KOMUNIKACIJA ČLOVEK–RAČUNALNIK

Saša Divjak



Kazalo

Predgovor 2

1. Kaj je HCI 3

2. Človek kot uporabnik 4

3. Ljudje s posebnimi potrebami in dostopnost 7

4. Modeli uporabnika in miselni modeli 9

5. Interakcija in povratna informacija 16

6. Zakoni UX (uporabniške izkušnje) 17

7. Informacijska arhitektura in navigacija 22

8. Oblikovanje uporabniškega vmesnika 24

9. Napake, razveljavitev in obnovitev 26

10. Uporabniške naloge in modeli interakcije 27

11. Prototipiranje pri uporabniško usmerjenem načrtovanju 30

12. Evalvacija uporabniškega vmesnika 36

13. Mobilni, spletni in multimodalni HCI 47

14. Umetna inteligenca in HCI 52

15. Proaktivni in agentski sistemi 59

16. XR, VR in prostorski uporabniški vmesniki 66

17. Prihodnost HCI in sinteza 70

Dodatek A: Orodja za načrtovanje in prototipiranje uporabniških vmesnikov 75

Dodatek B: Kontrolni seznam za načrtovanje in vrednotenje uporabniškega vmesnika 81

Dodatek C: Slovarček izrazov HCI 85

Dodatek D: Literatura in priporočeni viri 89

Predgovor

Ta dokument je mišljen kot učbenik. Besedilo je namenjeno kot opora pri predavanju, izhodišče za razpravo in podlaga za vaje.

Pedagoško izhodišče
Študent naj najprej opazi problem na konkretnem vmesniku. Šele nato uvedemo strokovni pojem, ga razložimo in uporabimo pri predlogu izboljšave.

V gradivu bomo kot kratico za Komunikacijo človek-računalnik uporabljali HCI (Human-Computer Interaction), saj s to kratico lažje najdemo ustrezne vsebine na spletu.

Začetna poglavja postavljajo osnovni okvir: HCI obravnava celotno interakcijo, uporabnik ima omejitve in različne potrebe, dostopnost pa mora biti del načrtovanja od začetka. Te temelje povežemo v nadaljevanju z miselnimi modeli, povratno informacijo, načeli oblikovanja, napakami, evalvacijo in sodobnimi oblikami interakcije. V posebnem poglavju so nanizani zakoni uporabniške izkušnje (UX, user experience) druga uporabna načela

Sledi sklop poglavij, ki tvorijo smiselno celoto: informacijska arhitektura določi, kako organiziramo informacije; oblikovanje uporabniškega vmesnika določi, kako jih predstavimo; obravnava napak omogoči varno interakcijo; analiza nalog preveri, ali vmesnik podpira tisto, kar želi uporabnik dejansko doseči; evalvacija pa pokaže, ali smo pri tem uspeli. Ne načrtujemo zaslonov. Načrtujemo interakcijo človeka s sistemom.

V zadnjih poglavjih sledijo sodobnejše teme: mobilnost in multimodalnost, AI, agenti, XR in nazadnje sinteza.

Skoraj vsako poglavje dopolnjuje predlog vaj, kar pomaga predvsem predavatelju.

Koristni so tudi dodatki. Zlasti kontrolni seznam B je praktičen: študent lahko po istem okviru preverja uporabnike in kontekst, navigacijo, miselni model, povratno informacijo, napake, vizualno zasnovo, dostopnost, prototipiranje in vrednotenje.

Saša Divjak (sasadiv@gmail.com)



1. Kaj je HCI

HCI (Human–Computer Interaction) oziroma komunikacija človek–računalnik je področje, ki preučuje, kako ljudje uporabljajo interaktivne računalniške sisteme, kako takšne sisteme načrtujemo in kako preverjamo njihovo kakovost. V ospredju ni samo zaslon ali program, temveč celota: uporabnik, njegova naloga, naprava, okolje, omejitve in posledice uporabe.

1.1 UI, UX in uporabljivost

Uporabniški vmesnik (UI – user interface) je del sistema, prek katerega uporabnik zaznava možnosti, vnaša podatke, sproža dejanja in prejema rezultate.

Uporabniška izkušnja (UX – user experience) je širša: vključuje pričakovanja pred uporabo, občutek razumljivosti, zaupanje, frustracijo, zadovoljstvo in izkušnjo po končani nalogi.

Uporabljivost (usability) je bolj osredotočena na to, ali lahko določeni uporabniki v določenem kontekstu dosežejo svoje cilje učinkovito, uspešno in zadovoljivo. UI, UX in uporabljivost se prekrivajo, vendar niso sinonimi.

Pojem	Osrednje vprašanje	Primer
UI	Kaj uporabnik vidi in s čim lahko deluje?	Gumb, meni, obrazec, glasovni ukaz.
UX	Kakšna je celotna uporabnikova izkušnja?	Ali se uporabnik počuti varno, razumljeno in v nadzoru?
Uporabljivost	Ali uporabnik uspešno opravi nalogo?	Ali zna rezervirati termin brez pomoči in napak?

1.2 Razvoj oblik interakcije

Oblike interakcije so se razvijale od ukazne vrstice do grafičnih vmesnikov, spleta, mobilnih naprav, glasovnih in multimodalnih sistemov, prostorskih vmesnikov ter sistemov z umetno inteligenco.Pri tem ne gre za preprosto zamenjavo starega z novim: različne oblike so primerne za različne naloge. K temu razvoju se bomo vrnili v zadnjem poglavju, ko bomo razmišljali o prihodnosti HCI.



Slika 1.1: Poenostavljen prikaz razvoja oblik interakcije.

Ključno vprašanje
Ali nova tehnologija res izboljša uporabnikovo nalogo ali samo naredi demonstracijo bolj zanimivo?

1.3 HCI kot načrtovalski in raziskovalni proces

HCI združuje analizo uporabnikov in nalog, oblikovanje rešitev, izdelavo prototipov ter evalvacijo. Postopek je iterativen: po testiranju se vrnemo k problemu in izboljšamo rešitev. Pri tem je pomembno, da ne začnemo pri izbrani tehnologiji, temveč pri uporabnikovi potrebi.

1. Razumi uporabnika, nalogo in kontekst.
2. Opredeli problem in merila uspeha.

3. Predlagaj več možnih rešitev.
4. Izdelaj prototip, ki je dovolj konkreten za preizkus.
5. Preizkusi rešitev z uporabniki ali s strokovnim pregledom.
6. Na podlagi ugotovitev izboljšaj rešitev.

Vaja:

Vzemite šest različnih uporabniških vmesnikov: na primer spletni obrazec, bankomat, mobilno aplikacijo, avtomobilski zaslon, predvajalnik glasbe in stran z napako. Brez predhodne razlage pojmov jih razvrstite od najboljšega do najslabšega ter navedite razloge.

- *Kateri vmesnik najhitreje razumete?*
- *Kje veste, kaj morate storiti?*
- *Kje niste prepričani, ali je bilo dejanje izvedeno?*
- *Kateri vmesnik bi bil problematičen za starejšega ali slabovidnega uporabnika?*
- *Katera opažanja lahko pozneje povežemo s pojmi HCI?*

2. Človek kot uporabnik

Načrtovalec pogosto pozna sistem predobro. Ve, kje je funkcija, kaj pomeni ikona in kakšen je pravilen postopek. Uporabnik tega znanja nima. Zato moramo vmesnik presojati z vidika človekovega zaznavanja, pozornosti, spomina, učenja in odločanja.

2.1 Zaznavanje in pozornost

Uporabnik običajno ne bere zaslona linearno. Najprej ga preleti, išče vizualno izstopajoče elemente in poskuša ugotoviti, katero dejanje je naslednje. Če so vsi elementi enako poudarjeni, se poveča čas iskanja in možnost napake.

- Pomembne elemente poudarimo s položajem, velikostjo, kontrastom in razmikom.
- Ne uporabljajmo preveč enakovrednih poudarkov.
- Besedilo naj bo kratko, vendar ne tako kratko, da postane dvoumno.
- Pri opozorilih je pomembnejša razumljivost kot dramatična barva.

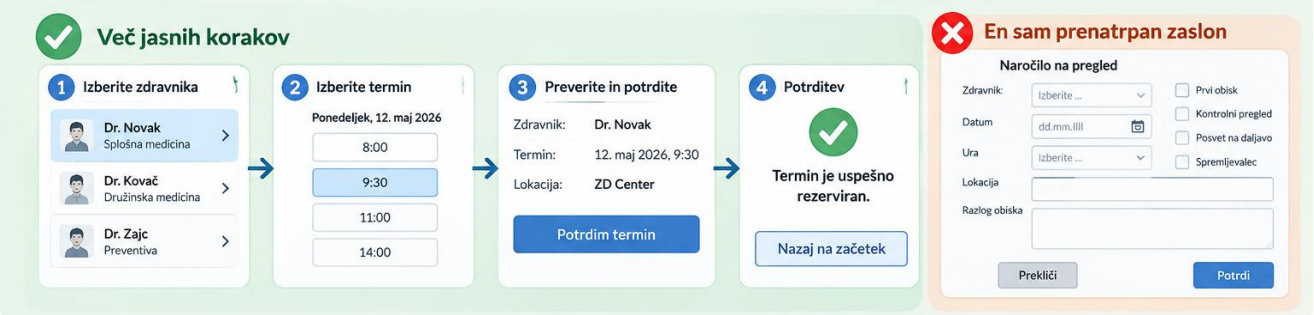
2.2 Spomin in kognitivna obremenitev

Delovni spomin je omejen. Vmesnik naj zato podpira prepoznavanje namesto priklica. Uporabnik naj vidi možnosti, prejšnje vnose, trenutno stanje in naslednji korak.



Slika 2.1: Prepoznavanje in priklic

Pri zahtevnih nalogah je bolje uporabiti več jasnih korakov kot en sam prenatrpan zaslon.



Slika 2.2: Več jasnih korakov ali prenatrpan zaslon

Primer
Če obrazec zahteva, da si uporabnik zapomni kodo, prikazano na prejšnjem zaslonu, sistem po nepotrebnem obremenjuje spomin. Boljša rešitev je prikaz kode ali možnost kopiranja.

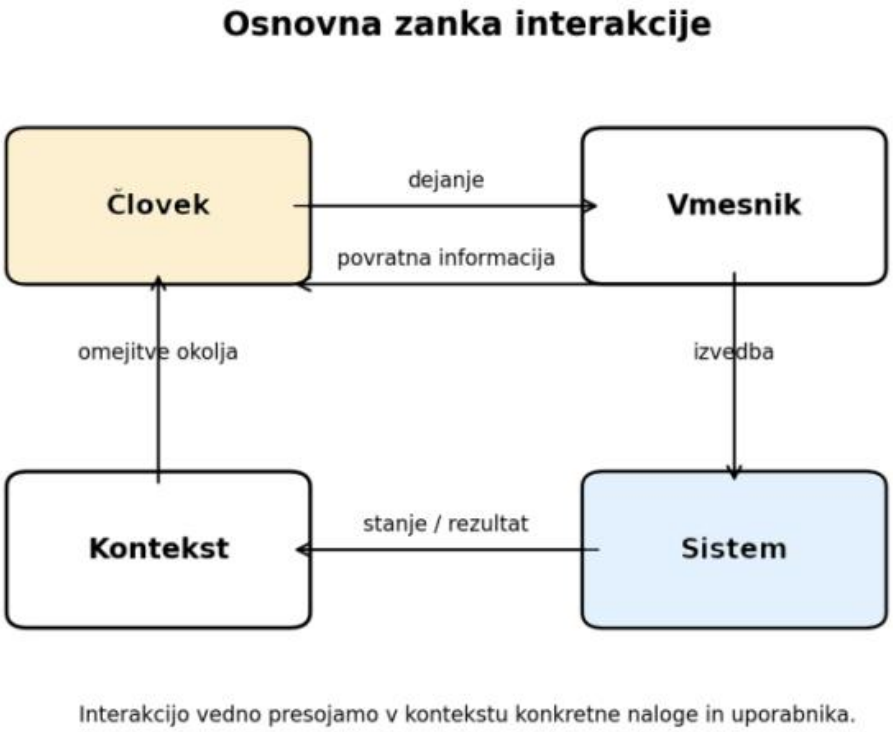
2.3 Človeške napake

Napaka ni vedno posledica malomarnosti. Uporabnik lahko klikne napačen gumb, ker sta gumba preblizu; lahko izbere napačno možnost, ker je oznaka dvoumna; ali pa razvije napačno razumevanje sistema, ker ta ne daje dovolj povratnih informacij. Pri analizi zato ne vprašamo samo »Kaj je uporabnik naredil narobe?«, temveč tudi »Kaj v vmesniku je njegovo napako omogočilo ali povečalo?«

Kako lahko vmesnik napake preprečuje in kako uporabniku pomaga pri njihovem odpravljanju, bomo podrobneje obravnavali v 9. poglavju.

2.4 Uporabniki se razlikujejo, spreminja pa se tudi kontekst

Uporabniki se razlikujejo po izkušnjah, jeziku, starosti, vidu, sluhu, motoriki, pozornosti in motivaciji. Tudi isti uporabnik se spreminja glede na okoliščine: doma je lahko zbran, na ulici raztresen, v avtomobilu pod časovnim pritiskom ali v temi slabše vidi.



Slika 2.3: Interakcija kot zanka med človekom, vmesnikom, sistemom in kontekstom.



Slika 2.4: Konkreten primer zanke interakcije

2.5 Praktična analiza človekovih omejitev

- Ali mora uporabnik kaj zapomniti, kar bi lahko bilo vidno?
- Ali so navodila razumljiva tudi brez poznavanja sistema?
- Ali se lahko uporabnik vrne in popravi odločitev?
- Ali vmesnik deluje pri povečani pisavi, slabši svetlobi ali hrupu?
- Ali je mogoče nalogo prekiniti in pozneje nadaljevati?

Vaja:

Pripravite prenatrpan zaslon. Najprej označite, kaj opazite v prvih petih sekundah. Nato navedite, katere informacije bi lahko spregledali in kako bi zaslon preuredili.

3. Ljudje s posebnimi potrebami in dostopnost

Dostopnost pomeni, da lahko sistem uporabljajo ljudje z različnimi zmožnostmi in okoliščinami. V praksi dostopnost ni omejena na formalno opredeljene skupine. Težave se lahko pojavijo zaradi staranja, poškodbe, utrujenosti, začasne omejitve, slabe svetlobe, hrupa ali uporabe naprave z eno roko.

Dostopnost je zato nadaljevanje načela iz prejšnjega poglavja: ne načrtujemo za abstraktnega povprečnega uporabnika, temveč za ljudi z različnimi sposobnostmi in v različnih okoliščinah.

3.1 Štiri pomembne skupine potreb



Slika 3.1: Primeri zahtev, ki jih moramo upoštevati pri dostopnosti.

Področje	Možne ovire	Oblikovalski odziv
Vid	slabovidnost, slepota, barvna slepota	kontrast, povečava, semantika, bralnik zaslona, neodvisnost od barve
Sluh	gluhost, naglušnost, hrup	podnapisi, prepis govora, vizualna opozorila
Motorika	tremor, omejena natančnost, uporaba ene roke	večji cilji, tipkovnica, dovolj časa, manj zahtevne geste
Kognicija	disleksija, slabša pozornost, težave s spominom	jasen jezik, struktura, doslednost, manj šuma, pomoč pri napakah

3.2 Slepi in slabovidni uporabniki

Sistem ne sme biti odvisen samo od vizualne predstavitve. Elementi morajo imeti smiselna imena, logičen vrstni red fokusa in ustrezno semantično strukturo. Slika brez opisa, gumb z nejasnim imenom ali polje brez oznake lahko postane za uporabnika bralnika zaslona praktično neuporabno.

- Kontrast naj bo zadosten, vendar barva sama ne sme nositi pomena.
- Povečava naj ne uniči postavitve ali skrije funkcij.
- Vse pomembne funkcije naj bodo dostopne s tipkovnico ali drugo alternativno metodo.
- Opis slike naj posreduje njen pomen za konkretno nalogo, ne samo videza.

3.3 Gluhi in naglušni uporabniki

Če je informacija posredovana samo z zvokom, je del uporabnikov izključen. Podnapisi morajo biti razumljivi, sinhronizirani in dovolj berljivi. Zvočna opozorila naj imajo tudi vizualni ali haptični ekvivalent, kadar je to smiselno.

3.4 Motorične in kognitivne omejitve

Pri motoričnih omejitvah so pomembni velikost ciljev, razmik, možnost upravljanja brez natančnih gest in odsotnost nepotrebnih časovnih omejitev. Pri kognitivnih težavah pomaga predvidljiva struktura, jasen jezik, omejeno število možnosti naenkrat ter sporočilo o napaki, ki pove, kako jo odpraviti.

3.5 Univerzalno oblikovanje in inkluzivni dizajn

Univerzalno oblikovanje poskuša rešitev zasnovati tako, da je uporabna za čim širši krog ljudi brez posebnih prilagoditev. Inkluzivni dizajn pa poudarja, da moramo različne uporabnike in njihove izkušnje vključiti v proces načrtovanja. To ne pomeni, da ena rešitev vedno ustreza vsem; pogosto je boljša prilagodljiva rešitev z več načini uporabe.

Pomembno
Samodejni preverjevalnik lahko odkrije manjkajoč atribut ali slab kontrast, ne more pa v celoti pokazati, ali človek dejansko razume in uspešno uporablja vmesnik.

Zato mora preverjanje dostopnosti poleg tehničnih meril vključevati tudi vrednotenje dejanske uporabe.

3.6 WCAG kot praktičen okvir

Smernice za dostopnost spletnih vsebin (WCAG – Web Content Accessibility Guidelines), ki jih razvija W3C, so pomemben praktični okvir za spletno dostopnost.

Pri poučevanju jih je smiselno povezati s konkretnimi primeri: ali je element zaznaven, ali je mogoče upravljati vmesnik, ali je vsebina razumljiva in ali deluje z različnimi tehnologijami. Skladnost s smernicami je pomembna, vendar ni nadomestilo za testiranje z uporabniki.

Vaja:

Analizirajte isti obrazec s štirih vidikov: slabovidni uporabnik, gluhi uporabnik, uporabnik z motorično omejitvijo in uporabnik z disleksijo. Vsaka skupina pripravi tri konkretne težave in tri izboljšave.

- *Katera informacija je dostopna samo z enim čutom?*
- *Ali je mogoče celoten postopek izvesti brez miške?*
- *Ali so oznake in sporočila dovolj jasna?*
- *Kaj se zgodi, če uporabnik potrebuje več časa?*
- *Kako bi predlog preverili z dejanskim uporabnikom?*

4. Modeli uporabnika in miselni modeli

4.1 Uvod

V 2. poglavju smo ugotovili, da »povprečni uporabnik« ni dobra osnova za načrtovanje. Zato pri uporabniško usmerjenem načrtovanju uporabljamo modele uporabnikov, persone, scenarije in uporabniške poti.

Pri tem razlik med uporabniki ne smemo enačiti s stereotipi. Starejši uporabnik na primer ni nujno računalniško neizkušen, mlajši uporabnik pa ni nujno vešč vseh digitalnih storitev. Zanimajo nas lastnosti, ki so povezane z uporabo konkretnega sistema.

> **Ključna ideja**
> Pri HCI ne načrtujemo za abstraktnega »povprečnega uporabnika«, temveč za ljudi z določenimi cilji, sposobnostmi, izkušnjami in okoliščinami uporabe.

4.2 Model uporabnika

Model uporabnika je poenostavljen opis značilnosti uporabnika, ki so pomembne za načrtovanje interakcije. Namen modela ni opisati človeka v vsej njegovi kompleksnosti, ampak izpostaviti lastnosti, ki lahko vplivajo na uporabo sistema.

Pri oblikovanju modela uporabnika nas lahko zanimajo predvsem:

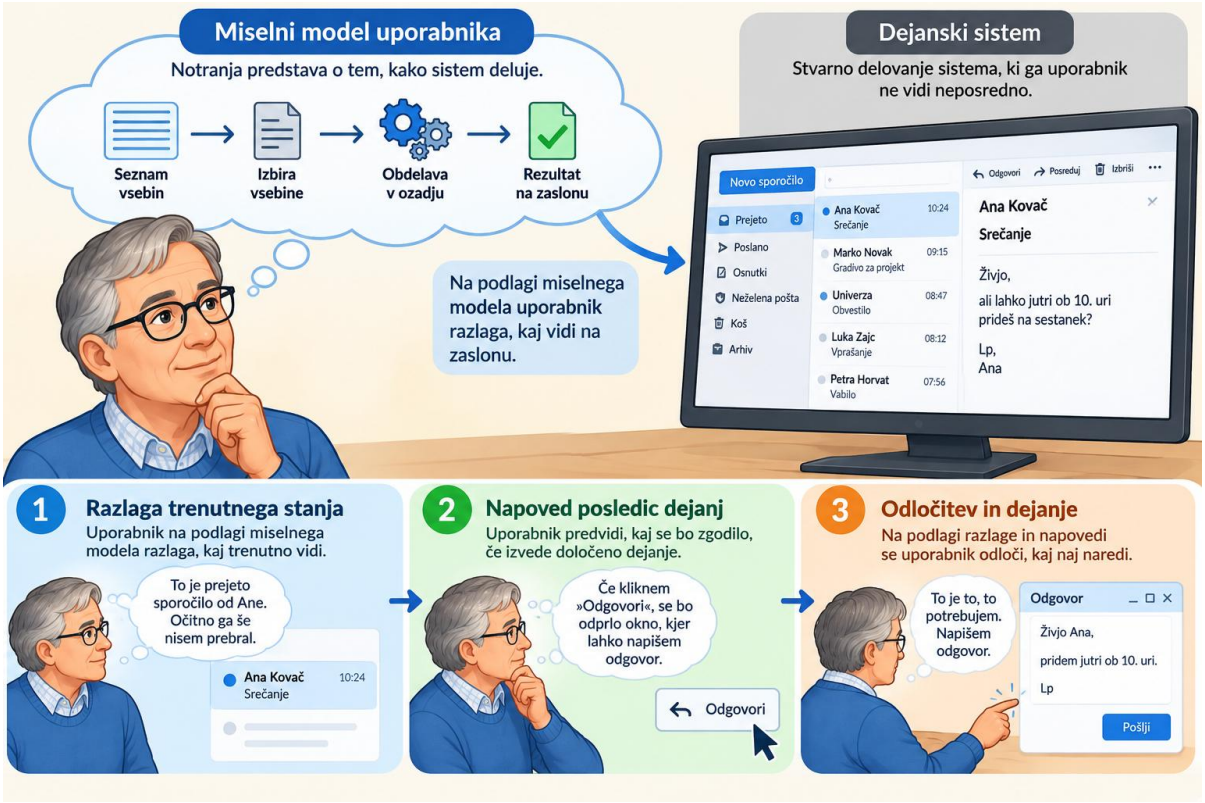
- znanje in izkušnje na področju, ki ga sistem podpira;

- izkušnje z računalniki, mobilnimi napravami in podobnimi sistemi;
- cilji, potrebe in pričakovanja;
- pogostost uporabe sistema in naloge, ki jih uporabnik opravlja;
- zaznavne, motorične in kognitivne sposobnosti;
- jezikovno in kulturno okolje;
- naprava ter fizične in družbene okoliščine uporabe.

Model uporabnika je vedno selektiven. Za spletno trgovino so pomembne druge značilnosti kot za sistem vodenja letalskega prometa. Starost sama po sebi pogosto pove manj kot izkušnje, sposobnosti, motivacija in konkretne okoliščine uporabe.

4.3 Miselni modeli

Miselni model (mental model) je uporabnikova notranja predstava o tem, kako sistem deluje. Uporabnik na njegovi podlagi razlaga trenutno stanje sistema, napoveduje posledice dejanj in se odloča, kaj naj naredi. Miselni model ni nujno tehnično pravilen; pomembno je, da uporabniku omogoča uspešno ravnanje.

