprava hroščev in napak ter posodabljanje orodja.

Literatura

[1] A. Collins McLaughlin, A. D. Fisk in W. A. Rogers. (2012). Using Direct and Indirect Input Devices - Attention Demands and Age-Related Differences, *PMC* [online].
Dosegljivo: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3342758/. [Dostopano 16. 9. 2018].

[2] A. Cooper, R. Reimann in D. Cronin, *About Face 3: The Essentials of Interaction Design*. Indianapolis, Indiana, Wiley Publishing, Inc., 2007.

[3] A. Marcus, N. Smilonich in L. Thompson, *The Cross-GUI Handbook for Multiplatform User Interface Design*. Boston, Massachusetts, Addison-Wesley, 1995.

[4] B. Schneiderman in C. Plaisant, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 4. izd. Boston, Massachusetts, Addison-Wesley, 2005.

[5] B. Schneiderman in C. Plaisant, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, 5. izd. Boston, Massachusetts, Addison-Wesley, 2010.

[6] B. Tognazzini. (2014). First Principles of Interaction Design - Revised and Expanded, *AskTOG, Interaction Solutions for the Real World* [online].
Dosegljivo: http://asktog.com/atc/principles-of-interaction-design. [Dostopano 13. 5. 2018].

[7] D. A. Norman, *The Design of Everyday Things*. New York, New York, Basic Books, 2002.

[8] D. Benyon, *Designing Interactive Systems: A comprehensive guide to HCI and interactive design*, 2. izd. Harlow, England, Pearson Education Limited, 2010.

[9] D. Norman. (2013). *The Design of Everyday Things - Revised and expanded edition* [online]. New York, New York: Basic Books.
Dosegljivo: http://www.nixdell.com/classes/HCI-and-Design-Spring-2017/The-Design-of-Everyday-Things-Revised-and-Expanded-Edition.pdf.
[Dostopano 15. 9. 2018].

[10] D. Stone et al., *User Interface Design and Evaluation*. Los Altos, California, Morgan Kaufmann Publishers, 2005.

[11] F. Jager, "Prosojnice za predavanja pri predmetu Uporabniški vmesniki". Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, šolsko leto 2014/2015.

[12] F. Solina, "Prosojnice za diplomski seminar pri predmetu Projektni praktikum". Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, šolsko leto 2017/2018.

[13] H. Loranger. (2014). Accordions Are Not Always the Answer for Complex Content on Desktops, *Nielsen Norman Group* [online].
Dosegljivo: https://www.nngroup.com/articles/accordions-complex-content/.
[Dostopano 14. 9. 2018].

[14] *Icon, Index and Symbol* [online].
Dosegljivo: https://www.cs.indiana.edu/ port/teach/103/sign.symbol.short.html.
[Dostopano 23. 9. 2018].

[15] *Infotrans d.o.o. – Informacijske rešitve za logistiko, transport, distribucijo, prevozništvo* [online].
Dosegljivo: http://www.infotrans.si/infotrans/.
[Dostopano 22. 5. 2018].

[16] *Interaction Design Foundation: Gestalt Principles* [online].
Dosegljivo: https://www.interaction-design.org/literature/topics/gestalt-principles.
[Dostopano 6. 9. 2018].

[17] *ISO 9241:2018, Ergonomics of human-system interaction — Part 11, Usability: Definitions and concepts* [online].
Dosegljivo: https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-2:v1:en.
[Dostopano 2. 9. 2018].

[18] J. Girard. (2018). Using Contrasting Foreground and Background Colors in Web Design, *Lifewire* [online].
Dosegljivo: https://www.lifewire.com/contrasting-foreground-background-colors-4061363.
[Dostopano 19. 9. 2018].

[19] J. Nielsen. (1995). 10 Usability Heuristics for User Interface Design, *Nielsen Norman Group* [online].
Dosegljivo: https://www.nngroup.com/articles/ten-usability-heuristics.
[Dostopano 13. 5. 2018].

[20] J. Nielsen, *Usability Engineering*. San Francisco, California, Academic Press, 1993.

[21] J. Nielsen. (2000). Why You Only Need to Test with 5 Users, *Nielsen Norman Group* [online].
Dosegljivo: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.
[Dostopano 19. 7. 2018].