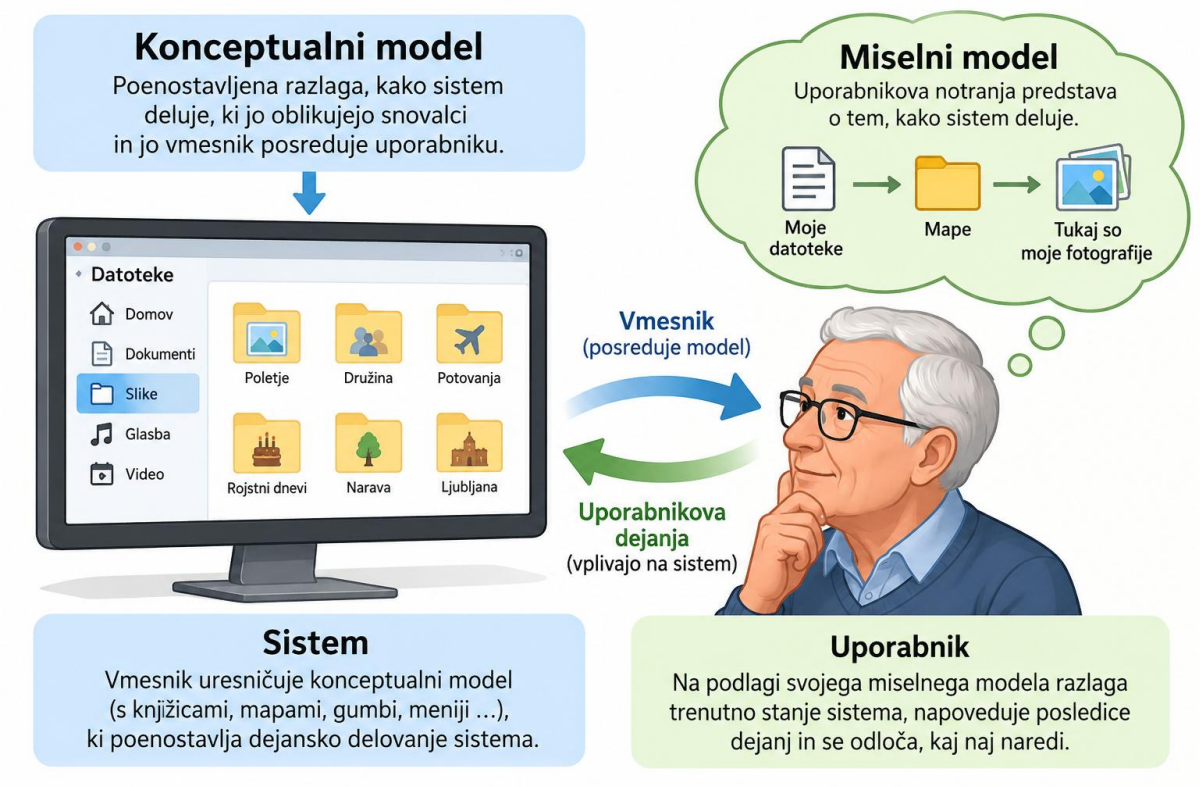


Slika 4.1: Delovanje miselnega modela uporabnika

Uporabnik elektronske pošte na primer ne potrebuje podrobnega razumevanja protokolov in strežnikov. Lahko si predstavlja, da sporočilo »pošlje« prejemniku podobno kot pismo. Tak model je poenostavljen, vendar pri običajnih nalogah dobro deluje. Težave nastanejo, kadar sistem deluje drugače, kot uporabnik pričakuje.

4.3.1 Konceptualni in miselni model

Konceptualni model je način, kako načrtovalec organizira pojme, objekte, dejanja in njihove odnose v sistemu. Miselni model pa je predstava, ki si jo na podlagi vmesnika in izkušenj ustvari uporabnik. Dober vmesnik pomaga, da se uporabnikov miselni model dovolj približa konceptualnemu modelu sistema.



Slika 4.2: Konceptualni in miselni model



Slika 4.3: Od zasnove sistema do uporabnikovega razumevanja

Vmesnik je torej posrednik med zasnovo sistema in uporabnikovim razumevanjem. Vidnost stanja, jasna poimenovanja, doslednost, povratne informacije in dobre metafore uporabniku pomagajo oblikovati ustrezen miselni model.

4.4 Zakaj »povprečni uporabnik« ni dovolj

Načrtovanje za »povprečnega uporabnika« je privlačno, ker poenostavi problem, vendar lahko zakrije pomembne razlike. Povprečje več lastnosti pogosto ne opisuje nobenega dejanskega človeka. Zato v uporabniško usmerjenem načrtovanju uporabljamo raziskovanje uporabnikov, skupine uporabnikov, persone, scenarije in uporabniške poti.

Posebej pomembno je, da lastnosti uporabnikov ne enačimo s stereotipi. Starejši uporabnik na primer ni nujno računalniško neizkušen, mlajši uporabnik pa ni nujno vešč vseh digitalnih storitev. Zanimajo nas podatki, ki so povezani z uporabo konkretnega sistema.

4.5 Persona

Persona uporabnika (krajše persona) je reprezentativen, na podatkih temelječ opis značilnega uporabnika ali skupine uporabnikov. Dobi ime in dovolj konkretnih lastnosti, da jo lahko razvojna skupina uporablja kot skupno referenčno točko pri odločanju. Persona ni statistično povprečje in ni opis konkretne osebe.

Namen persone je abstraktne podatke o uporabnikih pretvoriti v razumljivo podobo uporabnika z določenimi cilji, potrebami, vedenjskimi vzorci in omejitvami. Ko razpravljamo o novi funkciji, se lahko vprašamo: Ali jo bo ta persona potrebovala? Ali bo razumela poimenovanje? Kaj jo lahko pri nalogi ovira?

4.5.1 Kaj naj persona vsebuje

- ime ali drugo jasno oznako;
- kratek opis življenjskega ali delovnega konteksta, če je pomemben za uporabo;
- glavne cilje in naloge;
- izkušnje in relevantno znanje;
- tipične vzorce uporabe;
- potrebe, ovire in frustracije;
- naprave in okoliščine uporabe.

Fotografija, starost ali kratek življenjepis lahko persono naredijo bolj zapomnljivo, vendar ne smejo postati nadomestilo za podatke, ki so dejansko pomembni za načrtovanje.

4.5.2 Primer persone

Marija, 72 let	**Samostojna uporabnica pametnega telefona**
Cilji	Naročiti se pri zdravniku brez telefonskega klica; dobiti jasno potrditev termina.
Izkušnje	Telefon uporablja vsak dan za klice, sporočila, fotografije in splet. Pri novih aplikacijah je previdna.
Potrebe	Jasno besedilo, dovolj veliki elementi, razumljivi koraki, možnost vrnitve in vidna potrditev dejanja.
Ovire	Majhne ikone brez besedila; nejasni strokovni izrazi; strah, da bo z napačnim dotikom kaj izgubila.
Okoliščine	Aplikacijo večinoma uporablja doma na telefonu; včasih brez očal za branje.

Ta persona nam ne pove, da moramo narediti »vmesnik za starejše«. Pove pa nam, katere konkretne lastnosti vmesnika so za Marijine cilje pomembne: čitljivost, razumljiva terminologija, jasen potek, možnost popravljanja in nedvoumna povratna informacija.

4.5.3 Primarne in sekundarne persone

Pri sistemu z več skupinami uporabnikov lahko oblikujemo več person. Primarna persona predstavlja uporabnika, katerega potrebe imajo pri ključnih oblikovalskih odločitvah osrednjo vlogo. Sekundarne

persone imajo dodatne potrebe, ki jih sistem prav tako poskuša zadovoljiti, če to ne ogrozi osnovne izkušnje primarne persone. Število person naj ostane obvladljivo; njihov namen je osredotočiti načrtovanje, ne ustvariti kataloga vseh možnih ljudi.

4.5.4 Persone morajo temeljiti na podatkih

Največja nevarnost pri uporabi person je, da jih razvojna skupina preprosto izmisli. Takšna persona lahko odraža predsodke načrtovalcev namesto dejanskih uporabnikov. Boljša osnova so intervjuji, opazovanje, ankete, analiza podpore uporabnikom, dnevniški zapisi, podatki o uporabi in druge raziskovalne metode.

> **Pozor**
> Persona ni dovoljenje za stereotipiziranje. Če za neko lastnost nimamo podatkov ali ta ni pomembna za uporabo, je ne smemo obravnavati kot oblikovalsko dejstvo.

4.6 Scenariji uporabe

Persona postane za načrtovanje posebej uporabna, ko jo postavimo v konkreten scenarij. Scenarij uporabe opisuje, kdo želi nekaj doseči, kaj želi doseči, v kakšnih okoliščinah in zakaj. V zgodnji fazi se osredotočamo predvsem na uporabnikovo nalogo in ne na tehnično rešitev.

Primer: Marija želi v nedeljo zvečer rezervirati termin pri svojem osebnem zdravniku. Ambulanta je zaprta, zato uporabi mobilno aplikacijo. Želi izbrati prost termin v naslednjem tednu in biti prepričana, da je naročilo uspešno. Ne želi raziskovati vseh funkcij aplikacije; zanima jo samo doseganje konkretnega cilja.

Dober scenarij nam pomaga odkriti zahteve, ki jih seznam funkcij lahko spregleda. V tem primeru postanejo pomembni jasen prikaz prostih terminov, razumljiva potrditev rezervacije in možnost preverjanja že rezerviranega termina.

4.7 Uporabniška pot

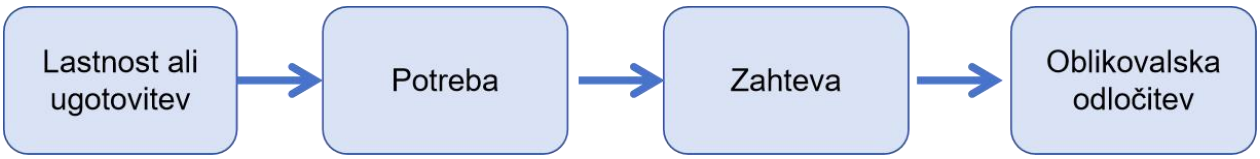
Uporabniška pot (user journey) prikaže zaporedje korakov, skozi katere gre uporabnik od začetne potrebe do cilja. Za vsak korak lahko zabeležimo uporabnikovo dejanje, vprašanja, pričakovanja, težave in priložnosti za izboljšanje vmesnika.

Korak	Dejanje	Vprašanje uporabnika	Možna težava	Oblikovalska priložnost
1	Odpre aplikacijo	Kje se naročim?	Preveč možnosti na začetnem zaslonu	Jasno izpostavljena glavna naloga
2	Izbere zdravnika	Je to moj zdravnik?	Nejasna imena ali filtri	Prepoznaven seznam in iskanje
3	Izbere termin	Kateri termini so prosti?	Slabo razločevanje prostih/zasedenih terminov	Jasna vizualna in besedna oznaka

4	Potrdi	Ali sem res naročena?	Neopazna povratna informacija	Nedvoumna potrditev z datumom in uro
5	Pozneje preveri	Kje vidim svoj termin?	Podatek je skrit	Preprost pregled prihodnjih terminov

4.8 Od ugotovitev o uporabniku do zahtev za vmesnik

Persone, scenariji in uporabniške poti niso cilj raziskovanja, ampak sredstvo za sprejemanje boljših oblikovalskih odločitev. Ključni korak je pretvorba ugotovitev o uporabnikih v zahteve za interakcijo in vmesnik.

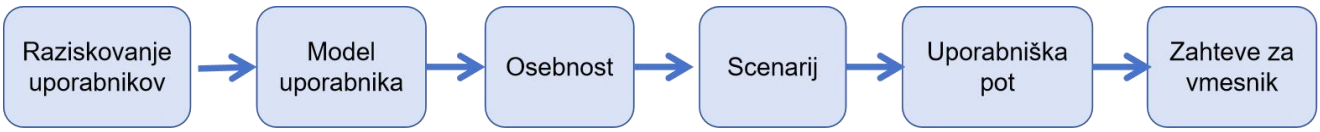


Slika 4.4: Povezava med uporabnikom in oblikovalsko odločitvijo

Ugotovitev	Potreba	Zahteva	Primer rešitve
Težje bere drobno besedilo	Dobra čitljivost	Besedilo mora biti dovolj veliko in prilagodljivo	Povečava besedila brez izgube vsebine
Boji se posledic napačnega dotika	Občutek nadzora	Pomembna dejanja morajo biti popravljiva ali potrjena	Razveljavi; potrditev pred brisanjem
Ne pozna pomena nekaterih ikon	Razumljivost	Ključne funkcije ne smejo biti odvisne samo od simbolov	Ikona skupaj z besedno oznako

4.9 Povezovanje: od raziskovanja uporabnikov do zasnove

Opisane metode lahko razumemo kot zaporedje, v katerem se podatki o resničnih uporabnikih postopoma pretvarjajo v oblikovalske odločitve. Proces ni strogo linearen. Ko izdelamo prvi predlog vmesnika in ga preskusimo, pogosto odkrijemo nova vprašanja o uporabnikih in se vrnemo k raziskovanju.



Slika 4.5: Od raziskovanja do zahtev za uporabniški vmesnik

Preskušanje in nove ugotovitve lahko proces vrnejo na prejšnje korake.

4.10 Pogoste napake

- Načrtovalec samega sebe obravnava kot tipičnega uporabnika.
- Persona je izdelana brez raziskovanja in temelji predvsem na domišljiji.
- Persone vsebujejo veliko biografskih podrobnosti, ki nimajo vpliva na uporabo sistema.
- Vse uporabnike skušamo združiti v eno »povprečno« persono.
- Scenarij že vnaprej predpisuje tehnično rešitev, namesto da bi opisal uporabnikov cilj.
- Uporabniška pot opisuje samo klike, ne pa uporabnikovih vprašanj, pričakovanj in težav.
- Ugotovitve o uporabnikih ostanejo v poročilu in se ne pretvorijo v konkretne zahteve za vmesnik.

4.11 Povzetek

Model uporabnika pomaga načrtovalcu sistematično razmišljati o značilnostih ljudi, za katere načrtuje sistem. Miselni model pa opisuje uporabnikovo predstavo o delovanju sistema. Uspešen uporabniški vmesnik pomaga uporabniku oblikovati dovolj ustrezen miselni model in zmanjšuje razkorak med tem, kar sistem omogoča, in tem, kar uporabnik pričakuje.

Persone, scenariji in uporabniške poti konkretizirajo podatke o uporabnikih. Njihova vrednost ni v dokumentu ali privlačni kartici sami, temveč v tem, da vodijo do preverljivih zahtev in boljših oblikovalskih odločitev. Temeljiti morajo na raziskovanju, uporabljati jih moramo kritično in jih po potrebi dopolnjevati z novimi ugotovitvami.

4.12 Vprašanja za razmislek

1. V čem se model uporabnika razlikuje od uporabnikovega miselnega modela sistema?
2. Zakaj je lahko načrtovanje za »povprečnega uporabnika« problematično?
3. Kakšna je razlika med persono in opisom konkretnega uporabnika?
4. Zakaj morajo persone temeljiti na podatkih?
5. Kako se scenarij uporabe razlikuje od seznama funkcionalnih zahtev?
6. Kaj nam uporabniška pot pokaže, česar sama persona ne pokaže?
7. Kako bi iz ugotovitve »uporabnik pogosto uporablja sistem med hojo« izpeljali zahteve za vmesnik?
8. Kako lahko uporabniški vmesnik vpliva na nastanek napačnega miselnega modela?

Vaje:

1.Persona za storitev. Izberite spletno ali mobilno storitev, ki jo dobro poznate. Določite ciljno skupino, zberite vsaj nekaj podatkov o njenih potrebah in izdelajte eno persono. Utemeljite, katere lastnosti ste vključili in zakaj.

2. Dve različni personi. Izdelajte dve personi za aplikacijo za naročanje pri zdravniku. Primerjajte njune cilje, izkušnje in ovire. Navedite najmanj tri oblikovalske odločitve, pri katerih bi se njune potrebe lahko razlikovale.

3. Scenarij. Za eno od person napišite kratek scenarij uporabe. Ne opisujte zaslonov in gumbov, temveč uporabnikov cilj, okoliščine in pričakovani rezultat.

4. Uporabniška pot. Iz scenarija izdelajte uporabniško pot s petimi do osmimi koraki. Pri vsakem koraku navedite dejanje, vprašanje ali pričakovanje uporabnika, možno težavo in priložnost za izboljšavo.

5. Od ugotovitve do zahteve. Izberite tri ugotovitve o uporabniku in za vsako izdelajte verigo: ugotovitev → potreba → zahteva → možna oblikovalska rešitev.

6. Analiza miselnega modela. Izberite vsakdanji interaktivni sistem. Opišite, kakšen miselni model si o njem ustvari začetnik. Poiščite mesto, kjer lahko vmesnik povzroči napačno pričakovanje, in predlagajte izboljšavo.

5. Interakcija in povratna informacija

Interakcija je izmenjava dejanj in odzivov med človekom in sistemom. Uporabnik nekaj naredi, sistem se odzove, uporabnik odziv zazna in interpretira ter izbere naslednji korak.

5.1 Interakcijska zanka

1. oblikujemo cilj
2. izberemo dejanje
3. sistem dejanje izvede ali zavrne
4. sistem prikaže spremembo oziroma rezultat
5. rezultat zaznamo in interpretiramo
6. izberemo naslednji korak

Zanka se nadaljuje, dokler ne dosežemo cilja, ne odnehamo ali ne ugotovimo, da je prišlo do napake.

Uporabnik odziv sistema interpretira na podlagi svojega miselnega modela, zato mora biti stanje sistema dovolj jasno in razumljivo.

5.2 Kaj je povratna informacija?

Povratna informacija je odziv sistema, ki pove, kaj se je zgodilo, kaj se dogaja ali kaj lahko naredimo naslednje. Lahko je vizualna, zvočna, haptična ali kombinirana.

Vrsta	Primer	Možna težava
Vizualna	sporočilo, sprememba gumba, indikator	spremembe ne opazimo
Zvočna	zvok ob opozorilu	zvoka ne slišimo ali je nejasen
Haptična	vibracija telefona	ne vemo, kaj pomeni
Besedilna	»Datoteka je shranjena«	sporočilo je dvoumno ali prepozno

5.3 Čas odziva

Pri kratkih dejanjih pričakujemo takojšnjo spremembo. Pri daljših operacijah mora sistem pokazati, da delo poteka, in po možnosti prikazati napredek. Pri napaki mora razložiti problem ter ponuditi naslednji korak.

Stanje	Ustrezna rešitev	Neustrezna rešitev
Kratko dejanje	takojšnja sprememba ali rezultat	ni jasno, ali je klik uspel
Daljša operacija	indikator napredka	zamrznjen zaslon
Napaka	razlaga in predlog rešitve	nejasna koda napake
Zaključek	potrditev in naslednji korak	vrnitev brez pojasnila

5.4 Vprašanja uporabnika

Pri interakciji si uporabnik pogosto zastavlja tri preprosta vprašanja:

- Ali je sistem zaznal moje dejanje?
- Ali se dejanje še izvaja?
- Ali je bilo uspešno?

Dobra povratna informacija mora uporabniku omogočiti, da na ta vprašanja odgovori brez ugibanja.

Vaja:

Izberite preprosto nalogo v poljubni aplikaciji in jo razdelite na posamezne korake. Pri vsakem koraku zabeležite, kakšno povratno informacijo daje sistem in ali uporabnik lahko jasno ugotovi:
– ali je sistem dejanje zaznal,
– ali se dejanje še izvaja,
– ali je bilo uspešno.

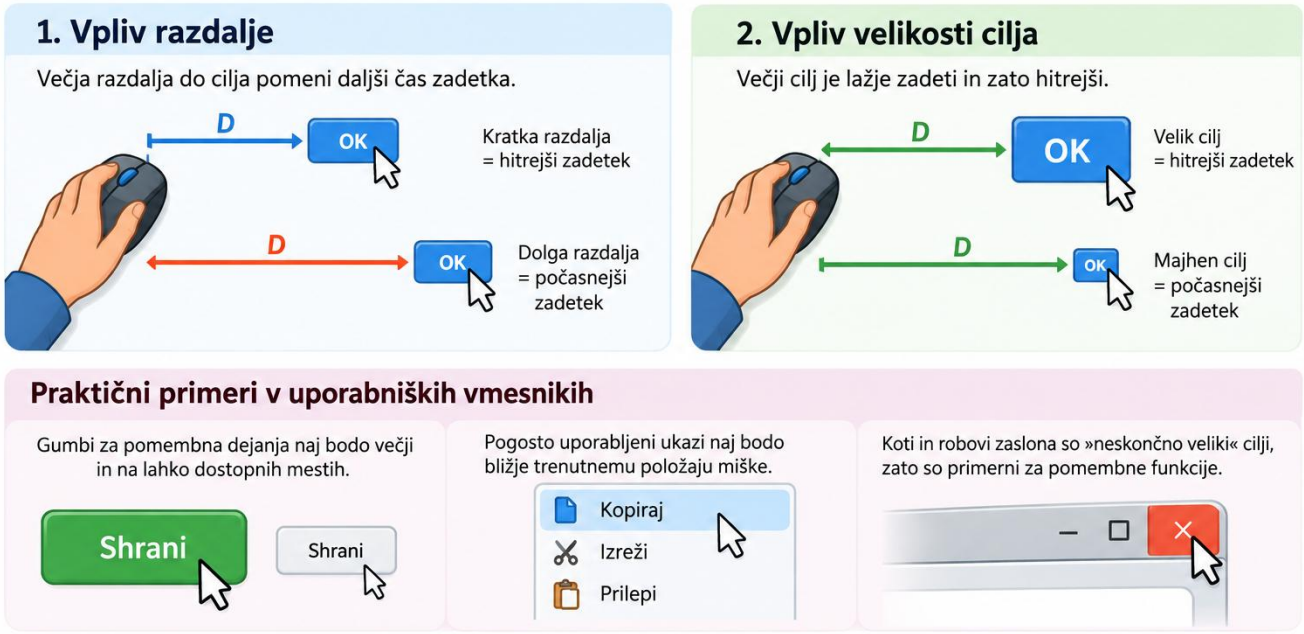
6. Zakoni UX (uporabniške izkušnje)

Zakoni UX so uporabna načela in empirične ugotovitve, ki pomagajo razložiti vedenje uporabnikov. Ne predstavljajo absolutnih pravil, temveč smernice za oblikovanje, preverjanje in utemeljevanje odločitev.

Izraz zakoni UX se v sodobni oblikovalski praksi uporablja precej široko. Nekateri izvirajo iz eksperimentalne psihologije in imajo formalnejšo empirično osnovo, drugi pa so predvsem uporabna oblikovalska načela oziroma posplošitve.

6.1 Fittsov zakon

Čas, potreben za doseganje cilja, je odvisen predvsem od razdalje do cilja in njegove velikosti. Večji in bližji gumbi so praviloma lažje dosegljivi. Na mobilnih napravah upoštevajte velikost prsta, razmik med elementi in posebej previdno namestite nevarne ukaze, kot sta *Brisanje* in *Potrdi*.



Slika 6.1: Fittsov zakon

Praktična uporaba: Pomembni interaktivni elementi naj bodo dovolj veliki in nameščeni tako, da jih uporabnik zlahka doseže. Na mobilnih napravah moramo upoštevati tudi velikost prsta in razmik med elementi.

6.2 Hickov zakon

Hickov zakon (oziroma Hick-Hymanov zakon) opisuje povezavo med številom možnosti in časom odločanja: več kot je možnosti in bolj ko so zapletene, več časa uporabnik praviloma potrebuje za odločitev.

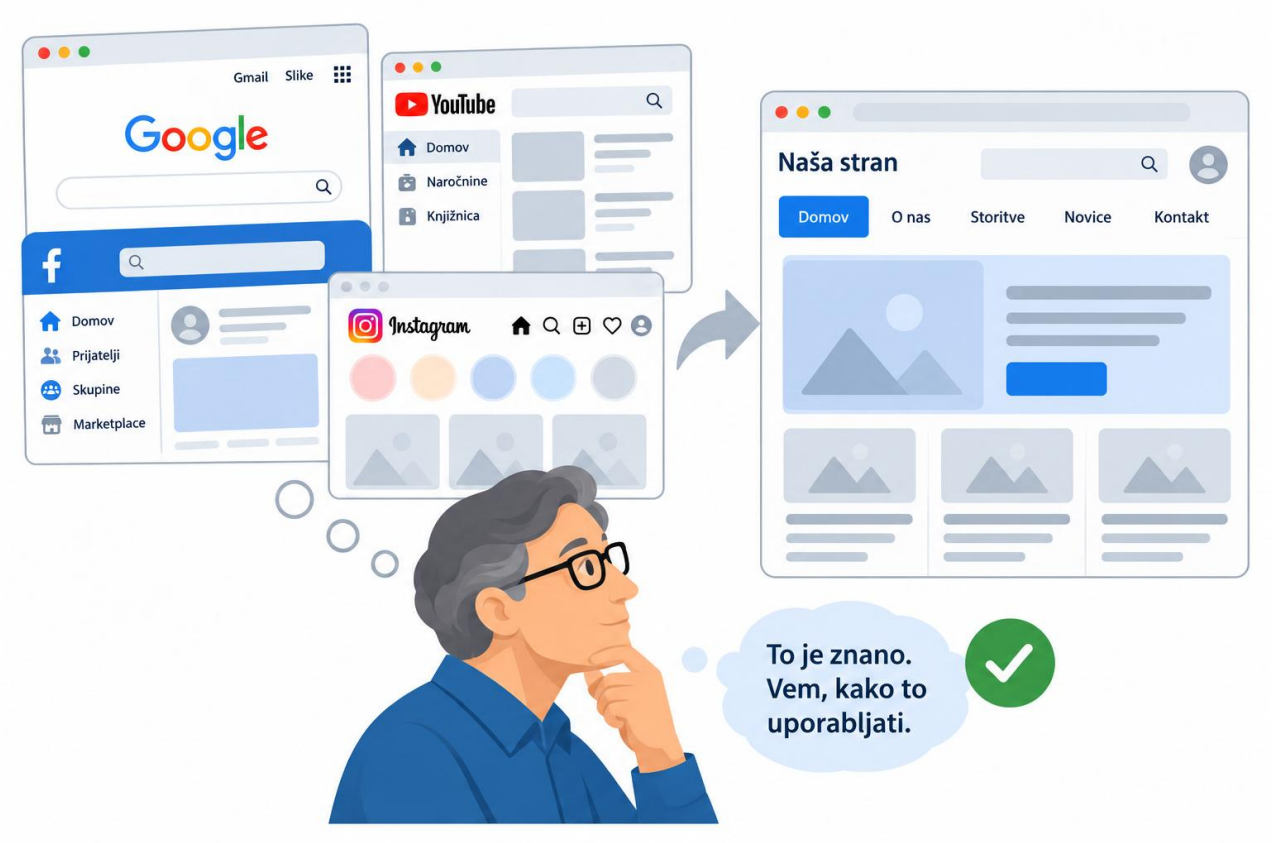


Slika 6.2: Hickov zakon

Praktična uporaba: Uporabnik, ki ga meni ne zmede, hitreje doseže želeni cilj – stran, na kateri bo opravil nakup ali stopil v stik z vami.

6.3 Jakobov zakon

Uporabniki večino svojega časa preživijo na drugih spletnih mestih in aplikacijah. To pomeni, da si želijo, da vaša stran deluje enako kot vse ostale strani, ki jih že poznajo.



Slika 6.3: Jakobov zakon

Zakaj je Jakobov zakon pomemben?

Miselni modeli: Uporabniki s seboj prinašajo že izoblikovane predstave (miselne modele) o tem, kako digitalni izdelki delujejo. Ko obiščejo novo stran, podzavestno pričakujejo enake vzorce.

Praktični primeri uporabe

Ikona nakupovalne košarice: Na spletnih trgovinah se skoraj vedno nahaja v zgornjem desnem kotu. Če bi jo postavili na dno strani, bi jo uporabniki težko našli.

Logotip podjetja: Praviloma je nameščen v zgornjem levem kotu in klik nanj vedno vodi nazaj na domačo stran.

Iskalna vrstica: Uporabniki pričakujejo, da jo bodo našli na vrhu strani, pogosto označeno z ikono povečevalnega stekla.

Ključni nauk za oblikovalce

Zakon ne pomeni, da morajo biti vse strani enake ali dolgočasne. Ustvarjalnost in inovativnost usmerite v vsebino, vizualno identiteto blagovne znamke in unikatne funkcije, medtem ko pri navigaciji in osnovnih interakcijah raje sledite uveljavljenim standardom in konvencijam.

6.4 Kognitivna obremenitev in delovni spomin

Omejitve pozornosti in delovnega spomina smo obravnavali v poglavju 2.2. Za oblikovanje iz tega izhaja preprosto načelo: zmanjšujmo količino informacij, ki jih mora uporabnik hkrati obdelovati ali si jih zapomniti. Uporabljajmo smiselne skupine, privzete vrednosti, zapletene postopke razdelimo na korake, potrebne informacije pa prikažimo tam, kjer jih uporabnik potrebuje.

6.5 Načela oblike (Gestalt)

Ta načela opisujejo, kako ljudje vizualne elemente zaznavamo in organiziramo v smiselne celote. Pri oblikovanju uporabniškega vmesnika nam pomagajo ustvariti strukturo, ki jo uporabnik hitro zazna in razume.

Tukaj so glavna "Gestaltova" načela zaznavanja:

1. Načelo bližine (Proximity)

Pravilo: Elemente, ki so postavljeni blizu drug drugemu, možgani zaznavajo kot povezano skupino ali eno celoto.

Primer: Besedilo na tej spletni strani je razdeljeno v odstavke. Ker so vrstice znotraj odstavka bližje skupaj kot posamezni odstavki med seboj, jih takoj prepoznamo kot zaključene miselne enote.

2. Načelo podobnosti (Similarity)

Pravilo: Elemente, ki si delijo podobne vizualne lastnosti (npr. barvo, obliko, velikost ali teksturo), zaznavamo kot del iste skupine ali funkcije.

Primer: Vsi spletni gumbi z isto funkcijo (npr. "Kupi zdaj") so enake barve in oblike, s čimer uporabnik takoj prepozna njihovo enako vlogo.

3. Načelo zaprtosti (Closure)

Pravilo: Če vizualni element ni popolnoma zaprt ali mu manjkajo deli, naši možgani samodejno zapolnijo praznine in zaznajo celotno obliko.

Primer: Znan logotip za Svetovni sklad za naravo (WWF), kjer črne oblike na beli podlagi takoj prepoznamo kot pando, čeprav zunanja črta pande ni zarisana.

4. Načelo kontinuitete ali neprekinjenosti (Continuity)

Pravilo: Oči naravno sledijo linijam, krivuljam ali zaporedju elementov. Elemente, razvrščene v linijo, raje povezujemo med seboj kot tiste, ki niso usklajeni.

Primer: Ko vidimo navigacijski meni s puščico ali časovnico, naše oči samodejno drsijo od ene točke do naslednje po začrtani poti.

5. Načelo figure in ozadja (Figure/Ground)

Pravilo: Vizualno polje vedno razdelimo na "figuro" (objekt v ospredju, ki mu posvečamo pozornost) in "ozadje" (prostor, v katerem se figura nahaja).

Primer: Rubinova vaza – klasična ilustracija, kjer lahko glede na osredotočenost vidite bodisi belo vazo na črni podlagi bodisi dva črna profila obrazov, ki gledata drug v drugega.

6. Načelo skupne usode (Common Fate)

Pravilo: Elemente, ki se premikajo v isto smer ali z enako hitrostjo, zaznavamo kot močno povezane ali kot del istega objekta.

Primer: Jata ptic, ki leti skupaj, ali spustni meniji v aplikacijah, kjer se vsi elementi hkrati pomaknejo navzdol.

7. Načelo enostavnosti ali dobre oblike (Prägnanz)

Pravilo: Možgani dvoumne ali kompleksne slike vedno interpretirajo v najpreprostejši možni obliki, saj to zahteva najmanj kognitivnega napora.

Primer: Olimpijski krogi – vidimo jih kot pet prepletenih krogov in ne kot zapleteno kombinacijo nepravilnih geometrijskih oblik.

6.6 Minimalizem

Vsak element vmesnika naj ima jasno funkcijo. Odstranite vsebino, ki ne pomaga pri trenutni nalogi ali tekmuje za uporabnikovo pozornost. Minimalizem ne pomeni pomanjkanja funkcij, temveč njihovo premišljeno razporeditev.

Ključne misli

- Zakoni UX pomagajo oblikovalcu predvideti težave, vendar jih je treba preverjati z dejanskimi uporabniki.
- Velikost, razporeditev, število možnosti in količina informacij neposredno vplivajo na zahtevnost interakcije.
- Dober vmesnik zmanjšuje nepotrebno odločanje in pomnjenje.

Vaje:

1. Izberite zaslon mobilne aplikacije in označite cilje, pri katerih sta pomembni velikost in oddaljenost.
2. Analizirajte meni z velikim številom možnosti. Predlagajte skupine in določite, katere možnosti bi prikazali najprej.
3. Poiščite primer znanega vzorca in primer odstopanja od pričakovanj. Ocenite posledice za uporabnika.
4. Preoblikujte obrazec tako, da si mora uporabnik zapomniti čim manj podatkov.
5. Na izbranem zaslonu poiščite primere bližine, podobnosti, poravnave in kontrasta.
6. Izberite prenatrpan zaslon in pripravite poenostavljeno različico. Utemeljite vsako odstranitev ali spremembo.

7. Informacijska arhitektura in navigacija

7.1 Kaj je informacijska arhitektura?

Uporabniški vmesnik ni samo zbirka gumbov, ikon, slik in besedila. Za temi elementi se skriva določena organizacija informacij. Uporabnik mora razumeti, katere informacije so na voljo, kako so med seboj povezane in kako pride do tistega, kar v danem trenutku potrebuje.

Temu pravimo informacijska arhitektura (Information Architecture – IA). Informacijska arhitektura določa predvsem, kako so informacije razvrščene, kako so združene v smiselne celote, kako so te celote poimenovane, kakšna je njihova hierarhija, kako uporabnik med njimi prehaja in kako poišče želeno informacijo.

Predstavljajmo si spletno trgovino z nekaj tisoč izdelki. Vsak izdelek je lahko lepo predstavljen, celoten uporabniški vmesnik pa vizualno privlačen. Če uporabnik ne zna najti zimskih čevljev svoje velikosti, je sistem zanj kljub temu slabo uporaben.

Informacijska arhitektura zato ni predvsem vprašanje videza, temveč vprašanje: Kako naj bodo informacije organizirane, da jih bo uporabnik razumel in našel?

7.2 Organiziranje informacij

Informacije lahko organiziramo hierarhično, kronološko, geografsko, abecedno, po nalogah, po uporabniških skupinah ali po temah. Nobena od teh organizacij ni sama po sebi najboljša. Pomembno je, da ustreza mentalnemu modelu uporabnika.

To lahko opazimo pri spletnih straneh javnih ustanov. Organizacija ima svojo notranjo strukturo: direktorate, oddelke, službe in sektorje. Uporabnik pa pogosto ne ve, kateri oddelek opravlja določeno nalogo. Njega zanima »Kako podaljšam dokument?« in ne »Kateri direktorat je pristojen za to področje?«. Dobra informacijska arhitektura zato ne posnema nujno organizacijske strukture ustanove.

7.3 Hierarhija

Najpogostejši način strukturiranja večje količine informacij je hierarhija. Na primer: Nastavitve → Zaslon, Zvok, Omrežje, Zasebnost, Dostopnost.

Hierarhija zmanjšuje število možnosti, ki jih mora uporabnik obravnavati hkrati. Toda pregloboka hierarhija povzroči drug problem: uporabnik se lahko začne spraševati, kje sploh je. Pri načrtovanju zato iščemo kompromis med širino in globino strukture.

7.4 Navigacija

Če informacijska arhitektura določa prostor informacij, navigacija določa, kako se uporabnik po tem prostoru premika. Dobra navigacija mora uporabniku pomagati odgovoriti na tri vprašanja: Kje sem? Kam lahko grem? Kako se lahko vrnem?

Uporabnik ne bi smel izdelati miselnega zemljevida celotnega sistema, preden lahko začne uporabljati sistem. Navigacijo omogočamo z meniji, zavihki, povezavami, navigacijskimi vrsticami, ikonami, gumbi in iskanjem. Pomembna je konsistentnost.

7.5 Globalna, lokalna in kontekstna navigacija

Globalna navigacija omogoča prehod med glavnimi deli sistema in je običajno dostopna skoraj povsod. Lokalna navigacija omogoča gibanje znotraj določenega področja. Kontekstna navigacija pa ponuja možnosti, povezane s trenutno vsebino, npr. »Podobni izdelki« ali »Preberite še«.

7.6 Drobtinice (breadcrumbs)

Pri globljih hierarhijah lahko uporabniku pokažemo pot, npr. Domov > Elektronika > Računalniki > Prenosniki. Tak prikaz imenujemo drobtinice. Njihova funkcija ni samo omogočanje vrnitve nazaj; uporabniku pokažejo tudi trenutni položaj v informacijski strukturi.

7.7 Iskanje kot del navigacije

Pri velikih informacijskih sistemih hierarhična navigacija pogosto ne zadostuje. Uporabnik, ki natančno ve, kaj išče, pogosto raje uporabi iskanje. Dobro iskanje mora upoštevati tudi nepopolne poizvedbe, tipkarske napake, sopomenke in različne izraze za isti pojem.

7.8 Prepoznavanje namesto pomnjenja

Navigacija je povezana z načelom: uporabniku omogočimo prepoznavanje možnosti, namesto da od njega zahtevamo njihovo pomnjenje. Če mora uporabnik vedeti, da se funkcija skriva pod nejasno ikono, povečujemo kognitivno obremenitev.

7.9 Primer: navigacija v avtomobilskem sistemu

Predstavljajmo si informacijsko-zabavni sistem v avtomobilu. Voznik želi med vožnjo spremeniti temperaturo. Če mora izvesti Domov → Vozilo → Udobje → Klimatizacija → Temperatura → Voznik → +, je funkcija sicer dosegljiva, vendar je informacijska arhitektura za kontekst vožnje neprimerna.



Slika 7.1: Kognitivna obremenitev voznika pri rokovanju z edutainment sistemom

Ni dovolj vprašanje »Ali lahko uporabnik pride do funkcije?«. Vprašati se moramo: Kako hitro, s koliko pozornosti in v kakšnih okoliščinah lahko pride do nje?

7.10 Ključne misli

Informacijska arhitektura organizira informacije, navigacija pa omogoča gibanje med njimi. Struktura mora ustrezati mentalnemu modelu uporabnika, ne nujno notranji strukturi organizacije ali programske opreme. Dobra navigacija uporabniku omogoča, da razume, kje je, kam lahko gre in kako se lahko vrne.

Vaje:

Izberite spletno mesto, ki ga pogosto uporabljate. Ne ocenjujte njegovega videza, ampak njegovo strukturo. Narišite drevo glavnih vsebinskih področij. Ali je struktura takšna, kot ste jo pričakovali?

Izberite eno pogosto uporabljeno funkcijo na telefonu. Zapišite vse korake od začetnega zaslona do izvedbe funkcije. Nato predlagajte pot z manjšim številom korakov.

8. Oblikovanje uporabniškega vmesnika

8.1 Od strukture do vidnega vmesnika

Ko določimo informacijsko arhitekturo, moramo informacije in funkcije predstaviti uporabniku. Vstopamo na področje oblikovanja uporabniškega vmesnika – UI design. Pri tem ne gre samo za estetiko. Lep vmesnik ni nujno uporaben vmesnik.

Naloga oblikovanja je doseči, da uporabnik hitro zazna, kaj je pomembno, kaj lahko naredi, kateri elementi sodijo skupaj, kakšno je trenutno stanje sistema in kaj se je zgodilo po njegovem dejanju.

8.2 Vizualna hierarhija

Vsi podatki niso enako pomembni. Z velikostjo, položajem, poudarkom in praznim prostorom oblikujemo vizualno hierarhijo. Uporabnik strani običajno ne bere linearno, ampak jo najprej pregleda in išče vizualne orientacijske točke.

8.3 Poravnava

Poravnava ustvarja vizualne povezave. Naključno postavljeni elementi povečajo čas, potreben za vizualno iskanje. Poravnava zato ni samo estetska odločitev, ampak vpliva na hitrost zaznavanja strukture.

8.4 Bližina

Elemente, ki pripadajo skupaj, postavimo blizu. To je povezano z načelom bližine, enim od načel oblike (Gestalt).. Z oblikovanjem uporabniku sporočamo odnose med elementi, ne da bi jih morali izrecno razlagati.

8.5 Prazen prostor

Prazen prostor ni izgubljen prostor. Z njim ločujemo skupine, poudarjamo pomembne elemente, izboljšujemo berljivost in zmanjšujemo vizualno obremenitev. Dober vmesnik ne prikaže največ informacij, temveč pravo količino informacij ob pravem času.

8.6 Tipografija

Pri tipografiji nas zanimajo velikost in vrsta pisave, debelina, razmik med vrsticami, dolžina vrstic, kontrast in hierarhija naslovov. Posebej pomembno je, da ne predpostavljamo idealnega vida uporabnika.

8.7 Barva in kontrast

Barva lahko izraža pomen, vendar pomena ne smemo kodirati samo z barvo. Namesto zgolj rdečega roba polja uporabimo tudi besedilo ali simbol. Pomemben je kontrast med besedilom in ozadjem.

8.8 Ikone

Ikone lahko prihranijo prostor in omogočijo hitro prepoznavanje pogosto uporabljanih funkcij, vendar ikona sama po sebi ni univerzalni jezik. Pri manj znanih ikonah je smiselna kombinacija ikone in besedila.

8.9 Dopustnost in zaznavni znaki

Dopustnost (affordance) opisuje možnosti dejanja, ki jih objekt omogoča. Pri grafičnih vmesnikih so pomembni tudi zaznavni znaki (signifiers), ki uporabniku povedo, kje in kako lahko deluje. Gumb mora biti videti kot nekaj, kar lahko pritisnemo.



Slika 8.1: Dopustnost in zaznavni znaki

8.10 Povratna informacija

Vsako pomembno uporabnikovo dejanje mora povzročiti razumljiv odziv sistema. Če uporabnik pritisne Pošlji, mora sistem jasno pokazati, ali je bilo sporočilo poslano. Povratna informacija je ključni mehanizem dialoga med človekom in sistemom.

8.11 Konsistentnost

Enaki elementi naj se obnašajo enako. Konsistentnost zmanjšuje potrebo po učenju. Ko se uporabnik nauči enega dela sistema, lahko pridobljeno znanje prenese na druge dele.

Vaja:

Izberite preprost obrazec in ga namenoma poslabšajte: odstranite poravnave, zmanjšajte kontrast, uporabite več pisav, odstranite besedila ob ikonah in gumbe razporedite na nepričakovana mesta. Za vsako spremembo pojasnite, zakaj je uporaba postala težja.

9. Napake, razveljavitev in obnovitev

9.1 Napake so del interakcije

Uporabniki delajo napake, vendar iz tega ne sledi, da so vse napake krivda uporabnika. Če veliko uporabnikov naredi enako napako na istem mestu, se vprašamo, ali je problem v vmesniku. Dober sistem preprečuje, zazna, pojasni in omogoči popravek napake.

9.2 Zdrsi in napake

Pri zdrsu (slip) uporabnik ve, kaj želi narediti, vendar izvede napačno dejanje. Pri napaki (mistake) pa je napačen že uporabnikov cilj ali razumevanje. Razlika je pomembna, ker zahtevata različne oblikovalske rešitve.

9.3 Preprečevanje napak

Najboljše sporočilo o napaki je tisto, ki ga uporabniku nikoli ni treba videti. Z omejitvami lahko vmesnik zmanjša prostor nesmiselnih ali nevarnih dejanj.

Na primer:

Če datum odhoda ne more biti pred datumom prihoda, naj vmesnik takšne izbire sploh ne omogoči.

9.4 Validacija vnosa

Pri vnosu podatkov mora sistem čim prej ugotoviti, ali je vnos sprejemljiv. Sporočilo mora biti razumljivo, konkretno, povezano s problemom in usmerjeno v rešitev.

9.5 Slaba sporočila o napakah

Kode napak so lahko koristne programerju, ne pa običajnemu uporabniku.



Slika 9.1: Le kaj pomeni ta koda napake

Dobro sporočilo odgovori na tri vprašanja: Kaj se je zgodilo? Zakaj je to problem? Kaj lahko naredim?

9.6 Potrjevanje nevarnih dejanj

Pri dejanjih z velikimi posledicami lahko zahtevamo potrditev. Toda potrditvenih oken ne uporabljajmo za vsako dejanje, saj se uporabnik sicer nauči avtomatično potrjevati.

9.7 Razveljavitev in obnovitev (Undo, Redo)

Pri urejevalnikih lahko "Undo" razumemo kot vračanje po zaporedju stanj, "Redo" pa kot ponovno vzpostavitev razveljavljenega dejanja. Za uporabnika je bistven občutek nadzora nad posledicami svojih dejanj.

Zelo močan pristop je omogočiti uporabniku, da dejanje preprosto razveljavi. Undo zmanjšuje strah pred uporabo sistema in uporabniku daje občutek nadzora. Pri urejevalnikih ga lahko razumemo kot vračanje po zaporedju stanj, Redo pa kot ponovno vzpostavitev razveljavljenega dejanja.

9.8 Okrevanje po napaki (recovery)

Vseh napak ne moremo preprosto razveljaviti. Program se lahko zapre, povezava se prekine ali zmanjka elektrike. Takrat potrebujemo recovery, npr. samodejno shranjevanje in obnovitev neshranjene različice.

9.9 Odpuščajoči vmesniki

Odpuščajoči vmesniki (Forgiving interfaces) uporabniku omogočajo raziskovanje brez strahu pred katastrofalnimi posledicami. Koš je klasičen primer: brisanje je začasno in omogoča poznejšo obnovitev.

Vaja.

Analizirajte brisanje datoteke v operacijskem sistemu. Ugotovite, katere napake sistem preprečuje, ali zahteva potrditev, ali omogoča Undo, ali omogoča poznejšo obnovitev in kdaj brisanje postane nepovratno.

10. Uporabniške naloge in modeli interakcije

10.1 Uporabnik ne uporablja funkcij – opravlja naloge

Programer pogosto razmišlja v funkcijah sistema, uporabnik pa v ciljih. Osnovna veriga je: cilj → naloga → podnaloga → dejanja → operacije v vmesniku.

10.2 Cilj in naloga

Cilj je stanje, ki ga želi uporabnik doseči. Naloga je dejavnost, s katero ga dosega. Razlikovanje preprečuje, da bi uporabniško nalogo opisovali samo kot zaporedje klikov.

10.3 Analiza naloge

Analiza naloge (task analysis) pomeni sistematično raziskovanje, kaj želi uporabnik doseči, katere informacije potrebuje, katere korake izvaja, kje sprejema odločitve in kje se pojavljajo težave. Analiziramo celotno dejavnost uporabnika.

10.4 Hierarhična analiza naloge

Kompleksno nalogo lahko razdelimo na podnaloge in te ponovno na manjše korake. Takšno razčlenjevanje imenujemo hierarhična analiza naloge – HTA. Namen ni izdelati čim večjega drevesa, ampak odkriti strukturo dejavnosti in problematična mesta.

Na primer, cilj »rezervirati termin pri zdravniku« lahko razdelimo na podnaloge: izbrati zdravnika → poiskati prost termin → izbrati termin → potrditi rezervacijo. Posamezne podnaloge lahko nato po potrebi razčlenimo še podrobneje.

10.5 Scenariji uporabe

Scenarij opisuje konkretno situacijo in vsebuje kontekst, ki ga abstrakten opis funkcije nima. Pomaga nam razmišljati o resnični uporabi, ne samo o tehničnih funkcijah.

10.6 Primer uporabe in scenarij

Primer uporabe (use case) opisuje interakcijo med uporabnikom oziroma drugim akterjem in sistemom pri doseganju določenega cilja. Običajno določa akterja, cilj, osnovni potek in možne alternativne poteke. Scenarij je bolj pripoveden in lahko vključuje širši kontekst uporabe, okoliščine, potrebe in pričakovanja uporabnika. Oba pristopa sta uporabna, vendar služita nekoliko različnim namenom.

10.7 Potek dela (Workflow)

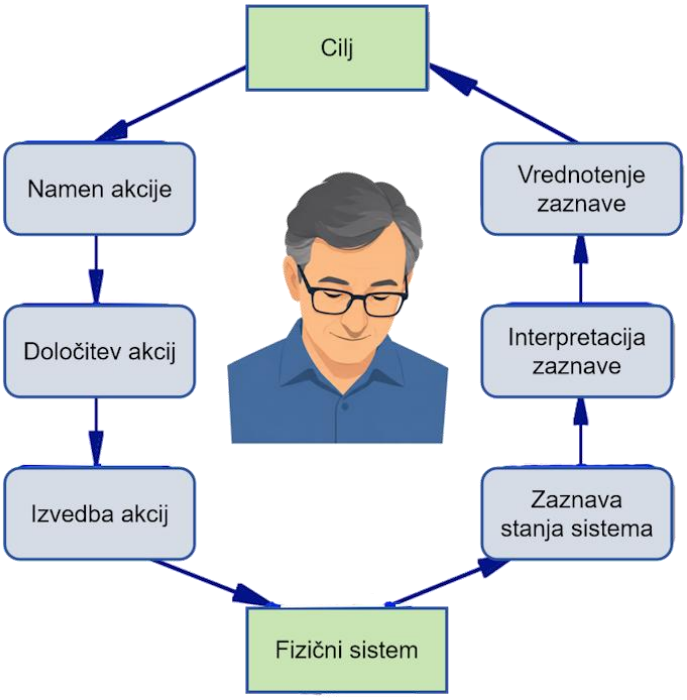
Pri poslovnih sistemih naloga pogosto prehaja med več ljudmi. Pri oblikovanju vmesnika moramo zato razumeti ne le posameznikovo nalogo, temveč tudi njeno mesto v širšem procesu.