[22] J. Raskin, *The Humane Interface: New Directions for Designing Interactive Systems*. Boston, Massachusetts, Addison-Wesley, 2000.

[23] K. Andrews. Human-Computer Interaction. (2018). Graz University of Technology [online].
Dosegljivo: http://courses.iicm.tugraz.at/hci/hci.
[Dostopano 13. 5. 2018].

[24] K. Matz. (2012). Donald Normans Design Principles for Usability, *Architecting Usability* [online].
Dosegljivo: http://architectingusability.com/2012/06/28/donald-normans-design-principles-for-usability/.
[Dostopano 14. 9. 2018].

[25] *Kobal transporti d.o.o. - Specializirano podjetje za transport znotraj Slovenije* [online].
Dosegljivo: https://www.kobaltransporti.si/o-nas.
[Dostopano 21. 5. 2018].

[26] *Microsoft: Error Message Guidelines* [online].
Dosegljivo: https://docs.microsoft.com/en-us/windows/desktop/Debug/error-message-guidelines.
[Dostopano 17. 9. 2018].

[27] N. Babich. (2018). 7 Basic Rules for Button Design, *UX Planet* [online].
Dosegljivo: https://uxplanet.org/7-basic-rules-for-button-design-63dcdf5676b4.
[Dostopano 16. 9. 2018].

[28] *NetBeans - An integrated development environment (IDE) for Java* [online].
Dosegljivo: https://en.wikipedia.org/wiki/NetBeans.
[Dostopano 14. 5. 2018].

[29] *Oracle: TextSamplerDemo (open source code)* [online].
Dosegljivo: https://docs.oracle.com/javase/tutorial/uiswing/examples/components/TextSamplerDemoProject/src/components/TextSamplerDemo.java.
[Dostopano: 30. 6. 2018].

[30] R. Miller. (2018). User Interface Design and Implementation - Readings. Massachusetts Institute of Technology [online].
Dosegljivo: http://web.mit.edu/6.813/www/sp18/.
[Dostopano 6. 5. 2018].

[31] S. Weinschenk, P. Jamar in S. C. Yeo, *GUI Design Essentials*. New York, New York, John Wiley & Sons, Inc., 1997.

[32] *T. Mandel: Golden Rules of User Interface Design* [online].
Dosegljivo: http://theomandel.com/resources/golden-rules-of-user-interface-design/.
[Dostopano 6. 5. 2018].

[33] T. Mandel, *The Elements of User Interface Design*. New York, New York, John Wiley & Sons, Inc., 1997.

[34] T. Mandel. (1997) The Golden Rules of User Interface Design. V *The Elements of User Interface Design* [online]. New York, New York, John Wiley & Sons, Inc., str. 47-79.
Dosegljivo: http://theomandel.com/wp-content/uploads/2012/07/Mandel-GoldenRules.pdf.
[Dostopano 6. 5. 2018].

[35] *TeamViewer - Remote Desktop Tool* [online].
Dosegljivo: https://www.teamviewer.com/en/products/teamviewer/.
[Dostopano 1. 7. 2018].

[36] *The Java Tutorials: How to Make Dialogs* [online].
Dosegljivo: https://docs.oracle.com/javase/tutorial/uiswing/components/

dialog.html.
[Dostopano 17. 9. 2018].

[37] *The Java Tutorials: How to Use Layered Panes* [online].
Dosegljivo: https://docs.oracle.com/javase/tutorial/uiswing/components/layeredpane.html.
[Dostopano 14. 9. 2018].

[38] *The Java Tutorials: How to Use Modality in Dialogs* [online].
Dosegljivo: https://docs.oracle.com/javase/tutorial/uiswing/misc/modality.html.
[Dostopano 17. 9. 2018].

[39] *The Java Tutorials: How to Use Tabbed Panes* [online].
Dosegljivo: https://docs.oracle.com/javase/tutorial/uiswing/components/tabbedpane.html.
[Dostopano 14. 9. 2018].

[40] W. O. Galitz, *It's Time to Clean Your Windows: Designing GUIs That Work*. New York, New York, John Wiley & Sons, Inc., 1994.