10.8 Normanov model interakcije

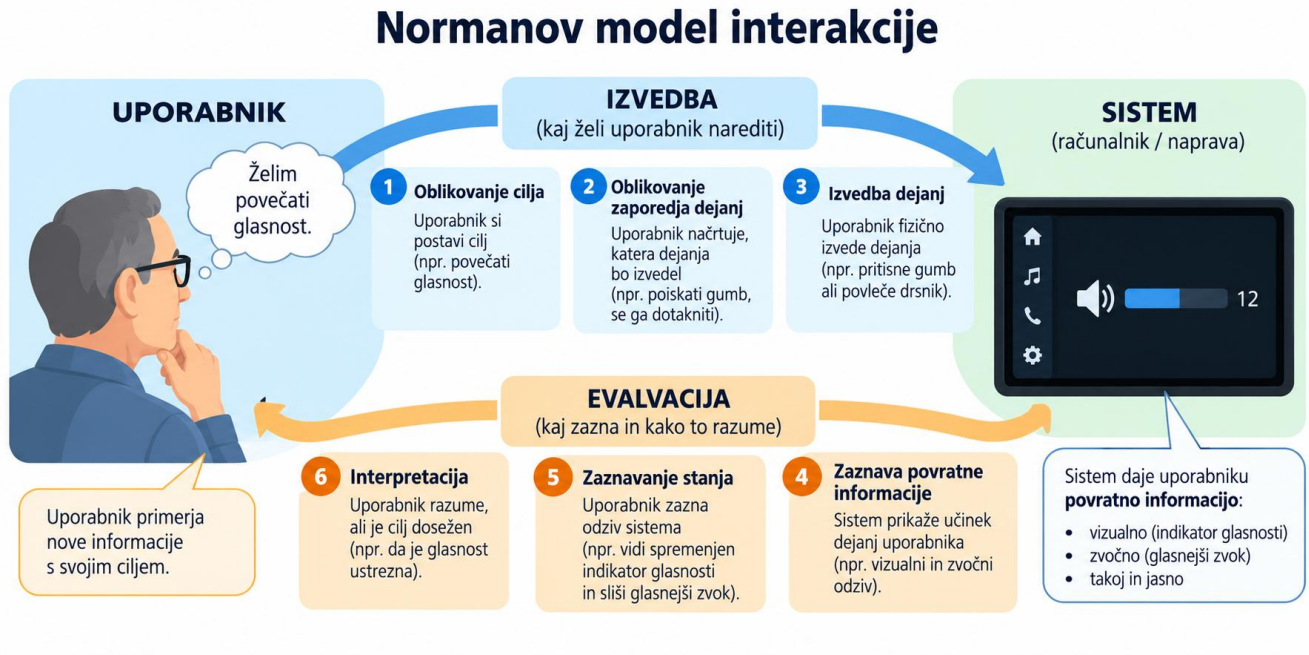
Analiza naloge opisuje, kaj želi uporabnik doseči in kako je naloga strukturirana. Model interakcije pa nam pomaga razumeti, kaj se dogaja med uporabnikom in sistemom med njenim izvajanjem.

Norman opisuje interakcijo kot sedem stopenj: oblikovanje cilja, namere, določitev dejanj, izvedba, zaznavanje stanja sistema, interpretacija in vrednotenje rezultata.

Poenostavljeno: Kaj želim? Kaj bom naredil? Naredim. Kaj se je zgodilo? Ali sem dosegel cilj?



Slika 10.1Normanov model interakcije



Slika 10.2: Primer poteka interakcije v skladu z Normanom

10.9 Razkorak (vrzel) pri izvedbi

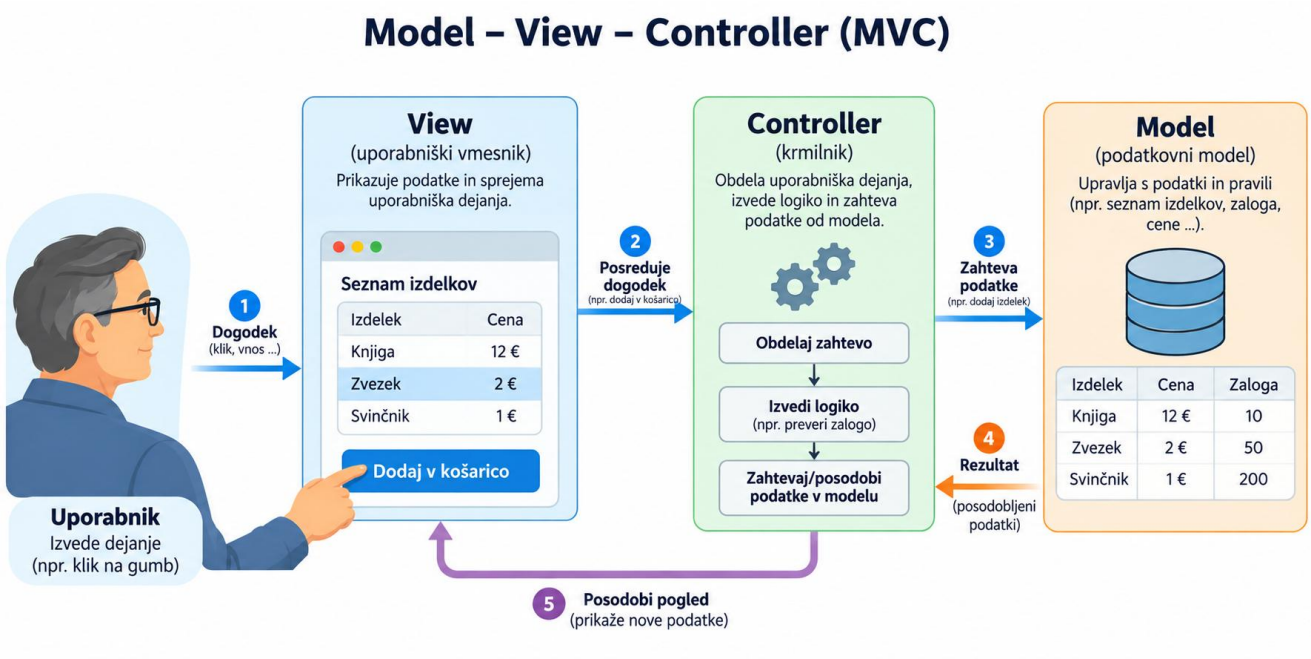
Med uporabnikovim ciljem in možnostmi vmesnika lahko obstaja vrzel – gulf of execution. Uporabnik ve, kaj želi, ne ve pa, kako naj to doseže. Dober vmesnik to vrzel zmanjšuje.

10.10 Razkorak (vrzel) pri vrednotenju

Druga vrzel se pojavi po dejanju, ko uporabnik ne razume rezultata. To je gulf of evaluation. Zmanjšujemo ga z dobro povratno informacijo in jasno predstavitvijo stanja sistema.

10.11 Model MVC

MVC ločuje Model – podatke in logiko, View – predstavitev uporabniku in Controller – obravnavo interakcije. Za HCI je pomembno, da način predstavitve uporabniku ni nujno enak notranji predstavitvi podatkov.



Slika 10.3: Ilustracija principa MVC

Vaja:

Izberite nalogo »Pošljite fotografijo prijatelju«. Razdelite jo na podnaloge in dejanja. Nato jo izvedite na telefonu in poiščite mesta, kjer nastaneta razkorak pri izvedbi in razkorak pri presoji.

11. Prototipiranje pri uporabniško usmerjenem načrtovanju

11.1Uvod: od razumevanja uporabnika do preverljive rešitve

Ko poznamo uporabnike, njihove cilje, potrebe, scenarije in uporabniške poti, še nimamo uporabniškega vmesnika. Imamo predvsem razloge, zaradi katerih mora biti vmesnik zasnovan na določen način. Naslednji korak je pretvarjanje teh ugotovitev v rešitve, ki jih lahko pokažemo, preskusimo, primerjamo in spreminjamo. Temu je namenjeno prototipiranje.

Prototip je namerno nepopolna predstavitev prihodnjega sistema. Njegov namen ni nujno pokazati, kako bo končni izdelek videti, temveč odgovoriti na določena vprašanja: Ali uporabnik razume strukturo? Ali najde pravo funkcijo? Ali je zaporedje korakov smiselno? Ali razume povratno informacijo? Ali je mogoče nalogo opraviti brez nepotrebnega napora?

Ključna ideja
Prototip izdelamo zato, da se lahko zmotimo zgodaj, hitro in poceni. Čim pozneje odkrijemo temeljno napako v zasnovi interakcije, tem dražje jo je praviloma odpraviti.

11.2 Uporabniško usmerjeno načrtovanje je iterativno

Uporabniško usmerjeno načrtovanje ni enosmerno zaporedje faz. Razumevanje uporabnikov vodi do prvih zamisli, te pretvorimo v prototip, prototip ovrednotimo z uporabniki, iz rezultatov pa nastanejo nove ugotovitve in popravki. Včasih popravimo le podrobnost, drugič moramo spremeniti informacijsko arhitekturo ali se vrniti k prvotni opredelitvi problema.

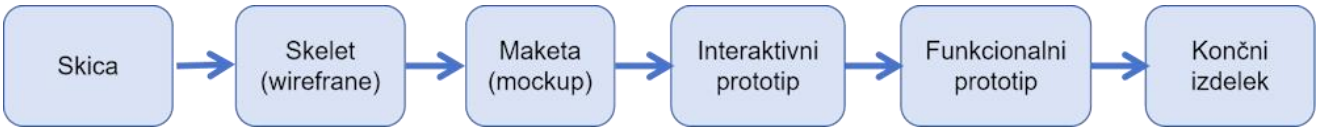


Slika 11.1: iterativni krog uporabniško usmerjenega načrtovanja

Rezultati vrednotenja lahko proces vrnejo na katerikoli prejšnji korak. Zato je koristneje govoriti o iteracijah kot o dokončno zaključenih fazah.

11.3 Razvojne stopnje prototipa

Za potrebe tega učbenika bomo uporabljali jasno delovno zaporedje: skica → wireframe → mockup → interaktivni prototip → funkcionalni prototip → končni sistem. V strokovni in praktični rabi poimenovanja niso vedno popolnoma enotna, zato je pomembneje razumeti namen posamezne stopnje kot vztrajati pri terminologiji.



Slika 11.2: šest stopenj od zamisli do sistema

1. Skica

Skica je najhitrejša in najcenejša oblika zunanjega prikaza zamisli. Lahko nastane na papirju, beli tabli ali digitalni površini. Njena vrednost je prav v nedokončanosti: spodbuja razpravo o strukturi, vsebini in poteku, ne pa o barvah, pisavah ali estetskih podrobnostih.

- primerna za raziskovanje več različnih rešitev;
- omogoča hitro zavračanje slabih zamisli;
- uporabnike in člane ekipe spodbuja, da brez zadržkov predlagajo spremembe;
- ni primerna za preverjanje natančnih vizualnih lastnosti ali tehničnega delovanja.

2. Skelet (wireframe)

Wireframe je strukturiran načrt zaslona ali strani. Določa območja, hierarhijo informacij, navigacijo in približen položaj elementov, vendar se praviloma še ne ukvarja z dokončno grafično podobo. Lahko ga primerjamo z arhitekturnim tlorisom: pokaže organizacijo prostora, ne pa še barve sten in materialov.

Vprašanje, na katero odgovarja wireframe
Ali so vsebine, funkcije in navigacijske poti organizirane tako, da uporabnik razume, kaj lahko naredi in kje to najde?

3. Maketa (mockup)

Mockup je vizualno precej natančnejši prikaz prihodnjega vmesnika. Vključuje tipografijo, barve, ikone, razmerja, grafične elemente in pogosto realistično vsebino. Običajno je statičen. Uporaben je za preverjanje vizualne hierarhije, čitljivosti, konsistentnosti in splošnega videza, vendar sam po sebi še ne pokaže, kako interakcija poteka skozi več zaslonov.

4. Interaktivni prototip

Interaktivni prototip poveže zaslone in uporabniku omogoči dejanja, kot so dotiki, kliki, odpiranje menijev in prehodi. V ozadju ni nujno prava poslovna logika ali podatkovna baza; pogosto gre za simulacijo pričakovanih odzivov. Za HCI je izredno uporaben, ker lahko z njim preskušamo potek naloge, navigacijo, razumljivost kontrol in povratne informacije, še preden razvijemo dejanski sistem.

5. Funkcionalni prototip

Funkcionalni prototip vsebuje del resničnega delovanja sistema. Lahko uporablja programsko kodo, dejanske podatke, povezave do storitev ali izbrane funkcije končne aplikacije. Ni nujno popoln ali pripravljen za produkcijsko uporabo. Namenjen je preverjanju tistih vidikov, ki jih s simulacijo ni mogoče dovolj dobro oceniti: odzivnosti, obnašanja z realnimi podatki, tehničnih omejitev ali kompleksnejše interakcije.

6. Končni sistem

Končni sistem je implementirana rešitev, namenjena dejanski uporabi. Vendar beseda »končni« ne pomeni, da je načrtovanje zaključeno. Po uvedbi dobimo nove podatke iz resnične uporabe, podpornih zahtevkov, analitike in nadaljnjih raziskav. Uporabniško usmerjeno načrtovanje se zato nadaljuje tudi po izdaji izdelka.

11.4 Primerjalna tabela šestih stopenj

Stopnja	Glavni namen	Realističnost	Interaktivnost	Strošek spremembe	Tipično vprašanje
Skica	Raziskovanje zamisli	zelo nizka	brez ali simulirana	zelo nizek	Ali je osnovna zamisel smiselna?
Skelet (Wireframe)	Struktura in razporeditev	nizka	majhna	nizek	Ali uporabnik najde pravo vsebino/funkcijo?
Maketa (Mockup)	Vizualna zasnova	srednja do visoka	navadno brez	nizek do srednji	Ali je hierarhija jasna in prikaz čitljiv?
Interaktivni prototip	Potek interakcije	visoka	visoka, simulirana	srednji	Ali lahko uporabnik uspešno opravi nalogo?

Funkcionalni prototip	Preverjanje dejanskega delovanja	visoka	visoka	višji	Kako rešitev deluje z realno logiko/podatki?
Končni sistem	Dejanska uporaba	polna	polna	najvišji	Kako sistem deluje v resničnem okolju?

11.5 Prototipi nizke in visoke vernosti

Pogosto prototipe delimo tudi glede na stopnjo podobnosti s končnim sistemom. Low-fidelity prototipi so hitri, grobi in poceni; mednje sodijo zlasti skice in preprosti wireframi. High-fidelity prototipi so po videzu ali obnašanju bližje končnemu izdelku. Meja med skupinama ni ostra.

Visoka realističnost ni samodejno prednost. Če želimo preveriti informacijsko arhitekturo, lahko podroben grafični prototip preusmeri pozornost na barve in estetiko. Če želimo preveriti razumljivost animacije ali občutek neposredne manipulacije, pa papirnata skica ne bo zadostovala. Stopnjo realističnosti izberemo glede na raziskovalno vprašanje.

> **Pravilo izbire**
> Ne vprašajmo: »Kako realističen prototip lahko naredimo?« Vprašajmo: »Kaj želimo s prototipom izvedeti in koliko realističnosti za to potrebujemo?«

11.6 Rdeča nit: Marijina aplikacija za naročanje pri zdravniku

Marija želi s telefonom rezervirati termin pri zdravniku. Na tem primeru lahko spremljamo razvoj vmesnika skozi vseh šest stopenj.

1. Skica: Na papir narišemo tri glavne zaslone: začetni zaslon, izbiro termina in potrditev. Namesto podrobnosti nas zanima, ali je pot od cilja do rezultata kratka in razumljiva.

2.Skelet: Določimo položaj naslova, zdravnika, koledarja, seznama prostih terminov ter gumba za potrditev. Preverimo, ali so ključne možnosti dovolj vidne in ali je vrstni red smiseln.

3. Maketa: Dodamo realistično tipografijo, velikosti elementov, kontraste, ikone in dejansko besedilo. Posebej preverimo čitljivost ter ali ikona koledarja potrebuje tudi besedno oznako.

4. Interaktivni prototip: Marija lahko izbere zdravnika, termin in pritisne »Potrdi«. Prehodi so simulirani. Opazujemo, ali razume naslednji korak in ali opazi potrditev rezervacije.

5. Funkcionalni prototip: Prototip pridobi resnične ali testne termine iz podatkovnega vira, prepreči rezervacijo že zasedenega termina in shrani izbrani termin. Preverjamo tudi odzivnost in ravnanje ob napakah.

6. Končni sistem: Rešitev je povezana z dejanskim sistemom naročanja, preverjena glede varnosti, dostopnosti in zanesljivosti ter uvedena v uporabo. Po uvedbi spremljamo težave in nadaljujemo izboljšave.



Slika 11.3: isti zaslon v šestih razvojnih stopnjah

Slika je bolj shematična: prikazuje predvsem naraščanje določnosti, vizualne podrobnosti, interaktivnosti in tehničnega delovanja.

11.7 Kaj vrednotimo na posamezni stopnji

Vsak prototip naj ima jasno vprašanje vrednotenja. Če uporabniku pokažemo prototip brez opredeljenega cilja, lahko dobimo veliko komentarjev, vendar malo uporabnih ugotovitev. Pri zgodnjih prototipih nas zanimajo predvsem koncept, struktura in potek; pri poznejših pa tudi podrobnosti interakcije, realni podatki, odzivni časi, dostopnost in robustnost.

- Pri skici: razumevanje osnovne zamisli in alternativnih poti.
- Pr iskeletu: informacijska arhitektura, prioritete in navigacija.
- Pri maketi: vizualna hierarhija, čitljivost, kontrast in doslednost.
- Pri interaktivnem prototipu: uspešnost nalog, zaporedje korakov, povratne informacije in napake.
- Pri funkcionalnem prototipu: obnašanje z realnimi podatki, odzivnost in tehnične omejitve.
- Pri končnem sistemu: uporaba v realnem okolju, dolgoročne težave in nove potrebe.

11.8 Horizontalni in vertikalni prototipi

Druga koristna delitev se nanaša na obseg. Horizontalni prototip pokaže širok nabor funkcij, vendar vsako le površinsko. Primeren je za preverjanje celotne strukture sistema. Vertikalni prototip pa podrobno izdela ozek del funkcionalnosti, na primer celoten postopek rezervacije termina, medtem ko so druge funkcije še nedodelane. V praksi pogosto kombiniramo oba pristopa.

11.9 Prototip ni samo slika zaslona

Pri prototipiranju se zlahka osredotočimo samo na vizualni videz. Toda interakcija vključuje tudi čas, odziv sistema, spremembe stanja, napake, prekinitve, povratne informacije in obnovitev po napačnem dejanju. Dober prototip mora zato, kadar je to pomembno za raziskovalno vprašanje, pokazati tudi ta vedenja.

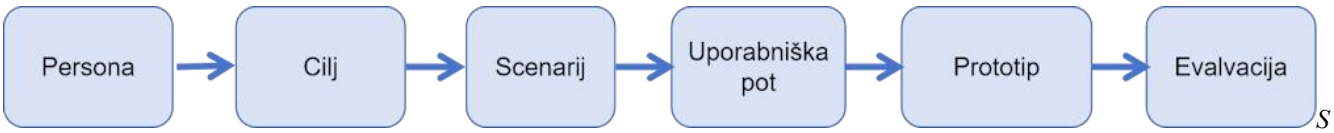
Za Marijo na primer ni dovolj, da je gumb »Potrdi termin« velik in čitljiv. Pomembno je, kaj se zgodi po pritisku: ali sistem takoj pokaže napredek, ali jasno potrdi datum in uro, ali prepreči dvojno oddajo ter ali Marija pozneje najde rezervacijo.

11.10 Pogoste napake pri prototipiranju

- Prehitro preskočimo na podroben grafični prototip in se prezgodaj navežemo na eno rešitev.
- Prototip izdelamo, ne da bi vedeli, katero vprašanje želimo z njim preveriti.
- Uporabnikom razlagamo, kako naj prototip uporabljajo, namesto da bi opazovali, kaj razumejo sami.
- Zamenjujemo vizualno dovršenost z dobro uporabnostjo.
- Preskušamo samo idealni potek in pozabimo na napake, vračanje ter prekinitve.
- Po testiranju zberemo pripombe, vendar ne spremenimo zasnove.
- Funkcionalni prototip postopoma postane produkcijski sistem, čeprav njegova programska zasnova za to ni bila primerna.

11.11 Od uporabnika do preverljive rešitve

Prototipiranje ni ločena dejavnost od modeliranja uporabnikov. Persona pomaga določiti, za koga načrtujemo. Scenarij določi cilj in okoliščine. Uporabniška pot razkrije korake in težavne točke. Prototip pa omogoči, da predlagano rešitev teh težav naredimo vidno in preverljivo.



lika 11.4: Od persone uporabnika do evalvacije aplikacije

Vaje:

1. Od skice do skeleta. Izberite preprosto mobilno storitev. Narišite najmanj tri različne skice začetnega zaslona. Ne izbirajte prve zamisli avtomatično. Primerjajte rešitve in najboljšo pretvorite v skelet.

2. Marijino naročanje. Na podlagi persone Marije izdelajte skelet postopka naročanja pri zdravniku. Pot naj vsebuje največ toliko korakov, kolikor jih lahko utemeljite z uporabnikovo nalogo. Nato s sošolcem izvedite kratko preskušanje.

3. Isti vmesnik, različna vernost. Isti zaslon predstavite kot skico, skelet in maketo. Pri vsaki različici napišite tri vprašanja, na katera bi lahko z njo smiselno odgovorili, ter eno vprašanje, za katero ta stopnja ni primerna.

4. Papirnati interaktivni prototip. V paru pripravite papirnati prototip s petimi zasloni. Eden igra sistem in menja liste glede na dejanja drugega, ki igra uporabnika. Zabeležite mesta, kjer uporabnik okleva ali izbere nepričakovano pot.

5. Horizontalno ali vertikalno. Za aplikacijo za univerzitetne študente določite primer horizontalnega in vertikalnega prototipa. Pojasnite, katera raziskovalna vprašanja bi preverjali z vsakim.

6. Načrt vrednotenja. Za interaktivni prototip določite tri naloge, ki jih bo opravil uporabnik, merila uspeha in podatke, ki jih boste opazovali. Izogibajte se vprašanju »Ali vam je všeč?« kot glavnemu merilu.

7. Iteracija. Po preskusu prototipa razvrstite opažene težave po pomembnosti. Izberite tri spremembe, izdelajte novo različico in ponovno preskusite vsaj eno ključno nalogo.

Vprašanja za razmislek

1. Zakaj najbolj realističen prototip ni vedno najboljši prototip?

2. Kakšna je razlika med wireframom in mockupom?

3. Kdaj zadostuje simulirana interaktivnost in kdaj potrebujemo funkcionalni prototip?

4. Zakaj je pomembno opredeliti raziskovalno vprašanje pred izdelavo prototipa?

5. Kaj pridobimo z več različnimi skicami istega problema?

6. Kako se horizontalni prototip razlikuje od vertikalnega?

7. Zakaj izraz »končni sistem« pri uporabniško usmerjenem načrtovanju ne pomeni konca načrtovanja?

8. Na kateri stopnji bi preverjali čitljivost, na kateri informacijsko arhitekturo in na kateri odziv z realnimi podatki?

Povzetek

Prototipiranje je osrednji mehanizem, s katerim uporabniško usmerjeno načrtovanje preide od razumevanja problema k preverjanju rešitev. Skica, wireframe, mockup, interaktivni prototip in funkcionalni prototip niso zgolj stopnje vse večje lepote ali tehnične dovršenosti. Vsaka oblika je raziskovalno orodje, primerno za druga vprašanja.

Dobro prototipiranje je hitro, namensko in iterativno. Rešitve izdelujemo dovolj podrobno, da lahko preverimo trenutno vprašanje, nato jih preskusimo, spremenimo in ponovno preverimo. S tem zmanjšujemo tveganje, da bi šele v končnem sistemu odkrili, da smo zelo učinkovito izdelali napačno rešitev.

Opomba o orodjih

Konkretna programska orodja za skiciranje, tvorbo skeletov, vizualno oblikovanje, interaktivne in funkcionalne prototipe v tem poglavju namenoma niso navedena. Orodja se hitro spreminjajo, temeljna načela prototipiranja pa ostajajo. Pregled aktualnih kategorij in primerov orodij je zato primerneje vzdrževati v dodatku učbenika.

12. Evalvacija uporabniškega vmesnika

12.1 Zakaj vmesnik vrednotimo?

Oblikovalec svoj vmesnik pozna. Ve, kaj pomenijo ikone, kje so funkcije in kakšen je predvideni vrstni red korakov. Uporabnik tega znanja nima. Zato trditev »Meni se zdi popolnoma jasno« ni dokaz, da je vmesnik razumljiv.

Evalvacija uporabniškega vmesnika je sistematično preverjanje, kako dobro vmesnik podpira uporabnike pri njihovih nalogah. Z njo poskušamo ugotoviti, ali uporabniki razumejo možnosti sistema, uspešno opravijo naloge, opazijo povratne informacije, se znajdejo ob napakah in dosežejo cilj brez nepotrebnega napora.

Evalvacija ni namenjena dokazovanju, da smo izdelali dober vmesnik. Njena največja vrednost je pogosto prav v tem, da odkrije probleme, ki jih načrtovalci niso pričakovali.

Ključna ideja

Ne vrednotimo uporabnika. Vrednotimo, kako dobro vmesnik podpira uporabnika pri njegovi nalogi.

Če uporabnik ne najde gumba, napačno razume oznako ali ne opazi, da je bilo dejanje uspešno izvedeno, nas ne zanima predvsem, kaj je uporabnik naredil narobe. Zanima nas, **zakaj mu je vmesnik omogočil takšno razumevanje.**

12.2 Kaj lahko vrednotimo?

Pri evalvaciji lahko opazujemo različne lastnosti uporabe:

- **uspešnost** – ali uporabnik doseže cilj;
- **učinkovitos**t – koliko časa, korakov in napora potrebuje;
- **napake** – katere napake se pojavijo in kako resne so;
- **naučljivos**t – kako hitro se nov uporabnik nauči uporabe;
- **zapomnljivost** – kako hitro se po daljšem premoru ponovno znajde;
- **zadovoljstvo** – kako uporabnik doživlja uporabo;
- **razumljivost** – ali razume stanje sistema in posledice svojih dejanj;
- **dostopnost** – ali lahko sistem uporabljajo ljudje z različnimi sposobnostmi in načini interakcije.

Samo ena mera praviloma ne pove dovolj. Uporabnik lahko na primer nalogo uspešno opravi, vendar potrebuje dvakrat več časa, kot smo pričakovali. Lahko jo opravi hitro, vendar pri tem ni prepričan, ali je bil rezultat pravilen.

Zato pri evalvaciji pogosto kombiniramo **kvantitativne podatke** – čas, število napak, uspešnost – in **kvalitativne ugotovitve** – kaj je uporabnika zmedlo, kaj je pričakoval in zakaj je izbral določeno pot.

12.3 Formativna in sumativna evalvacija

Glede na namen razlikujemo predvsem med **formativno** in **sumativno evalvacijo**.

Formativno evalvacijo izvajamo med načrtovanjem in razvojem. Njeno glavno vprašanje je:

Kaj moramo izboljšati?

Uporabljamo jo pri skicah, skeletih, prototipih in zgodnjih različicah sistema. Rezultat evalvacije niso predvsem številčne ocene, ampak ugotovljeni problemi in predlogi sprememb.

Sumativna evalvacija pa bolj odgovarja na vprašanja:

Kako dobro sistem deluje? Ali smo dosegli zastavljena merila? Kako se dve rešitvi primerjata?

Uporabna je pri bolj izdelanih sistemih, ko želimo meriti uspešnost, primerjati različice ali preveriti doseganje vnaprej določenih zahtev.

Meja med obema ni vedno ostra. Pomembno je predvsem, da **evalvacije ne pustimo samo za konec projekta**.

12.4 Evalvacija brez uporabnikov in z uporabniki

Metode evalvacije lahko zelo grobo razdelimo v dve skupini.

Pri **strokovnem pregledu** lahko strokovnjak analizira vmesnik brez neposredne vključitve uporabnikov. Išče odstopanja od uveljavljenih načel, probleme konsistentnosti, nejasne povratne informacije, slabo preprečevanje napak in podobno.

Pri **testiranju z uporabniki** pa opazujemo predstavnike ciljnih uporabnikov med izvajanjem konkretnih nalog.

Metodi se dopolnjujeta.

Strokovnjak lahko hitro odkrije veliko znanih oblikovalskih problemov, vendar ne more zanesljivo predvideti vseh odzivov resničnih uporabnikov. Uporabnik pa lahko pokaže težavo, ki je strokovnjak sploh ni pričakoval.

Pomembno

Strokovnjak ni nadomestilo za uporabnika, uporabniški test pa ne pomeni, da strokovni pregled ni potreben.

12.5 Hevristična evalvacija

Pri hevristični evalvaciji strokovnjak pregleda vmesnik glede na uveljavljena načela oziroma hevristike uporabnosti. Najbolj znan nabor takšnih načel je deset Nielsenovih hevristik uporabnosti, ki se pogosto uporabljajo kot kontrolni seznam pri strokovnem pregledu vmesnika.

Med tipičnimi vprašanji so:

- Ali sistem uporabniku jasno kaže svoje trenutno stanje?
- Ali uporablja jezik in pojme, ki jih uporabnik razume?
- Ali ima uporabnik občutek nadzora?
- Ali je vmesnik konsistenten?
- Ali sistem preprečuje napake?
- Ali uporabniku omogoča prepoznavanje namesto pomnjenja?
- Ali so sporočila o napakah razumljiva in uporabna?
- Ali uporabnik dobi pomoč, kadar jo potrebuje?

Hevristična evalvacija je relativno hitra in jo lahko izvedemo že na prototipu. Posebej koristna je pred uporabniškim testom, saj lahko z njo odstranimo očitne probleme, preden porabimo čas uporabnikov.

Vendar ima omejitev: strokovnjak vmesnik presoja na podlagi svojega znanja in izkušenj. Ne more popolnoma simulirati začetnika, starejšega uporabnika ali človeka, ki sistem uporablja v stresni situaciji.

Resnost problema

Pri odkritih težavah ni dovolj zapisati samo, da problem obstaja. Koristno je oceniti tudi njegovo pomembnost.

Vprašamo se:

Kako pogosto se problem pojavi? Kako močno ovira nalogo? Ali ga uporabnik lahko sam odpravi? Kakšne so posledice?

Napačna barva manj pomembne ikone in gumb, zaradi katerega uporabnik pomotoma izbriše svoje delo, očitno nista enako pomembna problema.

12.6 Kognitivni sprehod

Hevristični evalvaciji je sorodna metoda kognitivnega sprehoda (cognitive walkthrough). Pri njej se osredotočimo predvsem na vprašanje, ali bo uporabnik znal ugotoviti, **kaj mora narediti v naslednjem koraku**.

Za vsak korak pomembne naloge se vprašamo:

- Ali bo uporabnik vedel, kaj želi v tem koraku doseči?
- Ali bo opazil ustrezno možnost oziroma kontrolnik?
- Ali bo razumel, da ta možnost vodi proti njegovemu cilju?
- Ali bo po izvedenem dejanju razumel odziv sistema?

Metoda je posebej uporabna pri sistemih, ki jih mora uporabnik razumeti brez predhodnega usposabljanja.

12.7 Testiranje z uporabniki

Pri testu uporabnosti **uporabniku damo nalogo**, ne navodil za uporabo.

Namesto:

Kliknite »Termini«, nato izberite zdravnika in pritisnite »Rezerviraj«.

rečemo:

Naročite se pri svojem zdravniku za prvi prosti termin prihodnji teden.

V prvem primeru smo uporabniku povedali rešitev. V drugem lahko opazujemo, ali mu vmesnik omogoča, da rešitev odkrije sam.

Pri uporabniškem testu zato velja pomembno načelo:

Ne testiramo, ali je uporabnik dovolj sposoben uporabljati naš vmesnik. Testiramo, ali je naš vmesnik dovolj razumljiv, da ga uporabnik lahko uporablja.

12.8 Načrt uporabniškega testa

Dober uporabniški test se začne pred prihodom prvega udeleženca. Opredeliti moramo predvsem:

Kaj želimo izvedeti?

Na primer:

- Ali uporabnik najde funkcijo za naročanje?
- Ali razume razliko med prostim in zasedenim terminom?
- Ali opazi, da je rezervacija uspešno zaključena?
- Ali zna rezervacijo preklicati?

Nato določimo:

Kdo naj sodeluje?
Udeleženci naj bodo dovolj podobni ciljnim uporabnikom glede tistih lastnosti, ki so pomembne za obravnavano nalogo.

Katere naloge bodo izvajali?
Naloge naj bodo realistične in oblikovane kot cilji, ne kot navodila za uporabo vmesnika.

Kaj bomo opazovali in merili?
Vnaprej določimo, katere podatke potrebujemo.

Kako bomo rezultate zabeležili?
Uporabimo lahko zapiske, časovne meritve, snemanje zaslona ali druge metode, če so primerne in če so udeleženci z njimi ustrezno seznanjeni.

12.9 Izbira nalog

Naloga je eden najpomembnejših delov uporabniškega testa. Slabo oblikovana naloga lahko uporabniku nehote razkrije rešitev.

Slabo:

Poiščite meni »Nastavitve«, izberite »Zasebnost« in izklopite deljenje lokacije.

Bolje:

Ne želite več, da aplikacija uporablja vašo lokacijo. Spremenite nastavitve tako, da je ne bo mogla uporabljati.

Druga različica opisuje **cilj, uporabnik pa mora sam ugotoviti pot.**

Naloge naj pokrivajo najpomembnejše in najbolj tvegane dele sistema. Ni potrebno testirati vsakega gumba.

12.10 Moderator in opazovalec

Moderator uporabniku predstavi nalogo, skrbi za potek testa in po potrebi postavlja dodatna vprašanja.

Njegova največja skušnjava je, da uporabniku **prehitro pomaga.**

Če uporabnik deset sekund išče funkcijo, je tišina moderatorju lahko neprijetna. Toda prav teh deset sekund je morda najpomembnejši rezultat testa.

Moderator zato ne sme razlagati vmesnika, kadar želimo preveriti njegovo razumljivost.

Namesto:

Poglejte zgoraj desno.

lahko pozneje vpraša:

Kaj ste pričakovali, da boste našli?

Cilj ni uporabnika spraviti v zadrego, ampak razumeti njegovo razmišljanje.

12.11 Opazovanje

Med testom ne beležimo samo tega, ali je bila naloga uspešno končana.

Opazujemo tudi:

- kje uporabnik začne;
- katere elemente opazi;
- kje okleva;
- kam se vrača;
- katere možnosti poskusi;
- kaj spregleda;
- katere napake naredi;
- kakšen odziv sistema pričakuje;
- ali opazi povratno informacijo.

Posebej zanimiva so mesta, kjer uporabnik naredi nekaj, česar načrtovalec ni pričakoval. Takšno vedenje pogosto razkrije **razliko med konceptualnim modelom sistema in miselnim modelom uporabnika**, ki smo ju obravnavali v 4. poglavju.

12.12 Razmišljanje na glas

Pri metodi razmišljanja na glas (think aloud) uporabnika prosimo, naj med izvajanjem naloge pove, kaj razmišlja.

Na primer:

»Mislim, da moram pritisniti ta simbol, ker izgleda kot koledar.«

ali:

»Ne vem, ali je termin že rezerviran ali moram še nekaj potrditi.«

S tem dobimo vpogled v uporabnikov miselni model. Ne izvemo samo, **kje je nastala težava**, ampak pogosto tudi **zakaj**.

Vendar ima metoda omejitev. Govorjenje lahko nekoliko spremeni način izvajanja naloge in ga upočasni. Zato časa, izmerjenega med razmišljanjem na glas, ne smemo nekritično obravnavati kot običajni čas uporabe.

12.13 Kvantitativne meritve

Pri uporabniškem testu lahko merimo na primer:

Meritev	Kaj nam pove
delež uspešno opravljenih nalog	ali uporabniki dosežejo cilj
čas izvedbe	koliko časa zahteva naloga
število napak	kje in kako pogosto prihaja do težav
število korakov	kako neposredna je pot
število vračanj	ali uporabnik pogosto zaide
zahteve za pomoč	koliko samostojnosti omogoča vmesnik

Primer:

Meritev	Vmesnik A	Vmesnik B
uspešno opravljena naloga	70 %	95 %
povprečni čas	85 s	48 s
povprečno št. napak	2,1	0,6

Takšni podatki pokažejo, da **razlika obstaja**, ne povedo pa nujno, **zakaj**.

Zato številčne meritve dopolnimo z opazovanjem in pogovorom z uporabniki.

12.14 Subjektivno zadovoljstvo

Po uporabi lahko uporabnika vprašamo, kako zahtevna, razumljiva ali prijetna se mu je zdela uporaba.

Subjektivno zadovoljstvo je pomemben del uporabniške izkušnje, vendar ga ne smemo enačiti z objektivno učinkovitostjo.

Možno je, da uporabnik reče:

»Aplikacija mi je zelo všeč.«

hkrati pa med testom trikrat zaide, potrebuje pomoč in ne uspe dokončati ključne naloge.

Nasprotno je lahko zelo učinkovit sistem vizualno manj privlačen.

Obe ugotovitvi sta veljavni — merita različna vidika izkušnje.

Subjektivno oceno lahko pridobimo z lastnimi vprašanji ali s standardiziranimi vprašalniki, na primer z vprašalnikom SUS (System Usability Scale).

12.15 Koliko uporabnikov potrebujemo?

Odgovor je odvisen od cilja evalvacije.

Pri zgodnjem **formativnem testiranju lahko že majhno število dobro izbranih uporabnikov** odkrije pomembne težave. Smisel zgodnjega testa ni statistično dokazati, koliko odstotkov populacije bo imelo težavo, ampak čim prej odkriti probleme in izboljšati zasnovo.

Po spremembi lahko izvedemo nov krog testiranja.

Pri **formalni raziskav**i, primerjavi populacij ali statističnem posploševanju pa potrebujemo ustrezno raziskovalno zasnovo in **dovolj velik vzorec**.

Zato vprašanje »Koliko uporabnikov je dovolj?« nima enega univerzalnega odgovora. Najprej moramo vedeti, kaj želimo z evalvacijo ugotoviti.

12.16 Primer: evalvacija Marijine aplikacije

Vrnimo se k Mariji iz prejšnjega poglavja.

Marijin cilj je naročiti se pri zdravniku. Za test pripravimo nalogo:

»Naročite se pri svojem zdravniku za prvi prosti termin prihodnji teden.«

Opazujemo:

- Ali Marija najde funkcijo za naročanje?
- Ali pravilno izbere zdravnika?
- Ali razume prikaz prostih terminov?
- Ali zna izbrati termin?
- Ali razume, da mora izbiro potrditi?
- Ali po potrditvi ve, da je naročilo uspešno zaključeno?
- Ali zna pozneje najti podatek o svojem terminu?

Morda ugotovimo, da več uporabnikov po izbiri termina obstane, ker ne opazijo gumba *Potrdi*.

To ni samo ugotovitev:

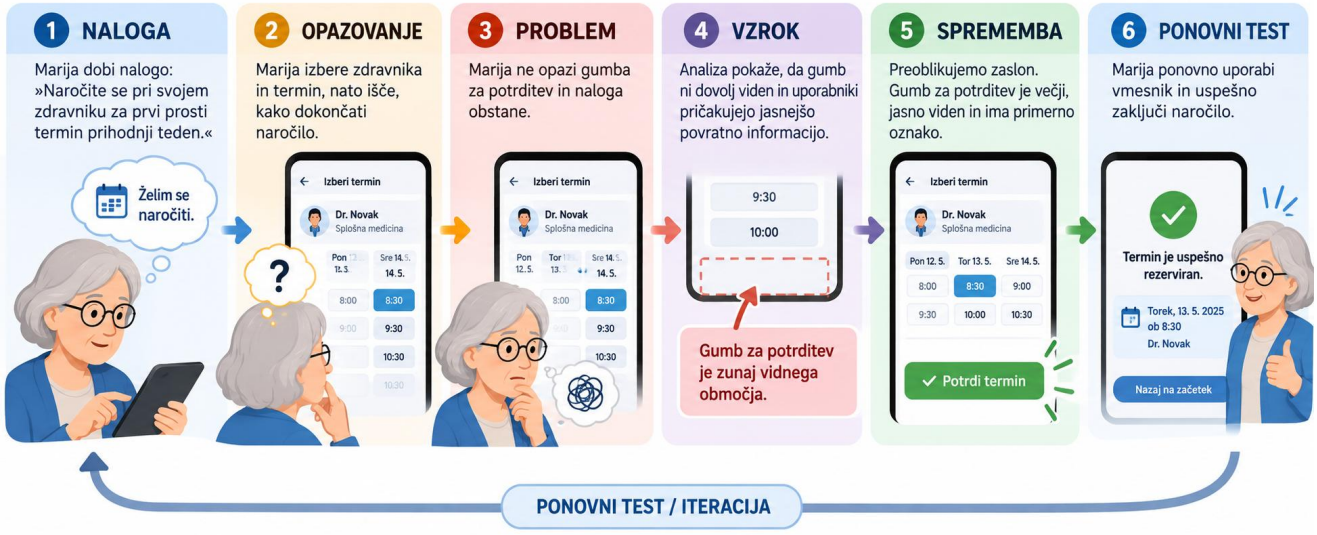
»Uporabniki niso našli gumba.«

Za načrtovanje je pomembnejše vprašanje:

Zakaj ga niso našli?

Morda je gumb zunaj vidnega območja. Morda nima dovolj vizualnega poudarka. Morda uporabniki pričakujejo samodejno potrditev. Morda oznaka ni razumljiva.

Šele odgovor na to vprašanje vodi do dobre spremembe vmesnika.



Slika 12.1: od opažanja delovanja Marije do izboljšave vmesnika

Zgornja slika demonstrira potek evalvacije, ki v splošnem poteka tako:

NALOGA → OPAZOVANJE → PROBLEM → VZROK → SPREMEMBA → PONOVNI TEST

In v Marijinem primeru:

Izbere termin → ne opazi gumba *Potrdi* → naloga obstane → gumb ni dovolj viden → preoblikujemo zaslon → ponovno testiramo.

12.17 Od podatka do ugotovitve

Pri evalvaciji je pomembno **razlikovati med opažanjem in razlago**.

Opažanje:

Trije uporabniki so po izbiri termina več kot deset sekund iskali naslednji korak.

Razlaga:

Gumb za potrditev je premalo viden.

Druga trditev je že interpretacija. Morda je pravilna, lahko pa je pravi razlog nejasen miselni model postopka.

Zato je koristno zapisovati:

Kaj smo opazili? → Kaj bi lahko bil vzrok? → Kako bomo hipotezo preverili?

S tem zmanjšamo nevarnost, da bi na podlagi enega vedenja prehitro sprejeli napačno oblikovalsko odločitev.

12.18 Razvrščanje ugotovljenih problemov

Po testu lahko dobimo dolg seznam težav. Vseh ne moremo vedno odpraviti hkrati.

Probleme lahko razvrstimo glede na:

- vpliv na dokončanje naloge;
- pogostost pojavljanja;
- posledice napake;
- možnost uporabnikovega okrevanja;
- pomembnost naloge.

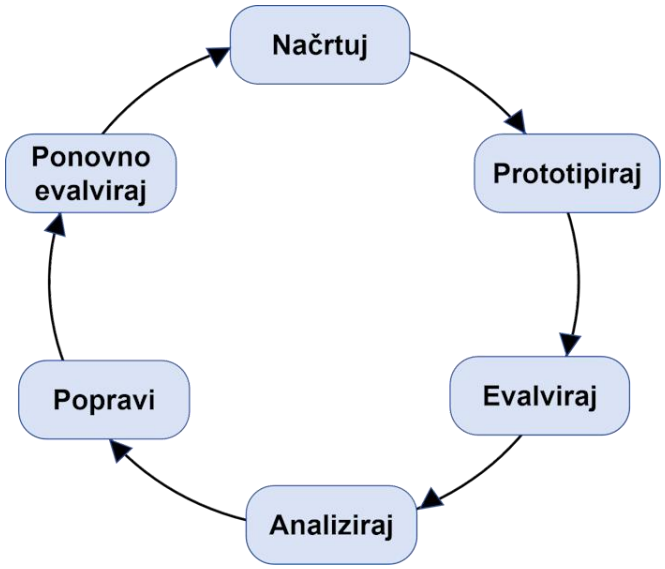
Prednost ima praviloma problem, ki uporabniku prepreči dokončanje pomembne naloge ali povzroči resne posledice.

Pri tem ne smemo avtomatično zanemariti problema samo zato, ker se je pojavil pri enem uporabniku. Redka težava ima lahko zelo velike posledice.

12.19 Evalvacija je del iterativnega procesa

Rezultat evalvacije ni poročilo, ki ga shranimo v arhiv.

Rezultat mora biti sprememba zasnove. Proces kaže spodnja slika:



Slika 12.2: Iterativni proces evalvacije

Tako se 11. in 12. poglavje neposredno povežeta.

Prototipiranje brez evalvacije pomeni, da izdelujemo različice, ne da bi vedeli, ali so boljše. Evalvacija brez nadaljnjih sprememb pa pomeni, da smo težave sicer odkrili, vendar jih nismo uporabili za izboljšanje sistema.

12.20 Pogoste napake pri evalvaciji

Pri evalvaciji se pogosto pojavijo naslednje napake:

- testiramo prepozno, ko so velike spremembe že drage;
- izberemo udeležence, ki niso podobni ciljnim uporabnikom;
- uporabniku razložimo, kako naj uporablja vmesnik;
- naloge vsebujejo besede, ki razkrivajo rešitev;

- spraŠujemo predvsem »Ali vam je všeč?«;
- moderator prehitro pomaga;
- beležimo samo uspeh ali neuspeh, ne pa poti;
- iz enega dogodka prehitro sklepamo na vzrok;
- zberemo veliko pripomb, vendar jih ne razvrstimo po pomembnosti;
- po evalvaciji vmesnika ne spremenimo;
- po spremembi ne preverimo, ali smo problem res odpravili.

12.21 Vprašanja za razmislek

- Zakaj načrtovalec težko objektivno presoja lasten vmesnik?
- Kakšna je razlika med formativno in sumativno evalvacijo?
- Zakaj hevristični pregled ne more v celoti nadomestiti uporabniškega testa?
- Zakaj uporabniku pri testu podamo cilj in ne zaporedja korakov?
- Kaj izgubimo, če merimo samo čas izvedbe naloge?
- Kaj pridobimo z metodo razmišljanja na glas in kakšna je njena omejitev?
- Zakaj majhno število uporabnikov lahko zadostuje za zgodnjo formativno evalvacijo, ne pa nujno za statistično posploševanje?
- Kakšna je razlika med opažanjem in interpretacijo opažanja?
- Kako določimo, kateri ugotovljeni problem je najpomembnejši?
- Zakaj se evalvacija ne konča z izdelavo poročila?

12.22 Povzetek

Evalvacija je osrednji del uporabniško usmerjenega načrtovanja. Z njo preverjamo, kako dobro vmesnik podpira resnične uporabnike pri njihovih nalogah.

Uporabimo lahko strokovne metode, kot sta hevristična evalvacija in kognitivni sprehod, ter metode z uporabniki, pri katerih opazujemo izvajanje realističnih nalog. Kvantitativni podatki povedo, kaj in koliko se je zgodilo, kvalitativne ugotovitve pa nam pomagajo razumeti zakaj.

Najpomembnejši rezultat evalvacije ni ocena vmesnika, temveč boljša naslednja različica vmesnika.

Vaje:

1. Hevristični pregled. Izberite spletno stran ali aplikacijo. Poiščite najmanj pet problemov uporabnosti. Za vsak problem navedite, katero načelo krši, kakšne so možne posledice in kako bi ga izboljšali.

2. Kognitivni sprehod. Izberite nalogo, ki jo mora nov uporabnik opraviti brez navodil. Za vsak korak preverite, ali bo uporabnik opazil pravo možnost, razumel njen pomen in pravilno interpretiral rezultat.

3. Oblikovanje nalog. Napišite tri naloge za uporabniški test. Nato preverite, ali ste v besedilu nehote uporabili izraze z vmesnika in s tem razkrili rešitev.

4. Test s sošolcem. Izvedite kratek test ene naloge. Med testom ne pomagajte. Zabeležite vsa mesta oklevanja, vračanja in nepričakovane izbire.

5. Opažanje ali razlaga? Iz zapiskov testa ločite objektivna opažanja od svojih interpretacij. Za vsako interpretacijo poiščite vsaj eno alternativno razlago.

6. Iteracija. Na podlagi testa spremenite tri elemente vmesnika. Nato ponovno testirajte isto nalogo in primerjajte rezultate.

13. Mobilni, spletni in multimodalni HCI

13.1 Računalnik ni več samo računalnik

Velik del zgodnjega HCI je obravnaval uporabnika pred računalnikom z zaslonom, tipkovnico in miško Danes uporabljamo telefone, tablice, pametne ure, televizorje, avtomobilske sisteme, pametne zvočnike, slušalke, očala XR in gospodinjske naprave. Interakcija se je preselila iz računalnika v okolje.

To pomeni pomembno spremembo za HCI. Uporabnik ne sedi več nujno pred napravo in ji namenja vse svoje pozornosti. Lahko hodi, vozi, govori z drugim človekom ali opravlja drugo nalogo.

Ključna ideja:

Pri sodobnem HCI ne načrtujemo samo za napravo, ampak za uporabnika, nalogo in okoliščine uporabe.

13.2 Posebnosti mobilne interakcije

Mobilni telefon ima majhen zaslon, vendar težava ni samo njegova velikost. Telefon uporabljamo med hojo, na soncu, v temi, na avtobusu, z eno roko, med pogovorom, ob prekinitvah in v hrupnem okolju. Mobilni HCI je zato predvsem HCI spremenljivega konteksta.

Na uporabo vplivajo vsaj štirje dejavniki:

Dejavnik	Primer	Posledica za načrtovanje
naprava	majhen zaslon, dotik	večji in jasni elementi
okolje	sonce, hrup, gibanje	dober kontrast, več načinov opozarjanja
uporabnik	uporablja eno roko	dosegljivi in dovolj veliki cilji
naloga	prekinitev, časovni pritisk	kratki postopki, možnost nadaljevanja



Slika 13.1 – Kontekst mobilnega uporabnika

.

13.3 Dotik ni miška

Pri miški imamo natančen kazalec. Pri telefonu uporabljamo prst, ki je relativno velik, zakriva del zaslona, nima stanja hover in ni tako natančen. Zato morajo biti pomembne tarče dovolj velike in dovolj narazen. Potreba po dovolj velikih in dosegljivih tarčah je neposredno povezana tudi s Fittsovim zakonom, ki smo ga spoznali v 6. poglavju.

Pomembna je tudi lega elementov. Pri uporabi telefona z eno roko nekaterih delov zaslona ne dosežemo enako zlahka. Pogoste in pomembne funkcije naj zato ne zahtevajo nepotrebnega preprijemanja naprave.

Mobilnega vmesnika ne načrtujemo kot pomanjšane namizne aplikacije.