[41] W. O. Galitz. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Tehniques*, 3. izd. Indianapolis, Indiana, Wiley Publishing, Inc., 2007.

[42] *W3schools: HTML* [online].
Dosegljivo: http://web.mit.edu/6.813/www/sp18/.
[Dostopano 16. 6. 2018].

[43] *Wikipedia: Swing (Java)* [online].
Dosegljivo: https://en.wikipedia.org/wiki/Swing_(Java).
[Dostopano 16. 6. 2018].

Priloge

Priloga 1: Kriteriji točkovanja (Testiranje uporabnikov)
Priloga 2: Primer izpolnjenega prašalnika (Testiranje uporabnikov)
Priloga 3: Primer preizkuševalčevih zapiskov (Testiranje uporabnikov)

TOČKOVANJE

1. Naloga

Modul avtopark, zavihek Osebna vozila	3 točke
Registrska številka	3 točke
Interna številka	1 točka
Osebno vozilo, kategorija, znamka, tip	1 točka
Letnik	1 točka
Datum prve registracije	1 točka
Tip karoserije	1 točka
Številka šasije	3 točke
Barva	1 točka
Vrsta goriva	1 točka
Podjetje	1 točka
Registracija	2 točki
Redni servis	2 točki
Samostojno reševanje brez pomoči	1 točka
Uporaba bližnjic	1 točka
Skupaj	23 točk

2. Naloga

Pravo vozilo	3 točke
Datum registracije	1 točka
Opomnik registracija	1 točka
Datum tahografa	1 točka
Opomnik tahografa	1 točka
Samostojno reševanje brez pomoči	1 točka
Skupaj	8 točk

3. Naloga

Pravo vozilo	3 točke
Moč motorja	1 točka
Prostornina motorja	1 točka
Tip motorja	1 točka
Emisijski razred	2 točki
Število osi	1 točka
Masa vozila	2 točki
Maksimalna masa vozila	1 točka
Višina, širina, dolžina	2 točki
Skupaj	14 točk

4. Naloga	
Pravo vozilo	3 točke
Prava prikolica	3 točke
Pravi potni nalog	3 točke
Sprememba statusa potnega naloga	1 točka
Samostojno reševanje brez pomoči	2 točki
Skupaj	12 točk
5. Naloga	
Pravo vozilo	3 točke
Datum točenja goriva	1 točka
Stanje števca	2 točki
Samostojno reševanje brez pomoči	1 točka
Skupaj	7 točk
6. Naloga	
Pravo vozilo	3 točke
Datum točenja goriva	1 točka
Vozilo	1 točka
Plačilo goriva z gotovino	2 točki
Količina	2 točki
Stanje števca	2 točki
Samostojno reševanje brez pomoči	1 točka
Skupaj	12 točk
7. Naloga	
Pravo vozilo	1 točka
Samostojno reševanje brez pomoči	1 točka
Skupaj	2 točki
8. Naloga	
Samostojno najde šifrant gesel	1 točka
Samostojno nastavi geslo	1 točka
Pravo geslo	1 točka
Male črke	3 točke
Skupaj	6 točk

9. Naloga	
Kratko in dolgo ime	1 točka
Zaupanja vreden poslovni partner	2 točki
Naslov	2 točki
Davčna številka	2 točki
Samostojno reševanje brez pomoči	1 točka
Skupaj	8 točk
10. Naloga	
Modul opomnik, zavihek Vozilo Podatki	2 točki
Pravo vozilo	1 točka
Uporaba bližnjic	1 točka
Kaj je preteklo in za koliko dni	3 točke
Samostojno reševanje brez pomoči	1 točka
Skupaj	8 točk

Priloga 1: Kriteriji točkovanja (Testiranje uporabnikov)

6

PROGRAM CLICKTRANS

Pred vami je kratek vprašalnik z nalogami. Besedilo naloge dobro preberite, jo izvedite in odgovorite na vprašanja. Če imate med reševanjem kakršnokoli vprašanje, se obrnite na izvajalca ankete. Vprašalnik je anonimen, odgovori pa se bodo uporabili izključno za namen diplomske naloge.