13.4 Geste

Dotik omogoča *tap*, *dvojni tap*, *dolg pritisk*, *poteg*, *vlečenje*, *uščip* (*pinch)* in *rotacijo*. Geste omogočajo neposredno in hitro interakcijo, vendar imajo pomembno pomanjkljivost: pogosto niso vidne.

Gumb uporabniku sporoča, da je mogoče nekaj narediti. Pri gesti *poteg* tega morda ne ve, dokler je ne pozna.

Zato velja:

- ključne funkcije naj ne bodo dostopne samo s skrito gesto;
- geste naj bodo skladne z običajnimi pričakovanji;
- sistem naj po gesti takoj pokaže njen učinek;
- pomembna dejanja naj bo mogoče razveljaviti.

Primer:

Brisanje sporočila s potegom je hitro za izkušenega uporabnika, začetnik pa te možnosti morda nikoli ne odkrije. Zato je smiselna tudi druga, vidna pot do funkcije. Potreba po dovolj velikih in dosegljivih tarčah je neposredno povezana tudi s Fittsovim zakonom, ki smo ga spoznali v 6. poglavju.

13.5 Odzivna spletna zasnova

Odzivna zasnova (responsive design) ne pomeni samo spreminjanja širine elementov. Pri manjšem prostoru moramo ponovno razmisliti o prioritetah vsebine, navigaciji, velikosti elementov, dolžini besedila in načinu interakcije.

Ista spletna stran na velikem monitorju in telefonu torej ni nujno samo večja oziroma manjša različica iste postavitve.

Na namiznem zaslonu je lahko hkrati prikazanih več informacij in stalni navigacijski meni. Na telefonu mora biti struktura pogosto bolj zaporedna, najpomembnejša dejanja pa bolj izpostavljena.

Vprašanje za načrtovalca:

Kaj uporabnik v tem trenutku res potrebuje na tem zaslonu?

To vprašanje je pomembnejše od vprašanja, kako na majhen zaslon spraviti vse elemente velikega zaslona.

Spletni vmesnik ima še eno posebnost: uporabnik lahko nanj pride neposredno na skoraj katerokoli stran, na primer prek iskalnika ali povezave. Zato ne moremo predpostavljati, da je do trenutne strani prišel po poti, ki smo si jo zamislili. Vedno mora dovolj jasno razumeti, kje je, kaj lahko naredi in kako lahko nadaljuje.

13.6 Multimodalna interakcija

Multimodalni sistem podpira več komunikacijskih kanalov. Vhod so lahko tipkovnica, miška, dotik, govor, pogled, gesta, gibanje in kamera; izhod pa slika, besedilo, zvok, govor, vibracija, haptični odziv in prostorski prikaz.

Prava moč multimodalnosti nastane, ko se modalnosti **dopolnjujejo**.

V avtomobilu lahko uporabnik reče:

»Pelji me domov.«

Sistem lahko ukaz sprejme z govorom, odgovori z glasom in pot hkrati prikaže na zemljevidu.

Izbira modalnosti je odvisna od konteksta. Govor je lahko zelo primeren med vožnjo, precej manj pa v hrupnem prostoru ali na sestanku. Zaslon je primeren za pregled zemljevida, manj pa takrat, ko mora uporabnik gledati drugam.

Ključna ideja:

Najboljša modalnost ni vedno ista. Odvisna je od uporabnika, naloge in okolja.

13.7 Redundanca in dostopnost

Če isto pomembno informacijo posredujemo po več kanalih, povečamo robustnost. Uporabnik morda ne vidi zaslona, ne sliši zvoka ali je okolje hrupno. Multimodalnost je zato pomembna tudi za dostopnost.

Opozorilo lahko na primer združuje: SLIKO + ZVOK + VIBRACIJO

Vendar več kanalov ni samodejno boljše. Če vsi hkrati zahtevajo uporabnikovo pozornost, lahko povečajo kognitivno obremenitev.

Pomembna informacija naj zato ne bo odvisna od enega samega čuta, različni kanali pa naj se smiselno dopolnjujejo.



Slika 13.2 – Večkanalno opozorilo

13.8 Prehajanje med napravami

Uporabnik pričakuje kontinuiteto: na telefonu poišče hotel, na računalniku ga rezervira, navigacija v avtomobilu pa ga naslednji dan pripelje do njega. Enota načrtovanja zato ni več nujno posamezen zaslon ali naprava, ampak **celotna uporabniška izkušnj**a.



Slika 13.3 – Kontinuiteta naloge med napravami

Pri prehodu med napravami naj uporabniku ne bo treba ponovno vzpostavljati konteksta. Sistem naj, kadar je to smiselno in dovoljeno, ohrani:

- trenutno stanje naloge,
- pomembne podatke,
- uporabnikove izbire,
- zgodovino,

- mesto, kjer je uporabnik nalogo prekinil.

Vendar kontinuiteta odpira tudi vprašanje **zasebnosti**. Podatek, primeren za osebni telefon, ni nujno primeren za zaslon avtomobila, televizor ali drugo napravo, ki jo vidi več ljudi.

13.9 Prekinitve in pozornost

Mobilne naprave niso samo orodja, ki jih uporabnik aktivno uporablja. **Z obvestili tudi same zahtevajo njegovo pozornost.**

Vsako obvestilo zato pomeni oblikovalsko odločitev:

Ali je informacija dovolj pomembna, da uporabnika prekinemo prav zdaj?

Pri tem sta pomembni dve lastnosti:

- pomembnost – kako pomembna je informacija za uporabnika;
- nujnost – kako hitro mora uporabnik nanjo reagirati.

Zelo pomembna, vendar nenujna informacija lahko počaka. Nujno varnostno opozorilo pa mora biti opazno takoj.

Preveč prekinitev povzroča izgubo pozornosti, povečuje kognitivno obremenitev in uporabnika spodbuja, da začne opozorila ignorirati.

	Ni nujno	**Nujno**
Pomembno	obvesti pozneje	opozori takoj
Manj pomembno	zabeleži/združi	kratko opozorilo

13.10 Povzetek

Mobilni HCI ni samo HCI na manjšem zaslonu. Naprave uporabljamo v spreminjajočem se okolju, z omejeno pozornostjo in z različnimi načini interakcije.

Dotik zahteva drugačno zasnovo kot miška, geste morajo biti odkrijljive, odzivna spletna zasnova mora upoštevati prioritete vsebine, multimodalnost pa omogoča izbiro primernega komunikacijskega kanala glede na kontekst.

Ko uporabnik prehaja med napravami, predmet načrtovanja postaja kontinuiteta celotne naloge, ne samo posamezni zaslon.

Vaje:

1. Mobilni kontekst. Izberite mobilno aplikacijo in jo uporabite v treh različnih okoliščinah, na primer sede, med hojo in na prostem pri močni svetlobi. Zabeležite, kateri problemi se spremenijo s kontekstom.

2. Dotik in geste. Poiščite tri funkcije, dostopne z gesto. Preverite, ali lahko nov uporabnik odkrije, da te funkcije obstajajo.

3. Odzivna zasnova. Isto spletno stran odprite na velikem in majhnem zaslonu. Primerjajte navigacijo, prioritete vsebine in način interakcije. Ne ocenjujte samo videza.

4. Multimodalnost. Izberite eno nalogo in predlagajte tri različne načine interakcije. Za vsakega navedite okolje, v katerem je primeren, in okolje, v katerem ni.

5. Obvestila. Deset obvestil iz izbrane aplikacije razvrstite glede na pomembnost in nujnost. Za vsako določite: "opozori takoj", "obvesti pozneje" ali " zabeleži".

14. Umetna inteligenca in HCI

14.1 Nova vrsta uporabniškega vmesnika

Klasični uporabniški vmesnik večinoma temelji na eksplicitnih dejanjih. Uporabnik izbere ukaz, pritisne gumb ali vnese podatke, sistem pa izvede vnaprej določeno operacijo.

Pri sistemih z umetno inteligenco se razmerje spremeni. Uporabnik lahko izrazi **namen**, sistem pa mora sam interpretirati, kaj uporabnik želi, in oblikovati ustrezen odziv.

Klasično:

UPORABNIK → UKAZ → SISTEM → REZULTAT

Z AI:

UPORABNIK → NAMEN → INTERPRETACIJA AI → REZULTAT

S tem uporabniški vmesnik postane manj podoben nadzorni plošči in bolj prostoru za komunikacijo med človekom in sistemom.

Ključna ideja:

Pri AI-vmesniku uporabnik ne upravlja samo funkcij. Sistem mora razumeti njegov namen, uporabnik pa mora razumeti, kaj je sistem naredil z njegovo zahtevo.



Slika 14.1 – Klasični in AI-vmesnik

AI-vmesnik ni nujno pogovorni vmesnik. Umetna inteligenca lahko deluje tudi v ozadju običajnega grafičnega vmesnika, kjer uporabniku predlaga, prilagaja ali poenostavlja posamezna dejanja.

14.2 Od ukazovanja k pogovoru

Pri klasičnem sistemu mora uporabnik poznati možnosti sistema in način njihovega aktiviranja. Pri generativni AI lahko svoj cilj izrazi z naravnim jezikom:

»Razloži mi Fittsov zakon tako, da ga bo razumel študent drugega letnika računalništva.«

To močno zmanjša zahtevo po učenju sintakse. Hkrati pa nastane nov problem: naravni jezik je dvoumen.

Če uporabnik reče: »Naredi to krajše.«

Kaj pomeni *krajše*? Za polovico? Za en odstavek? Samo glavne točke?

Dober AI-vmesnik zato ne sme samo sprejeti besedila. Kadar je namen nejasen, lahko uporabniku pomaga z vprašanjem, predlogom ali prikazom možnih interpretacij.

To je pomembna sprememba HCI: **dialog lahko postane del uporabniškega vmesnika.**

Pogovorni vmesnik ima tudi slabost: uporabniku ni vedno jasno, katere funkcije sistem podpira in koliko predhodnega konteksta upošteva. Zato so koristni primeri možnih zahtev, vidna zgodovina pogovora in možnost ponovnega začetka.

14.3 Poziv (*prompt*) kot uporabniški vmesnik

Poziv je način, s katerim uporabnik AI posreduje svoj cilj, kontekst in pričakovanja. Učinkovit poziv pogosto vsebuje:

NAMEN + KONTEKST + OMEJITVE + ŽELENI FORMAT

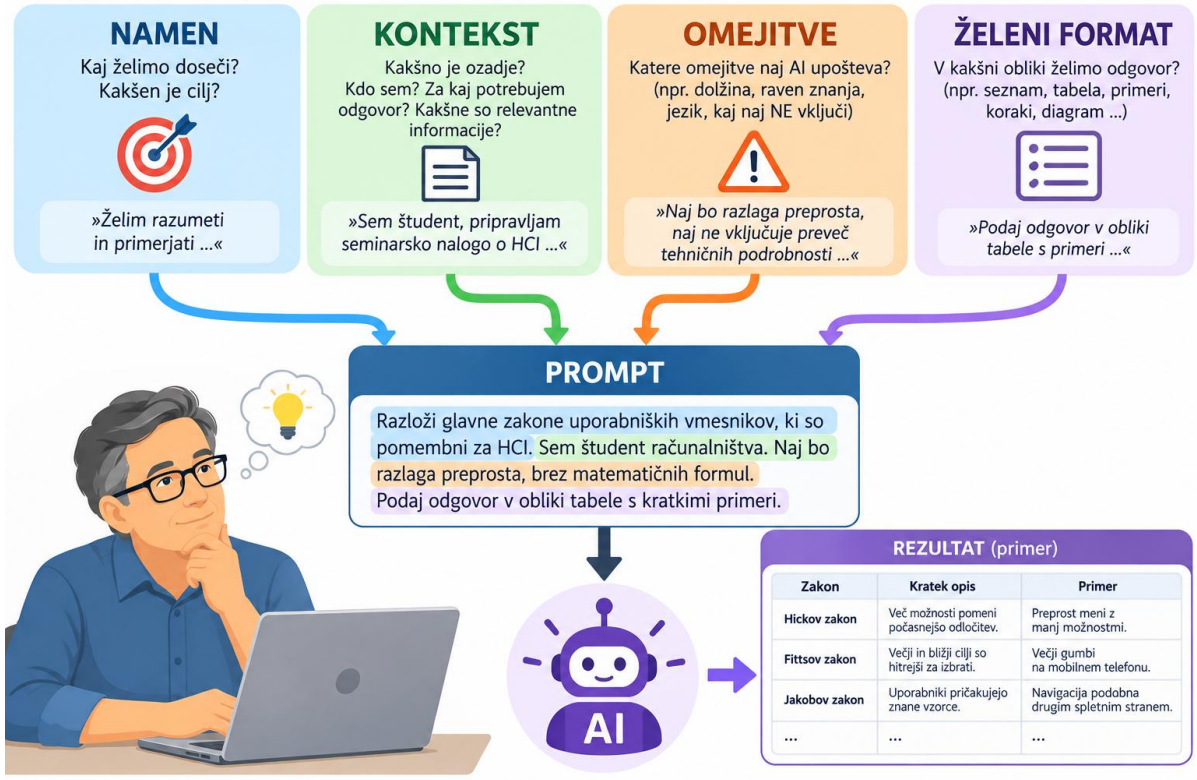
Na primer:

»Za študente drugega letnika računalništva v največ 150 besedah razloži razliko med UI in UX ter dodaj en konkreten primer.«

Vendar HCI-perspektiva opozarja na pomembno nevarnost:

Dober AI-sistem ne bi smel zahtevati, da vsak uporabnik postane strokovnjak za "prompt engineering".

Če uporabnik ne zna napisati »pravega« poziva, mu lahko sistem pomaga s predlogi, primeri, strukturiranim vnosom ali dodatnimi vprašanji.



Slika 14.2 – Sestavine učinkovitega poziva

Pogovorni vmesniki uporabnika spodbujajo, da svoje zahteve izrazi čim bolj naravno in podrobno. Pri tem lahko nehote razkrije osebne, poslovne ali druge občutljive podatke. Dober AI-vmesnik mora zato jasno sporočati, kateri podatki se obdelujejo, kako se uporabljajo in kakšen nadzor ima uporabnik nad njimi.

14.4 Negotovost in spremenljivost odgovorov AI

Od kalkulatorja pričakujemo, da bo za isti izračun vedno vrnil isti pravilen rezultat. Pri generativnem AI je položaj drugačen.

Na isto ali zelo podobno vprašanje lahko dobimo različne odgovore. Odgovor je lahko zelo prepričljiv in dobro napisan, vendar kljub temu napačen.

To ima za HCI pomembno posledico:

Tekoč in samozavesten odgovor ni isto kot zanesljiv odgovor.

Uporabniški vmesnik mora zato pomagati uporabniku razlikovati med rezultatom in njegovo zanesljivostjo. To je posebej pomembno pri nalogah, kjer ima napaka resne posledice.

14.5 Kalibrirano zaupanje

Cilj dobrega AI-vmesnika ni, da uporabnik sistemu čim bolj zaupa.

Premalo zaupanja pomeni, da uporabnik zavrača koristno pomoč. Preveč zaupanja pa lahko pomeni nekritično sprejemanje napačnih rezultatov.

Želimo **kalibrirano zaupanje**:

PREMALO ZAUPANJA ← USTREZNO ZAUPANJE → PREVEČ ZAUPANJA

Ustrezna raven zaupanja je odvisna tudi od naloge. Napaka pri predlogu naslova predstavitve ima povsem drugačne posledice kot napaka pri medicinskem, finančnem ali varnostnem priporočilu.

Ključna ideja:

Zaupanje v AI naj bo sorazmerno z zanesljivostjo sistema in posledicami morebitne napake.



Slika 14.3 – Kalibracija zaupanja

14.6 Razložljivost

Uporabnik lahko vpraša: »Zakaj predlagate prav ta rezultat?«

Razlaga mu lahko pomaga razumeti uporabljene kriterije, podatke ali razloge in presoditi, ali je rezultat smiseln.

Toda več razlage ni nujno bolje. Dolga tehnična razlaga delovanja modela uporabniku pogosto ne pomaga pri njegovi nalogi.

Z vidika HCI je zato pomembno vprašanje:

Kakšno razlago potrebuje ta uporabnik, da se lahko pravilno odloči?

Razlaga za razvijalca, zdravnika in običajnega uporabnika ni nujno enaka.

Razložljivost je posebej pomembna pri odločitvah z večjimi posledicami, kjer mora uporabnik razumeti vsaj osnovo priporočila in njegove omejitve.

Pri generativni AI moramo razlikovati med uporabniku razumljivo obrazložitvijo odgovora in dejansko razlago delovanja modela. Prepričljiva obrazložitev sama po sebi še ni dokaz pravilnosti rezultata.



Slika 14.4: Razložljivost

14.7 Človek v zanki

Pri pomembnih odločitvah pogosto uporabljamo načelo človek v zanki (human in the loop):

AI → PREDLOG → ČLOVEK → ODLOČITEV

Človek lahko predlog:

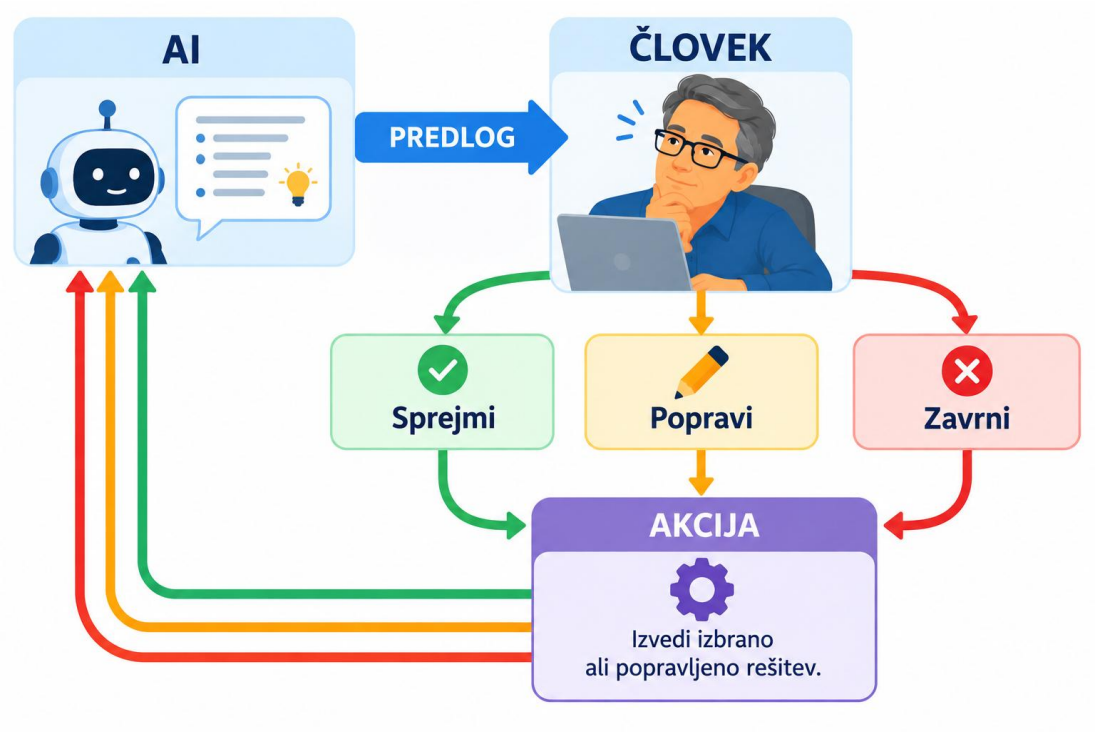
sprejme – popravi – zavrne.

Vendar človeška potrditev sama po sebi še ne pomeni resničnega nadzora.

Če sistem uporabniku predstavi rezultat zelo avtoritativno, uporabnik nima dovolj informacij za preverjanje ali pa mora pregledati več sto predlogov, lahko potrjevanje postane samo rutina.

Zato mora imeti človek:

- dovolj informacij za presojo;
- možnost rezultat spremeniti ali zavrniti;
- dovolj časa za odločitev;
- jasno predstavo o tem, za kaj je odgovoren AI in za kaj človek.



Slika 14.5 – Človek v zanki

14.8 Napake, negotovost in možnost preverjanja

Pri klasičnem vmesniku napako pogosto prepoznamo kot nedovoljeno operacijo, napačen vnos ali tehnično okvaro. Pri generativni AI je težava drugačna: rezultat je lahko videti povsem normalen, čeprav vsebuje napačne podatke, izmišljene vire ali neutemeljene sklepe.

Zato mora AI-vmesnik podpirati preverjanje.

Uporabniku lahko pomaga tako, da:

- jasno loči dejstva od predlogov ali interpretacij;
- omogoči prikaz virov, kadar obstajajo;
- opozori na pomembno negotovost;
- omogoči popravljanje rezultata;
- ohrani možnost vrnitve na prejšnjo različico;
- pri pomembnih odločitvah spodbuja preverjanje.

Primer

AI napiše:

»Raziskava Smitha iz leta 2024 je pokazala ...«

Uporabnik ne bi smel sklepati, da raziskava obstaja samo zato, ker je navedba videti verodostojna. Če je vir pomemben za nalogo, ga mora biti mogoče preveriti.

To lepo poveže AI s prejšnjimi poglavji o povratni informaciji, napakah in obnovitvi.

14.9 AI kot partner

AI lahko razumemo kot orodje, vendar je pri nekaterih nalogah koristnejši model sodelovanja človek ⇄ AI.

Človek prispeva predvsem:

cilj – kontekst – vrednote – presojo – odgovornost

AI pa lahko prispeva:

hitrost – generiranje možnosti – iskanje vzorcev – obdelavo velike količine informacij

Dober sistem zato ne poskuša nujno odstraniti človeka iz procesa. Lahko razdeli delo tako, da vsak opravlja tisto, pri čemer je primernejši.

Primer je pisanje besedila. AI lahko hitro pripravi več osnutkov, človek pa presodi, kateri ustreza namenu, popravi napake in prevzame odgovornost za končno besedilo.

Pri evalvaciji AI-vmesnika ne preverjamo samo, ali uporabnik uspešno opravi nalogo, temveč tudi, ali pravilno razume možnosti in omejitve sistema. Pomembno je na primer preveriti, ali uporabnik prepozna negotov odgovor, zna zahtevati popravek in ne sprejema rezultatov nekritično. Ker lahko AI na isto zahtevo odgovori različno, je smiselno nekatere naloge preizkusiti večkrat.

14.10 Povzetek

Umetna inteligenca spreminja HCI, ker uporabnik sistemu ne posreduje več samo ukazov, ampak tudi namen, sistem pa ta namen interpretira.

S tem se pojavijo novi problemi: dvoumnost naravnega jezika, spremenljivi in negotovi rezultati, napačno kalibrirano zaupanje ter potreba po razlagi in preverjanju.

Dober AI-vmesnik zato uporabniku ne omogoča samo, da nekaj vpraša. Pomaga mu izraziti namen, razumeti rezultat, presoditi njegovo zanesljivost, ga popraviti ali zavrniti ter ohraniti nadzor nad končno odločitvijo.

Vaje:

Preizkusite generativni AI pri nalogi s področja, ki ga dobro poznate.

- *Napišite začetni poziv in shranite odgovor.*
- *Poiščite vsaj eno trditev, ki zahteva preverjanje.*
- *Spremenite poziv tako, da dodate kontekst, omejitve in želeni format.*
- *Zahtevajte pojasnilo enega dela odgovora.*
- *Eno pomembno trditev preverite iz neodvisnega vira.*
- *Presodite, pri kakšni uporabi bi bilo nevarno rezultat sprejeti brez preverjanja.*

Izberite pogovorni AI-sistem in analizirajte njegov uporabniški vmesnik. Preverite:

- Ali uporabnik razume, kaj sistem zmore in česa ne?
- Kako sistem pokaže negotovost?
- Ali je mogoče odgovor preprosto popraviti, zavrniti ali ponovno ustvariti?
- Ali uporabnik razume, kateri podatki se uporabljajo?

Predlagajte tri izboljšave vmesnika z vidika HCI.

15. Proaktivni in agentski sistemi

15.1 Od odzivnega k proaktivnemu sistemu

Klasični interaktivni sistem praviloma čaka na uporabnika:

UPORABNIK → UKAZ → SISTEM → ODZIV

Proaktivni sistem pa lahko na podlagi konteksta sam presodi, da bi bilo koristno nekaj predlagati ali narediti:

KONTEKST → SISTEM → PREDLOG ALI AKCIJA → UPORABNIK

Na primer, koledar ne čaka, da uporabnik preveri naslednji sestanek. Upošteva čas, lokacijo in promet ter opozori:

»Sestanek se začne čez 40 minut. Zaradi gostega prometa bi morali oditi čez 10 minut.«

Sistem je s tem prevzel **pobudo**.

To je pomembna sprememba v HCI. Pri klasičnem vmesniku se predvsem sprašujemo, ali sistem pravilno reagira na uporabnikovo dejanje. Pri proaktivnem sistemu pa se moramo vprašati tudi:

Ali je sistem sploh ravnal ob pravem času in na primeren način?

Proaktivnost torej ni samo tehnična sposobnost predvidevanja. Je tudi oblikovalska odločitev o tem, kdaj naj sistem poseže v uporabnikovo dejavnost. To je eno osrednjih vprašanj sodobnega raziskovanja proaktivnih sistemov.



Slika 15.1 – Od odzivnega k proaktivnemu

Proaktivnost in agentnost nista sopomenki. Proaktiven sistem sam prevzame pobudo, agent pa lahko poleg tega samostojno načrtuje in izvaja zaporedja dejanj za dosego določenega cilja. Proaktiven sistem zato ni nujno agent, agent pa ni nujno vedno proaktiven.

15.2 Kaj je agent?

Agent je sistem, ki lahko v določeni meri:

sprejme cilj → načrtuje → izvede akcijo → opazuje rezultat → presodi → prilagodi načrt

Če uporabnik reče:

»Organiziraj mi potovanje na konferenco v Pariz.«

agent lahko nalogo razdeli na podnaloge: poišče možne lete, preveri termine konference, poišče hotel, primerja možnosti in pripravi predlog.

Bistvena razlika glede na običajni pogovorni sistem je v tem, da agent ne ustvarja samo odgovora, ampak lahko načrtuje več korakov in deluje v okolju. Sodobni agentski sistemi so zato opisani kot ciljno usmerjeni sistemi, ki lahko načrtujejo, uporabljajo orodja ter izvajajo zaporedja akcij.



Slika 15.2 – Agentska zanka

Agent lahko pri izvajanju naloge uporablja tudi zunanja orodja, na primer koledar, spletni iskalnik, elektronsko pošto ali rezervacijski sistem. S tem ne ustvarja več samo informacij, temveč lahko neposredno spreminja stanje drugih sistemov.

15.3 Problem avtonomije

Ko sistem samo predlaga hotel, ima uporabnik še vedno popoln nadzor nad rezervacijo. Če ga agent sam rezervira, je prevzel del uporabnikovega nadzora.

Zato vprašanje ni samo:

»Kaj agent zna narediti?«

ampak predvsem:

»Kaj mu dovolimo narediti brez uporabnikove potrditve?«

Predstavljajmo si agenta, ki upravlja elektronsko pošto.

Lahko:

- razvrsti sporočila;
- pripravi osnutek odgovora;
- pošlje odgovor;
- izbriše sporočilo.

Te akcije se razlikujejo po posledicah. Napačna razvrstitev je večinoma popravljiva. Napačno poslano zaupno sporočilo pa morda ni.

Raziskave HCI zato pri agentih posebej poudarjajo **ravnotežje med avtonomijo sistema in agency oziroma nadzorom uporabnika.**

15.4 Stopnje avtonomije

Avtonomije ne smemo razumeti samo kot izbiro med:

človek naredi vse ↔ agent naredi vse

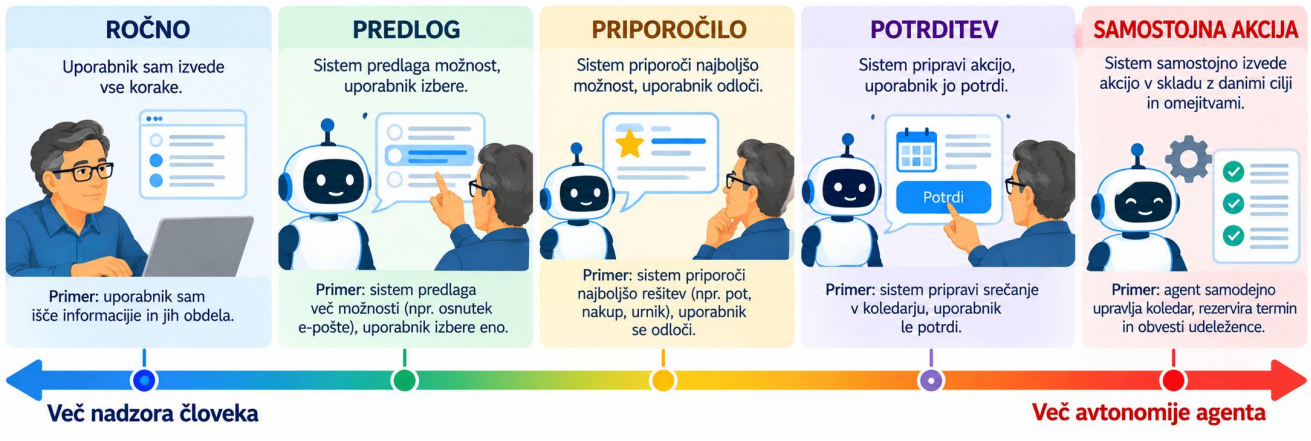
Med skrajnostma obstaja več stopenj:

ROČNO → PREDLOG IZBIRE → PRIPOROČILO → POTRDITEV → SAMOSTOJNA AKCIJA

Na primer pri rezervaciji hotela:

Ročno: uporabnik sam poišče in rezervira hotel.
Predlog izbire: agent pokaže nekaj možnosti.
Priporočilo: agent sam izbere možnost, ki se mu zdi najboljša.
Potrditev: agent pripravi rezervacijo, uporabnik jo potrdi.
Samostojna akcija: agent rezervacijo izvede sam.

Prava stopnja avtonomije je odvisna od naloge, posledic napake, zaupanja, možnosti razveljavitve in uporabnikovih želja. Več avtonomije torej ni samodejno boljša uporabniška izkušnja.



Slika 15.3 – Kontinuum avtonomije

15.5 Reverzibilnost

Pomemben kriterij pri določanju avtonomije je vprašanje:

Kaj se zgodi, če se agent zmoti?

Če agent prestavi datoteko v napačno mapo, lahko dejanje enostavno razveljavimo. Če pošlje sporočilo napačnemu naslovniku, je popravek težji. Če opravi finančno transakcijo, so lahko posledice še resnejše.

Zato velja uporabno načelo:

Bolj ko je dejanje tvegano in težko popravljivo, več uporabnikovega nadzora potrebujemo pred izvedbo.

Pri načrtovanju agentskega vmesnika zato preverimo:

posledice napake + reverzibilnost + stroške napake + možnost pregleda pred izvedbo.

15.6 Obveščanje in prekinjanje

Proaktivni sistem mora odločati ne samo kaj uporabniku sporočiti, ampak tudi kdaj.

Obvestilo:

»Čez pet minut bo začelo deževati.«

je lahko koristno človeku, ki se pripravlja na odhod, in nepotrebno človeku, ki predstavlja projekt pred občinstvom.

Zato mora sistem upoštevati vsaj:

pomembnost + nujnost + kontekst uporabnika.

To navezuje poglavje neposredno na 13. poglavje, kjer smo obravnavali prekinitve in pozornost. Pri agentih pa je vprašanje še pomembnejše, saj sistem sam odloča, kdaj bo prevzel pobudo. Raziskave proaktivnih agentov prav primernost časa in nevsiljivost izpostavljajo kot pomembna dejavnika uporabniške izkušnje.

Uporabnik mora imeti možnost določiti, kdaj in pri katerih vrstah dogodkov sme sistem sam prevzeti pobudo.



Slika 15.4 – Odločanje o prekinitvi

15.7 Nadzor nad agentom

Pri klasičnem programu uporabnik večinoma ve, kaj je pravkar naredil: kliknil je gumb in sistem se je odzval.

Agent pa lahko izvaja več korakov v ozadju. Zato mora uporabniški vmesnik omogočiti odgovor na vprašanja:

- Kaj agent trenutno počne?
- Zakaj to počne?
- Kaj namerava narediti naslednje?
- Kaj je že naredil?
- Katere akcije čakajo na mojo potrditev?
- Kaj lahko ustavim?
- Kaj lahko razveljavim?

To je pravzaprav staro HCI-načelo vidnosti stanja sistema, vendar postane pri agentu zahtevnejše.

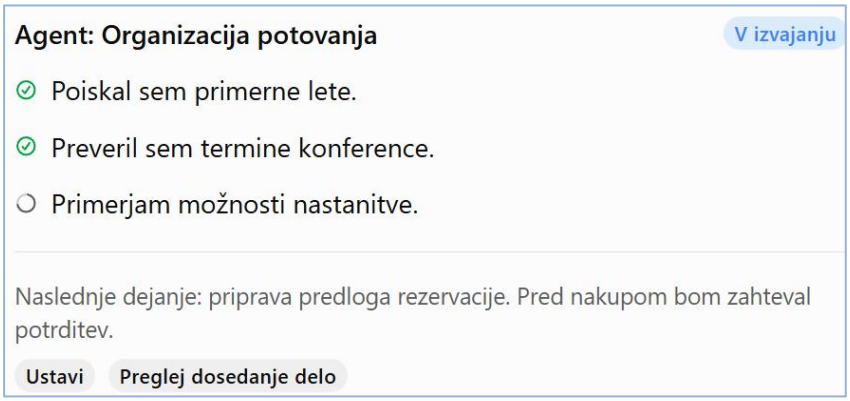
Ni dovolj prikazati:

»Agent dela ...«

Koristneje je na primer:

»Primerjam tri možnosti leta. Rezervacije še nisem izvedel.«

Primer prikaza stanja agenta:



Slika 15.5: Konceptualni primer agentskega uporabniškega vmesnika.

Uporabnik tako pozna **stanje, namen in mejo agentovega trenutnega delovanja**. Tudi novejše raziskave uporabniške izkušnje pri računalniških agentih med ključnimi elementi posebej izpostavljajo uporabnikov nadzor, razložljivost in uporabnikov miselni model delovanja agenta.

Pomemben del nadzora so tudi dovoljenja. Uporabnik mora razumeti, do katerih podatkov in zunanjih storitev ima agent dostop ter katere akcije sme izvajati. Smiselno je načelo najmanjših potrebnih dovoljenj: agent naj ima na voljo samo tiste pravice, ki jih potrebuje za trenutno nalogo. Uporabnik mora imeti tudi možnost dovoljenja pozneje spremeniti ali preklicati.

15.8 Kdaj mora agent vprašati uporabnika?

Agent ne bo vedno vedel, kaj uporabnik želi. Lahko naleti na nejasen cilj, manjkajoč podatek ali več enakovrednih možnosti.

Predstavljajmo si zahtevo:

»Rezerviraj mi dober hotel blizu konference.«

Kaj pomeni *dober*? Kako *blizu*? Kolikšna *cena* je sprejemljiva?

Slab agent lahko ugiba in ukrepa.

Boljša možnost je:

»Našel sem tri hotele v bližini. Pred rezervacijo potrebujem še vaš najvišji sprejemljivi znesek na noč.«

Pomembna lastnost dobrega agenta je torej tudi sposobnost, da ve, kdaj ne sme samostojno nadaljevati.

Pri negotovosti lahko:

vpraša → predlaga možnosti → zahteva potrditev → ali se ustavi.

Pri odločanju o prehodu na človeški nadzor so pomembni prav negotovost, dvoumnost uporabnikovega namena, posledice in reverzibilnost dejanja.

15.9 Povzetek

Proaktivni sistemi ne čakajo vedno na uporabnikov ukaz, agentski sistemi pa lahko poleg predlaganja tudi načrtujejo in izvajajo zaporedja dejanj.

S tem se temeljno vprašanje HCI spremeni. Ne sprašujemo se več samo:

Kako naj uporabnik upravlja sistem?

ampak tudi:

Koliko sme sistem narediti namesto uporabnika?

Dober agentski vmesnik zato išče ravnotežje med koristno avtonomijo in človeškim nadzorom. Uporabniku mora omogočiti, da razume, kaj agent počne, določi meje njegovega delovanja, poseže v proces ter ustavi ali razveljavi dejanja, kadar je to mogoče.

Vaje:

1. *Predstavljajte si osebnega AI-agenta, ki upravlja vaš elektronski koledar. Razvrstite naslednje akcije v tri skupine:*

   A – agent lahko izvede sam

   B – agent mora pred izvedbo vprašati

   C – agent sme ukrepati samo na neposredni ukaz

   Primeri akcij:

   - opozori na sestanek;
   - predlaga nov termin;
   - prestavi sestanek;
   - zavrne povabilo;
   - povabi novo osebo;
   - izbriše sestanek;
   - rezervira prostor;
   - drugim udeležencem pošlje sporočilo.

   Za vsako odločitev odgovorite: "Kaj bi se zgodilo, če bi se agent zmotil?" *To vprašanje je pomembnejše od same razvrstitve.*

2. *Ali bi se vaša odločitev spremenila, če bi agent deloval v zasebnem koledarju namesto v službenem koledarju, ki ga uporablja več ljudi? Utemeljite odgovor.*

16. XR, VR in prostorski uporabniški vmesniki

16.1 Od zaslona k prostoru

Pri klasičnem grafičnem uporabniškem vmesniku je prostor interakcije predvsem dvodimenzionalen zaslon. Uporabnik izbira gumbe, menije, ikone in druge elemente, ki imajo položaj na površini.

Pri prostorskem vmesniku postane del uporabniškega vmesnika **tridimenzionalni prostor okoli uporabnika**.

XR (Extended Reality) je skupni izraz za tehnologije, ki na različne načine povezujejo fizični in digitalni svet. Običajno razlikujemo:

VR – Navidezna resničnost (virtual reality): uporabnik je potopljen predvsem v virtualno okolje.
AR – obogatena resničnost (augmented reality): digitalne informacije so dodane pogledu na fizični svet.
MR – mešana resničnost (mixed reality): digitalni objekti so prostorsko povezani s fizičnim okoljem in lahko postanejo del uporabnikove interakcije z njim.

Za HCI niso najpomembnejše oznake tehnologij, temveč vprašanje:

Kako človek zaznava prostor in kako v njem izbira, premika ter upravlja digitalne objekte?

16.2 Kontinuum fizičnega in virtualnega

Prehod med fizičnim in popolnoma virtualnim okoljem lahko razumemo kot kontinuum:

FIZIČNI SVET → AR → MR → VR → VIRTUALNI SVET

Meje med posameznimi kategorijami niso vedno ostre. Sistem lahko na primer prikazuje fizično okolje s kamerami in vanj vstavi virtualne predmete.



Slika 16.1 – Kontinuum od fizičnega do povsem virtualnega

Za HCI je zato koristneje vprašati:

- koliko fizičnega okolja uporabnik še zaznava;
- kateri elementi so digitalni;
- s katerimi lahko uporabnik interagira;
- kako se digitalni in fizični objekti med seboj prostorsko povezujejo.

16.3 Prostorska interakcija

Pri običajnem GUI je objekt praviloma določen s položajem na zaslonu:

(x, y).

V prostorskem okolju ima položaj:

(x, y, z),

poleg tega pa še orientacijo, oddaljenost in odnos do telesa uporabnika.

Tudi načini interakcije se spremenijo. Uporabljamo lahko:

pogled – gibanje glave – roke – kontrolerje – govor – položaj telesa.

Pri izbiri gumba na zaslonu uporabnik premakne kazalec in klikne. V VR lahko predmet pogleda, nanj pokaže s kontrolerjem, ga prime z virtualno roko ali uporabi neposredno gesto.

S tem se pojavi klasično vprašanje HCI v novi obliki:

Kako uporabniku pokažemo, kaj lahko v prostoru naredi?

Tudi v XR torej potrebujemo dobre zaznavne znake (signifiers) in jasno povratno informacijo.

V prostorskem vmesniku mora uporabnik poleg tega razumeti, kje je objekt glede na njegovo telo, kako daleč je in ali je dosegljiv oziroma ga je mogoče izbrati.

16.4 Dosegljivost in ergonomija

V običajnem grafičnem vmesniku se pogosto ukvarjamo z velikostjo in razporeditvijo elementov. V XR moramo razmišljati tudi o človeškem telesu.

Pomembni postanejo:

doseg rok – višina uporabnika – položaj glave – vidno polje – oddaljenost objekta – udoben kot gledanja.

Predstavljajmo si virtualni meni, postavljen visoko nad uporabnikovo glavo. Uporabnik ga sicer lahko vidi, toda pogosto dvigovanje glave postane neprijetno.

Podobno velja za element, ki zahteva ponavljajoče seganje daleč pred telo.

Tehnološko dosegljivo še ne pomeni ergonomsko primerno.

16.5 Utrujenost pri gestah

Filmi pogosto prikazujejo prostorske vmesnike, pri katerih človek dolgo maha z rokami po zraku. Videti je privlačno, vendar človeško telo za tak način dela ni posebej primerno.

Dolgotrajno držanje rok pred telesom povzroča utrujenost. Zato mora oblikovalec razmisliti, ali je velika gesta res potrebna. Utrujenost zaradi dolgotrajnega držanja rok v zraku je znana tudi pod izrazom *gorilla arm*.

Pogosto je boljša:

majhna gesta namesto velike,
kratek gib namesto dolgega,
počitek roke namesto stalnega držanja v zraku.

To je dober primer splošnega načela HCI: Naravna interakcija ni nujno tudi udobna interakcija.

16.6 Premikanje in kibernetska slabost

Vestibularni sistem v notranjem ušesu sodeluje pri zaznavanju gibanja in ravnotežja.

V VR lahko nastane nenavadna situacija: oči uporabniku sporočajo, da se premika skozi prostor, vestibularni sistem pa ne zazna ustreznega fizičnega gibanja. Takšno neskladje lahko prispeva k nelagodju, slabosti ali dezorientaciji, ki jih pogosto označujemo kot **kibernetsko slabost** (cybersickness).

Zato je način premikanja po virtualnem prostoru tudi problem HCI.

Uporabniku mora biti vedno omogočeno, da premikanje ustavi, izbere udobnejši način navigacije ali izkušnjo prekine. Pri prvih uporabah VR je smiselno začeti s kratkimi, manj intenzivnimi izkušnjami.

Uporabljamo lahko fizično hojo, teleportiranje ali nadzorovano virtualno premikanje. Različni načini imajo različne prednosti in slabosti glede občutka prisotnosti, orientacije in udobja.

Pomembno načelo je:

Če sistem pri delu povzroča telesno nelagodje, to ni samo problem uporabnika, ampak tudi problem uporabniškega vmesnika.

16.7 Prostorski infozabavni sistem v avtomobilu

Predstavljajmo si potnika v avtomobilu z očali XR. Med vožnjo mimo Blejskega jezera se lahko nad resničnim pogledom prikažejo informacije o Blejskem gradu, zgodovini kraja ali možnostih za obisk.

Sistem mora pri tem poznati položaj vozila in smer pogleda ter digitalno vsebino pravilno povezati z okolico.

Toda z vidika HCI so pomembnejša druga vprašanja:

- Ali vsebina ovira pogled?
- Ali preveč obremenjuje pozornost?
- Ali ostaja stabilna med premikanjem vozila?
- Ali povzroča slabost?
- Ali lahko potnik prikaz enostavno izključi?



Slika 16.2 – Koncept XR-infozabavnega sistema v vozilu

Primer tako združuje kontekst, multimodalnost, ergonomijo, AI, pozornost in prostorsko interakcijo.

Takšna uporaba je namenjena predvsem potnikom. Pri vozniku bi lahko dodatne prostorske informacije predstavljale nevarno odvračanje pozornosti od prometnega okolja.

16.8 Socialni HCI v VR

V virtualnem okolju lahko sodeluje več ljudi, ki jih predstavljajo avatarji.

S tem postane vmesnik tudi posrednik med ljudmi:

ČLOVEK ↔ RAČUNALNIŠKO OKOLJE ↔ ČLOVEK

Pojavijo se vprašanja, ki jih pri klasičnem GUI skoraj ni:

Kaj pomeni osebni prostor avatarja? Kako blizu se mu lahko približamo? Kam gledamo med pogovorom? Kako prikažemo gesto, izraz obraza ali dotik?

Poleg tega se pojavijo vprašanja identitete, zasebnosti in neželenega vedenja.

Prostorski HCI zato ni samo vprašanje, kako upravljamo virtualne predmete, temveč tudi, **kako prek virtualnega prostora komuniciramo z drugimi ljudmi.**

Uporabnik mora imeti možnost nadzorovati svoj virtualni osebni prostor, na primer z omejitvijo približevanja drugih avatarjev, utišanjem ali blokiranjem neželenih udeležencev.

16.9 Povzetek

XR spremeni prostor interakcije. Vmesnik ni več omejen na pravokotno površino zaslona, temveč lahko vključuje prostor okoli uporabnika in njegovo telo.

S tem dobijo klasična vprašanja HCI nove oblike. Zaznavni znaki morajo pokazati možnosti interakcije v prostoru, povratna informacija mora potrditi prostorska dejanja, ergonomija mora upoštevati telo, načini premikanja pa tudi fiziološko udobje.

Osrednje vprašanje zato ni:

> Kaj lahko v XR tehnično naredimo?

temveč:

> Kaj lahko človek v takem okolju učinkovito, razumljivo, varno in udobno uporablja?

Vaja z očali VR

Delajte v parih. Eden naj uporablja izbrano VR-aplikacijo, drugi pa naj opazuje njegovo interakcijo, ne da bi mu pri tem dajal navodila. Pred začetkom zagotovite dovolj velik in varen prostor brez ovir. Če uporabnik občuti slabost, vrtoglavico ali drugo nelagodje, uporabo takoj prekinite. Opazovalec skrbi tudi za fizično varnost uporabnika:

Zabeležite:

- *kako uporabnik odkrije, kaj lahko naredi;*
- *kako izbira in prijema objekte;*
- *ali razume povratno informacijo;*
- *katere gibe ponavlja;*
- *kdaj postane negotov;*
- *ali se pojavi fizična utrujenost ali nelagodje.*

Nato zamenjajte vlogi.

Na koncu izberite tri opažene probleme in za vsakega predlagajte izboljšavo uporabniškega vmesnika.

17. Prihodnost HCI in sinteza

17.1 Od računalnika k interakcijskemu okolju

Razvoj HCI lahko razumemo tudi kot postopno spreminjanje mesta računalnika v človekovem okolju.

Nekoč je moral uporabnik priti do računalnika in se naučiti njegovega jezika. Z grafičnimi vmesniki je interakcija postala bolj neposredna. Mobilne naprave so računalnik prenesle v vsakdanje življenje, pametne naprave v okolje, AI pa omogoča, da sistem vedno bolj interpretira uporabnikov namen in kontekst.

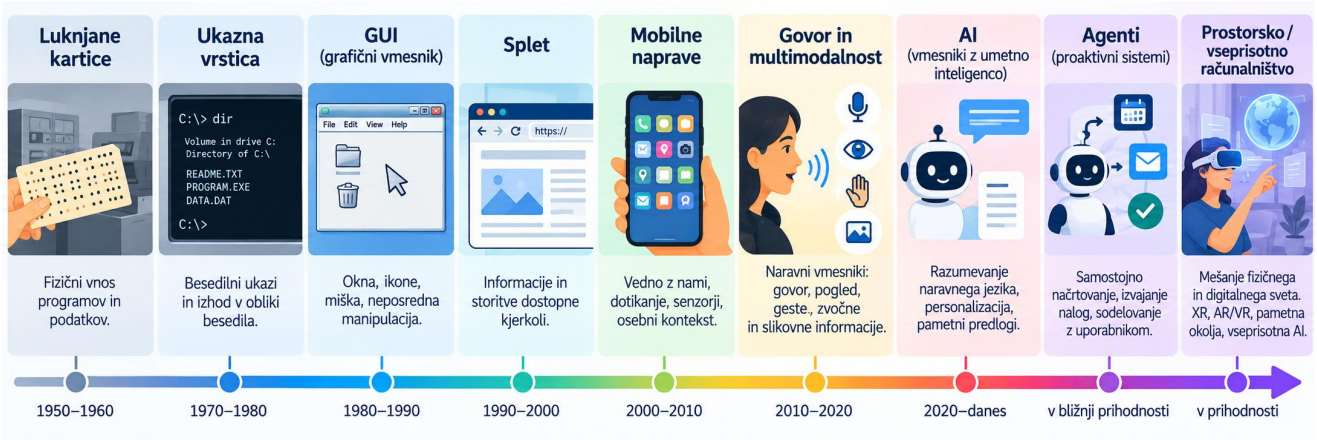
UKAZNA VRSTICA → GUI → SPLET → MOBILNO → MULTIMODALNO → AI/AGENTI → PROSTORSKO IN AMBIENTNO

Paradoks razvoja je zanimiv:

Računalniški sistemi postajajo vse bolj kompleksni, uporabnik pa pričakuje vse preprostejšo interakcijo.

Zato prihodnost HCI ni samo vprašanje novih tehnologij. Je predvsem vprašanje, kako naraščajočo kompleksnost sistema skriti, ne da bi pri tem skrili njegovo delovanje in uporabniku odvzeli nadzor.

Nove oblike interakcije praviloma ne nadomestijo povsem prejšnjih, temveč razširjajo možnosti, med katerimi izbiramo glede na uporabnika, nalogo in okoliščine uporabe.



Slika 17.1 – Razvoj oblik interakcije

17.2 Izginjanje uporabniškega vmesnika

Najboljši vmesnik prihodnosti morda ne bo zaslon.

Vrata se odklenejo, ko prepoznajo pooblaščenega uporabnika. Luči se prilagodijo prostoru. Avtomobil predlaga primerno pot. AI pripravi informacije, še preden jih uporabnik zahteva.

Vmesnik postaja ambienten – vgrajen v okolje in manj opazen.

Toda izginjanje vidnega vmesnika povzroči nov problem. Pri gumbu vidimo, kaj lahko pritisnemo. Pri meniju vidimo možnosti. Kako pa uporabnik ve, kaj lahko naredi v sistemu, ki nima vidnega vmesnika?