Preden začnemo z reševanjem nalog, prosim, izpolnite naslednje podatke:
Spol: Ž Starost: 23
Najvišja dokončana izobrazba: dipl. kršč. inf.
Delovno mesto: tajnica / poslovna sekretarka
Okviren čas delovanja na tem delovnem mestu: 2leti
Opis funkcije, nalog, zadolžitev: knjiženje računov, potni nalogi, kondenca stroškov vozil

Kako dobro poznate program Clictrans, obkrožite:
1 – danes sem se prvič seznanil s tem programom
2 – program poznam, ampak se ga še učim uporabljati
3 – program uporabljam samostojno, občasno za nasvet vprašam sodelavca ali podporo
4 – program obvladam, ne potrebujem več pomoči
5 – sem avtor programa/član IT skupine, ki je izdelala program in ga tudi posodablja

Naloge

1. Na glavnem menuju izberite in odprite modul avtopark. Med osebna vozila dodajte novo vozilo. Vnesite naslednje podatke:
   Registrska številka: LJ AB-123
   Interna številka: AB123
   Osebno vozilo, kategorija A, Mercedes, C/250T/CDI
   Letnik: 2012
   Datum prve registracije: 6.3.2012
   Tip karoserije: karavan
   Številka šasije: WDD2042031F375935
   Barva: S0M
   Vrsta goriva: Diesel
   Podjetje: Tovornjak d.o.o.
   Registracija: 5.5.2016, opomnik 30 dni pred iztekom
   Redni servis: 4.5.2016, 35700 km, perioda 20000 km

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: 1 2 3 4 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj?
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?
Ne, podatki opomnika, interval naslednjega servisa (mi bil prikazan)

2. Še vedno se nahajamo v modulu avtopark. Tovornemu vozilu z registrsko številko LJ AH-150 popravite veljavnost registracije na datum 16.5.2017 z opomnikom 30 dni pred iztekom. Istemu vozilu popravite tudi veljavnost tahografa na datum 12.5.2018 z opomnikom 30 dni pred iztekom.

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: 1 **(2)** 3 4 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj? ______
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?
Ne, dodajanje novega polja (+ zelo majhen)

3. Tovornemu vozilu LJ HV-300 dodajte naslednje manjkajoče podatke:

Moč motorja: 300 kW
Prostornina motorja: 11946 ccm^3
Tip motorja: OM 501 LA.V/8
EURO 5 emisijski razred
Število osi: 2
Masa vozila: 7441 kg
Maksimalna masa vozila: 18000 kg
Višina: 3,45 m, širina: 2,5 m, dolžina: 5,81 m

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: 1 **(2)** 3 4 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj? ______
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?

Opomba: Povsod bi lahko bile enote, poravnanost elementov

4. Tovorno vozilo z registrsko številko LJ AM-566 smo odjavili iz prometa. Nastavite, da je vozilo neaktivno. Prav tako na neaktivno nastavite tudi prikolico, ki jo to vozilo vleče. Zatem odprite modul potni nalogi in potnemu nalogu, ki je dodeljen vozilu LJ AM-566, spremenite status iz 1 (Odprt PN) na 9 (Storno PN).

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: 1 2 **(3)** 4 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj? ______
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?
tovornem vozilu bi lahko bila povezava na prikolico. PN

5. Še vedno se nahajamo v modulu potni nalogi. Voznik tovornega vozila, z registrsko številko LJ IR-207, je 23.3.2016 točil na Petrolovi bencinski črpalki in pri tem zapisal napačno stanje števca. Popravite zapis tako, da bo stanje števca vozila na ta dan 128914 km.

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: 1 2 3 **(4)** 5

Če ste obkrožili številko 4 ali 5, pojasnite zakaj? Težko najti
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?

6. Voznik tovornega vozila, z registrsko številko LJ KC-111, občasno toči gorivo tudi na drugih bencinskih črpalkah in pri tem plača z gotovino. Ročno vnesite zapis, ki vsebuje naslednje podatke:
   Datum: 20.3.2016
   Vozilo: LJ KC-111
   Plačilo goriva Q MAX Diesel z gotovino
   Količina: 76,35 l
   Stanje števca: 296830 km

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: (1) 2 3 4 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj?
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?
Da.

7. V modulu potni nalog se premaknite v Pregled stroškov in odčitajte povprečno mesečno porabo za vozilo z registrsko številko LJ 34-5X za mesec februar 2016. Poraba:

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: 1 2 3 (4) 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj?
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?
Pov. mesečnem pregledu bi bolje pisalo

8. Zaprite modul potni nalogi in na glavnem menuju izberite modul šifrant gesel. Geslo za odklep modula je »1«. Geslo za cenik popravite na »cenik«.

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: 1 2 3 (4) 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj?
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?
Pisalo bi bolje s spremi gesel

9. Zaprite modul šifrant gesel in na glavnem menuju izberite modul Partnerji. Dodajte novega poslovnega partnerja z naslednjimi podatki:
   Kratko in dolgo ime: Hitri in drzni d.o.o.
   Zaupanja vreden poslovni partner (zelena luč)

Naslov: Ob zeleni jami 5, 1000 Ljubljana, Slovenija
Davčna številka: SI - 12582367

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: 1 2 (3) 4 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj? ______
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?
Spodnjih polj se ni videlo, naslova še ne
da vnesti, naziv 2 ali lahko bil "dolg naziv"

10. Zaprite modul Partnerji in na glavnem menuju izberite modul Opomnik. Premaknite se na zavihek Vozilo Podatki in preverite, če je vozilu z registrsko številko LJ 34-09X potekla registracija, tahograf ali gasilni aparat. Na črto zapišite kaj je preteklo in za koliko dni (če ni preteklo nič, napišite »nič«):
nič

Kako zahtevna se vam je zdela naloga (1 – lahka, 5 – težka), obkrožite: (1) 2 3 4 5
Če ste obkrožili številko 4 ali 5, pojasnite zakaj? ______
Ali ste vse gumbe in polja našli takoj? Če ne, napišite, kateri so vam povzročali težave in zakaj?
/

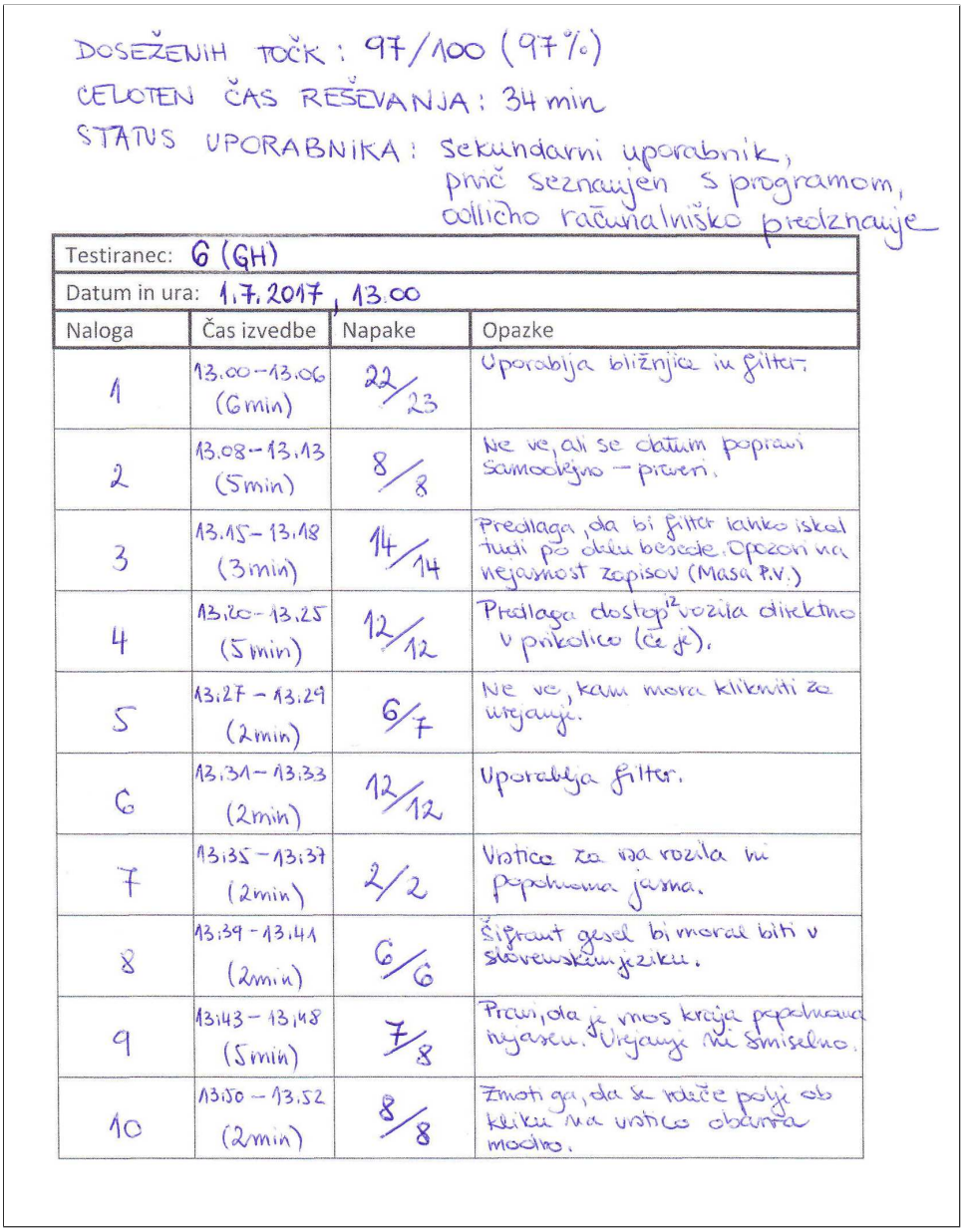
Na kratko opišite, katere lastnosti oz. funkcije so vam na programu Clicktrans še posebej všeč oz. se vam zdijo zelo uporabne?
+ Rdeče obarvani pretečeni roki, filtri

Kaj pa vam ni všeč, ali se vam zdi odveč, ali pa bi spremenili, popravili, izboljšali?
− Poravnanost, enote (ne pišejo povsod),
"ikona zapri – na desno

Ali imate še kakšno dodatno opombo, kritiko, predlog za izboljšavo ali mnenje o programu Clicktrans na splošno?
Domača stran je razmetana, nekonsistentna (barve, velikost, položaj, jezik

Najlepša hvala za sodelovanje ☺

Priloga 2: Primer izpolnjenega prašalnika (Testiranje uporabnikov)



DOSEŽENIH TOČK: 97/100 (97%)
CELOTEN ČAS REŠEVANJA: 34 min
STATUS UPORABNIKA: Sekundarni uporabnik, prvič seznanjen s programom, odlično računalniško predznanje

Testiranec: G (GH)
Datum in ura: 1.7.2017, 13.00

Naloga	Čas izvedbe	Napake	Opazke
1	13.00–13.06 (6min)	22/23	Uporablja bližnjice in filter.
2	13.08–13.13 (5min)	8/8	Ne ve, ali se datum popravi samodejno – preveri.
3	13.15–13.18 (3min)	14/14	Predlaga, da bi filter lahko iskal tudi po delu besede. Opozori na nejasnost zapisov (Masa P.V.)
4	13.20–13.25 (5min)	12/12	Predlaga dostop[2] vozila direktno v prikolico (če je).
5	13.27–13.29 (2min)	6/7	Ne ve, kam mora klikniti za urejanje.
6	13.31–13.33 (2min)	12/12	Uporablja filter.
7	13.35–13.37 (2min)	2/2	Vrstice za vsa vozila in popotnina jasna.
8	13.39–13.41 (2min)	6/6	Šifrant gesel bi moral biti v slovenskem jeziku.
9	13.43–13.48 (5min)	7/8	Pravi, da je vnos kraja popotnice nejasen. Urejanje ni smiselno.
10	13.50–13.52 (2min)	8/8	Zmoti ga, da se rdeče polje ob kliku na vrstico obarva modro.

Priloga 3: Primer preizkuševalčevih zapiskov (Testiranje uporabnikov)