In še pomembneje:

Če vmesnik izgine, ne sme z njim izginiti tudi uporabnikov občutek nadzora.

Uporabniku mora biti še vedno jasno:

kaj se je zgodilo – zakaj se je zgodilo – kdo ali kaj je odločilo – kako lahko odločitev spremeni.

To lepo povezuje prihodnost HCI z našimi starimi znanci: zaznavnimi znaki, povratno informacijo, vidnostjo stanja sistema in nadzorom.

17.3 Od uporabniškega vmesnika k odnosu

Pri klasičnem računalniku je veliko interakcij kratkotrajnih:

ukaz → rezultat.

Z AI-agentom pa je lahko odnos dolgotrajen. Sistem se lahko prilagaja uporabniku, si ohranja kontekst in sodeluje z njim pri različnih nalogah.

Zato se pojavi novo vprašanje:

Kakšen odnos naj ima človek s sistemom, ki ga uporablja mesece ali leta?

Pri takem sistemu postanejo zaupanje, zasebnost, transparentnost in nadzor enako pomembni kot velikost gumba ali struktura menija.

Uporabniška izkušnja tako ni več samo izkušnja enega opravila, temveč se oblikuje skozi daljše obdobje uporabe.

17.4 Poosebljenje in prilagajanje

Sistem se lahko prilagaja uporabnikovemu znanju, sposobnostim, napravi, okolju, prejšnjim dejanjem in trenutni nalogi.

Takšno poosebljenje (personalization) je lahko koristno. Začetniku lahko sistem ponudi več pomoči, izkušenemu uporabniku hitrejše poti.

Toda prilagajanje lahko postane tudi problematično, če uporabnik ne ve, zakaj vidi drugačno vsebino ali zakaj se sistem obnaša drugače kot prej.

Dobro prilagajanje mora zato biti:

razumljivo – nadzorovano – po možnosti reverzibilno.

Pomembno je tudi, da sistem uporabnika zaradi preteklega vedenja ne zapre v vedno isti način uporabe. Uporabnik se uči, spreminja in lahko želi nekaj drugega kot včeraj.

17.5 HCI in etika

Ko sistemi samo izvajajo uporabnikove ukaze, se veliko vprašanj nanaša predvsem na uporabljivost.

Ko pa začnejo priporočati, prilagajati, napovedovati in samostojno ukrepati, se pojavijo tudi etična vprašanja.

Pri načrtovanju se moramo vprašati:

- Ali sistem uporabniku pomaga pri odločitvi ali ga skuša vanjo usmeriti?
- Ali uporabnik razume, zakaj je nekaj priporočeno?
- Ali lahko priporočilo zavrne?
- Katere podatke sistem zbira in zakaj?
- Ali so različne skupine uporabnikov obravnavane ustrezno?
- Kdo nosi odgovornost, če sistem naredi napako?

Temni vzorci so načini oblikovanja uporabniških vmesnikov, ki uporabnika zavajajo ali usmerjajo k odločitvam, ki niso nujno v njegovem interesu. Primer je velika in poudarjena možnost »Sprejmi vse« ob skoraj skriti možnosti za zavrnitev nepotrebnega zbiranja podatkov. Etično oblikovanje mora uporabniku omogočiti razumljivo, prostovoljno in enakovredno izbiro.

Na primer: velika barvita možnost »Sprejmi vse« in skoraj skrita možnost za zavrnitev nepotrebnega zbiranja podatkov.

17.6 Človek naj ostane v središču

Pri vsaki novi tehnologiji se lahko vprašamo:

Kaj ta tehnologija pomeni za človeka, ki jo bo uporabljal?

Nova rešitev je lahko tehnološko izjemna, vendar je to še ne naredi dobre z vidika HCI.

Sistem je uspešen šele takrat, ko ljudem omogoča učinkoviteje, varneje, razumljiveje ali prijetneje dosegati njihove cilje.

Zato pri HCI ne izhajamo iz vprašanja:

»Kaj omogoča nova tehnologija?«

temveč:

»Kateri človekov problem rešujemo in ali ga nova tehnologija res rešuje bolje?«

S tem se pravzaprav vrnemo na začetek učbenika.

17.7 Sinteza učbenika

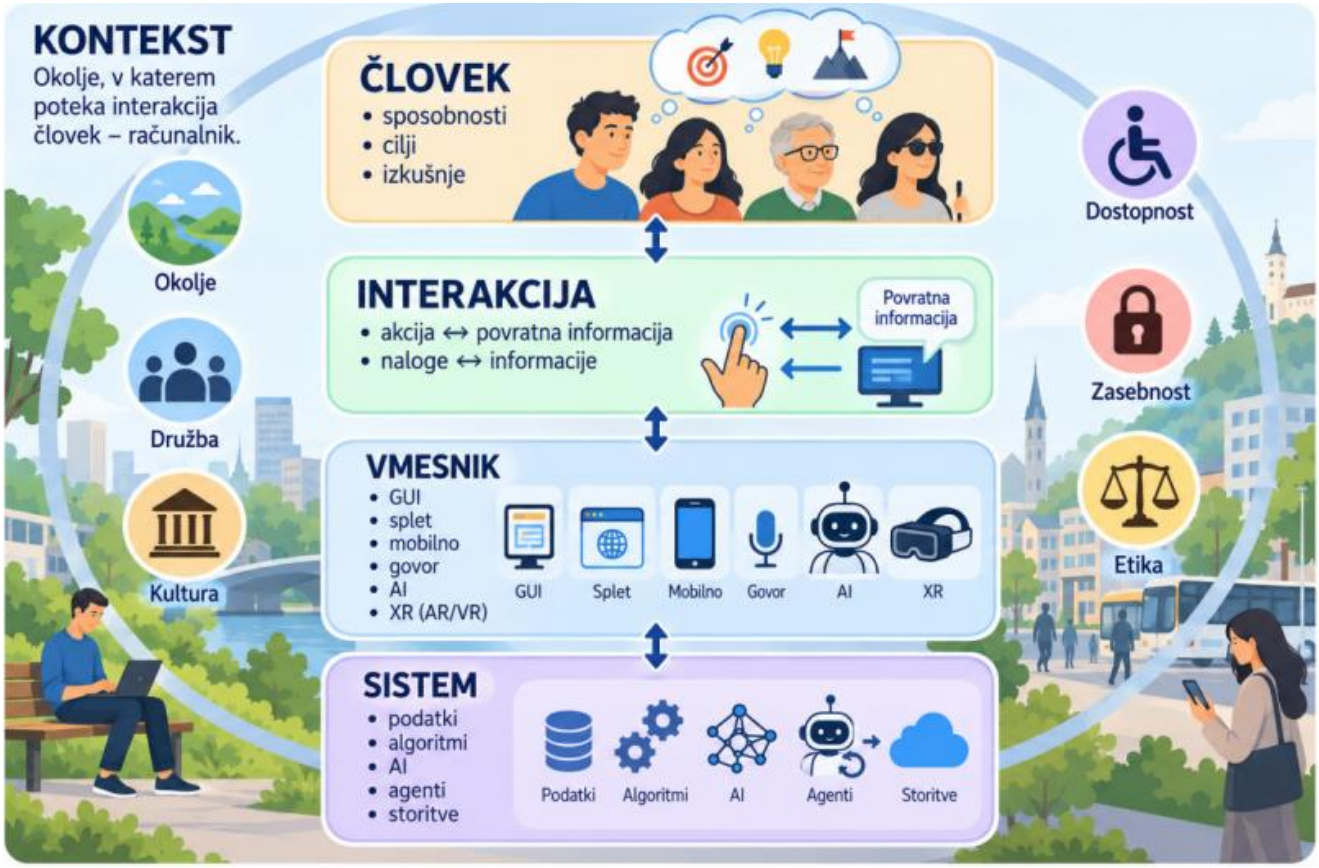
HCI ni samo oblikovanje zaslonov.

Je področje, ki povezuje:

ČLOVEKA – NALOGO – TEHNOLOGIJO – KONTEKST

in preučuje interakcijo med njimi.

Komunikacijo človek–računalnik lahko zato opredelimo kot interdisciplinarno področje, ki preučuje, načrtuje in vrednoti interakcijo med ljudmi in računalniškimi sistemi z namenom ustvarjanja sistemov, ki so uporabni, razumljivi, učinkoviti, dostopni, varni in primerni za človekove potrebe ter kontekst uporabe.



Slika 17.2 – HCI kot sistem

V različnih obdobjih se spreminja tehnologija, vendar ostajajo temeljna vprašanja presenetljivo podobna:

- Kdo je uporabnik?
- Kaj želi doseči?
- Kaj razume?
- Kaj lahko naredi?
- Kako ve, kaj se je zgodilo?
- Kaj se zgodi, če naredi napako?
- Ali lahko ohrani nadzor?
- Ali mu sistem dejansko pomaga?

Miška lahko izgine. Zaslon lahko izgine. Tipkovnico lahko nadomesti govor. Klasično aplikacijo lahko nadomesti AI-agent.

Ena stvar pa ostaja: človek.

Zato temeljna načela HCI z razvojem umetne inteligence, agentskih sistemov in prostorskih vmesnikov ne postajajo manj pomembna, temveč še pomembnejša.

Zaključna vaja: HCI skozi celoten učbenik

Izberite poljubno aplikacijo ali računalniški sistem, ki ga dobro poznate. Analizirajte ga z vidika naslednjih vprašanj:

- *Kdo so njegovi uporabniki in katere naloge želijo opraviti?*
- *Kakšen miselni model sistema si lahko ustvari uporabnik?*
- *Kako sistem zagotavlja povratno informacijo in obravnava napake?*
- *Ali je dostopen različnim skupinam uporabnikov?*
- *Kako bi preverili njegovo uporabnost?*
- *Ali bi lahko uporaba AI, multimodalnosti ali drugih sodobnih tehnologij izboljšala uporabniško izkušnjo?*
- *Predlagajte tri izboljšave in jih utemeljite z načeli HCI.*

Dodatek A: Orodja za načrtovanje in prototipiranje uporabniških vmesnikov

Praktični pregled • stanje september 2026

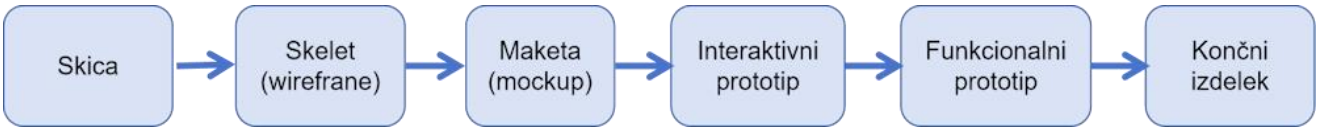
A.1 Zakaj so orodja v dodatku

Načela uporabniško usmerjenega načrtovanja, modeliranja uporabnikov in prototipiranja se spreminjajo precej počasneje kot programska orodja. Posamezni izdelki dobivajo nove funkcije, spreminjajo cenovne modele, se združujejo z drugimi izdelki ali izginejo. Zato je smiselno konkretna orodja obravnavati v dodatku, ki ga lahko posodobimo, ne da bi spreminjali temeljno besedilo učbenika.

Ta dodatek ni katalog vseh razpoložljivih programov. Izbrani so predstavniki različnih pristopov: od papirnate skice do namenskega wireframinga, vizualnega načrtovanja, bogatega interaktivnega prototipiranja in low-code/no-code razvoja funkcionalnih prototipov.

Temeljno pravilo
Najprej določimo, kaj želimo izvedeti ali preveriti, šele nato izberemo orodje. Najzmogljivejše orodje ni nujno najboljše za zgodnjo fazo načrtovanja.

A.2 Orodje izbiramo glede na razvojno stopnjo



Meje niso ostre. *Figma* ali *Penpot* lahko na primer uporabljamo že za skelete in nato nadaljujemo do zelo podrobnega interaktivnega prototipa. *FlutterFlow* lahko uporabimo za vizualno zasnovo, nato pa iz istega projekta razvijemo dejansko delujočo aplikacijo. Zato spodnja razvrstitev pomeni tipično, ne izključno uporabo.

A.3 Najpreprostejša orodja: papir, svinčnik in bela tabla

Papir in svinčnik sta še vedno resni orodji za načrtovanje uporabniških vmesnikov. Njuna glavna prednost je skoraj ničelna cena spremembe. V nekaj minutah lahko izdelamo več različic, jih primerjamo, prečrtamo in začnemo znova. Papirnati prototip lahko postane celo interaktiven: moderator menja narisane zaslone glede na uporabnikove dotike.

- najboljša izbira za prve zamisli in skupinsko razpravo;
- ne spodbuja prezgodnjega ukvarjanja z barvami in tipografijo;
- odličen za pouk, ker jasno loči načrtovanje interakcije od programiranja;
- slabši za oddaljeno sodelovanje, natančno vizualno oblikovanje in kompleksno dinamično vedenje.

Marijin primer
Na list papirja narišemo začetni zaslon, izbiro zdravnika, seznam terminov in potrditev. Sošolec igra Marijo, drugi pa »računalnik« in ob dotiku zamenja list. Že brez programske kode lahko opazujemo, kje uporabnik okleva.

A.4 Namensko orodje za skelet (wireframe): Balsamiq

Balsamiq

Tipična vloga: skica, skelet in zgodnji interaktivni prototip

Balsamiq je namensko orodje zaskelete nizke vernosti. Njegova vizualna govorica namenoma spominja na skico, zato ekipo usmerja k strukturi, vsebini in toku namesto k »piksel popolni« grafiki. V aktualni različici 2026 podpira tudi ustvarjanjeskeletov in uporabniških tokov z umetno inteligenco ter generiranje interaktivnih prototipov iz skeletov.

Posebej uporabno: hitra izdelava grobih zaslonov, zgodnje testiranje, razprava z naročniki in študenti, ko želimo preprečiti prezgodnjo osredotočenost na estetiko.

Omejitve oziroma opozorila: Ni namenjen izdelavi končne vizualne podobe ali produkcijske aplikacije. Njegova namerna grobost je prednost v zgodnji fazi, ne slabost.

Mini primer – Marija: Iz knjižnice elementov sestavimo zaslon »Izberite termin«, nato povežemo zaslone in preverimo, ali Marija brez razlage najde naslednji korak.

A.5 Vizualno načrtovanje in prototipiranje

Figma

Tipična vloga: skeleti, makete, design sistema in interaktivni prototip visoke vernosti.

Figma združuje vizualno načrtovanje uporabniških vmesnikov, komponente, sodelovanje in prototipiranje. Prototipi lahko vključujejo različne dogodke, prehode, animacije, spremenljivke, pogojno logiko in matematične izraze. To omogoča precej realistično simulacijo izkušnje brez izdelave celotnega produkcijskega sistema.

Posebej uporabno: delo od srednje podrobnega skeleta do zelo dodelanega prototipa, skupinsko sodelovanje, ponovna uporaba komponent in predstavitev uporabniških tokov.

Omejitve oziroma opozorila: Ker omogoča zelo hitro vizualno dodelavo, obstaja nevarnost, da se ekipa prezgodaj ukvarja z videzom. Visoka vizualna podobnost še ne pomeni dobre uporabnosti.

Mini primer – Marija: Iz skeleta zaslona za termin izdelamo čitljivo maketo, dodamo velika polja in jasne oznake ter povežemo izbiro termina s potrditvenim zaslonom.

Penpot

Tipična vloga: skeleti, makete, interaktivni prototipi in povezava med oblikovanjem ter spletno implementacijo

Penpot je odprtokodna platforma za načrtovanje digitalnih izdelkov. Podpira skelete, interaktivne prototipe, komponente, oblikovalske žetone (design tokens) ter postavitve, povezane s spletnimi standardi CSS Grid in Flex. Posebnost je tesnejša povezava z odprtimi standardi, kot so CSS, HTML, SVG in JSON. Uporabljamo ga lahko kot storitev v oblaku ali ga namestimo na lasten strežnik.

Posebej uporabno: izobraževalna okolja, odprtokodni pristop, sodelovanje oblikovalcev in razvijalcev, organizacije, ki želijo več nadzora nad podatki in možnost lastne namestitve.

Omejitve oziroma opozorila: Tudi tu velja, da koda oziroma tehnični opis, pridobljen iz zasnove, ne odpravi potrebe po inženirskem razvoju, preverjanju dostopnosti in testiranju.

Mini primer – Marija: Marijin zaslon sestavimo z odzivno postavitvijo; nato povežemo table v uporabniški tok in preverimo obnašanje na različnih velikostih zaslona.

A.6 Orodje za zahtevnejše vedenjske prototipe: Axure RP

Axure RP

Tipična vloga: podrobni interaktivni in vedenjski prototipi

Axure RP je posebej močan, kadar mora prototip simulirati kompleksnejšo logiko. Podpira dogodke, pogoje, spremenljivke, večstanja, delujoča vnosna polja ter podatkovno vodene tabele in sezname. Z njim lahko brez klasičnega programiranja zgradimo prototip, ki se po vedenju približa funkcionalnemu sistemu.

Posebej uporabno: kompleksni poslovni sistemi, obrazci, pogojni tokovi, dinamična vsebina in situacije, kjer navaden klik med statičnimi zasloni ni dovolj.

Omejitve oziroma opozorila: Za preprost wireframe je lahko preobsežen. Večja moč pomeni tudi več časa za izdelavo in več možnosti, da prototip postane cilj sam zase.

Mini primer – Marija: Gumb za potrditev termina omogočimo samo, ko je termin izbran; ob manjkajočem izboru pokažemo opozorilo, po potrditvi pa shranimo izbrano vrednost v spremenljivko.

A.7 Od prototipa proti delujoči aplikaciji

FlutterFlow

Tipična vloga: vizualna zasnova in funkcionalni prototip oziroma aplikacija v Flutterju

FlutterFlow je vizualno razvojno okolje, ki sega precej dlje od klasičnega prototipa. Omogoča načrtovanje uporabniškega vmesnika, delo s podatki, API-ji, stanjem in logiko. Projekt predstavlja Flutter aplikacijo, generirano kodo pa je mogoče izvoziti in nadalje razvijati kot običajen Flutter projekt. FlutterFlow Designer leta 2026 omogoča tudi generiranje zasnov iz besedilnega opisa in prenos teh zasnov v dejanske FlutterFlow gradnike.

Posebej uporabno: mobilni in spletni funkcionalni prototipi, preverjanje na pravi napravi, prehod od vizualnega načrtovanja k dejanski aplikaciji ter pouk, kjer želimo pokazati povezavo med UI in implementacijo.

Omejitve oziroma opozorila: Ko prototip postane prava aplikacija, se pojavijo vprašanja arhitekture, varnosti, vzdrževanja, testiranja in kakovosti kode. Vizualno razvojno okolje teh vprašanj ne odpravi.

Mini primer – Marija: Zaslon za termin prenesemo v dejanske gradnike, povežemo testni vir terminov in na telefonu preverimo izbiro ter shranjevanje rezervacije.

Bubble

Tipična vloga: no-code razvoj funkcionalnih spletnih in mobilnih aplikacij

Bubble združuje vizualni uporabniški vmesnik, podatkovni model, delovne tokove in logiko. Njegov namen ni le prikaz prototipa, temveč izdelava delujoče aplikacije. Aktualna platforma vključuje tudi generiranje aplikacij iz besedilnega opisa ter vizualno urejanje ustvarjene rešitve.

Posebej uporabno: hiter funkcionalni dokaz koncepta, spletne storitve in primeri, kjer želimo brez tradicionalnega programiranja preveriti celoten tok od vmesnika do podatkov in logike.

Omejitve oziroma opozorila: No-code ne pomeni brez načrtovanja. Kompleksnejši sistemi še vedno zahtevajo dobro podatkovno zasnovo, varnost, upravljanje stanj, testiranje in razmislek o dolgoročnem vzdrževanju.

Mini primer – Marija: Izdelamo testno zbirko zdravnikov in terminov, uporabniku prikažemo proste termine in rezervacijo zapišemo v podatkovno zbirko.

A.8 Klasični razvoj kot prototipno orodje

HTML/CSS/JavaScript, React, Flutter, Android Compose in druga razvojna okolja lahko prav tako uporabimo za prototipiranje. To je smiselno, kadar je raziskovalno vprašanje povezano z lastnostjo, ki jo grafični prototip težko simulira: zmogljivostjo, odzivnostjo, uporabo senzorjev, animacijo, dostopnostjo na dejanski platformi ali integracijo z zunanjimi storitvami.

Slabost je cena spremembe. Če že na začetku vložimo veliko časa v programsko kodo, se težje odločimo za radikalno spremembo koncepta. Zato je programiranje pogosto učinkovitejše po tem, ko smo osnovno strukturo in tok že preverili s cenejšimi prototipi.

A.9 Orodja za uporabniške poti, tokove in sodelovanje

Načrtovanje UI ne poteka samo na ravni posameznega zaslona. Potrebujemo tudi prikaz uporabniških poti, informacijskih struktur, tokov in povezav med scenariji. Pri tem uporabljamo digitalne bele table, diagramiranje in skupne delovne površine. Primer takšnega pristopa je FigJam v ekosistemu Figma; podobne funkcije ponujajo tudi druga sodelovalna okolja.

Za učbenik je pomembnejši tip orodja kot znamka: študent mora znati narisati uporabniški tok in ga povezati s persono, scenarijem ter prototipom. Orodje je zamenljivo.

A.10 Umetna inteligenca pri načrtovanju vmesnikov

Leta 2026 je generativna umetna inteligenca postala del več kategorij UI-orodij. Iz besedilnega opisa lahko predlaga zaslone, uporabniške tokove, wireframe, vizualne različice ali celo delujočo aplikacijo. Balsamiq lahko iz besedila, kode, posnetka zaslona ali ročne skice pripravi wireframe; FlutterFlow Designer iz opisa ustvarja zaslone in jih prenaša v razvojno okolje; Bubble iz opisa ustvarja delujočo aplikacijo. Tudi sodobna oblikovalska okolja vključujejo vse več AI-podprtih postopkov.

Pomembno
UI, ki ga ustvari umetna inteligenca, je predlog rešitve, ne dokaz, da je rešitev uporabna. AI lahko pospeši izdelavo alternative, ne more pa nadomestiti razumevanja konkretnih uporabnikov in vrednotenja z njimi.

Didaktično je AI posebej zanimiva, če študenti ne sprejmejo prve generirane rešitve. Boljša vaja je: ustvarimo tri alternative, utemeljimo razlike, preverimo jih glede na persono in scenarij, nato pa eno preskusimo z uporabnikom.

A.11 Primerjalni pregled

Orodje	Skica	Skelet	Maketa	Interaktivni	Funkcionalni	Posebna vrednost
Papir / tabla	●●●	●	–	○	–	hitrost, svoboda
Balsamiq	●●	●●●	●	●●	–	namerno low-fi
Figma	●	●●	●●●	●●●	●	vizualna zasnova + prototip
Penpot	●	●●	●●●	●●●	●	open source, spletni standardi
Axure RP	●	●●	●●	●●●	●●	kompleksna logika prototipa
FlutterFlow	○	●	●●	●●	●●●	prehod v Flutter aplikacijo
Bubble	○	●	●●	●●	●●●	no-code full-stack aplikacija

Legenda: ●●● zelo primerno · ●● primerno · ● mogoče · ○ le obrobno · – ni glavni namen. Tabela je didaktična orientacija, ne absolutna ocena kakovosti orodij.

A.12 Katero orodje bi izbrali za Marijin primer?

Če želimo v eni učni nalogi prikazati celoten proces, ni treba uporabiti vseh orodij. Smiselna pot bi bila: prve skice na papirju; wireframe v Balsamiqu ali Penpotu; vizualno dodelan in interaktiven prototip v Figmi ali Penpotu; funkcionalni prototip v FlutterFlowu, Bubbleu ali s klasičnim programiranjem. Axure je posebej zanimiv, če želimo brez razvoja kode raziskati zahtevnejšo pogojno logiko.

Didaktično priporočilo
Študent naj vsaj enkrat začne s papirjem. Če takoj začne v zmogljivem grafičnem okolju, se lahko nauči uporabljati program, ne da bi se naučil prototipirati.

A.13 Kako presojamo novo orodje

Ker se bo seznam konkretnih izdelkov spreminjal, je koristno imeti merila, s katerimi lahko presodimo tudi orodje, ki danes še ne obstaja.

- Za katero fazo procesa je orodje namenjeno?
- Kako hitro lahko izdelamo in spremenimo alternativo?
- Ali podpira sodelovanje in komentiranje?
- Kako dobro podpira komponente in oblikovalske sisteme?
- Ali omogoča interakcije, stanja, pogoje in podatke, če jih potrebujemo?
- Ali lahko prototip preskusimo na ciljni napravi?
- Kako poteka prehod od zasnove do razvoja?
- Ali omogoča izvoz in prenosljivost podatkov oziroma kode?
- Kakšni so stroški, licenca in omejitve brezplačne različice?
- Kakšne so možnosti glede zasebnosti, lastne namestitve in dolgoročnega dostopa?
- Ali AI-funkcije res skrajšajo delo ali samo hitreje ustvarijo več nepreverjenih rešitev?

A.14 Kratke praktične vaje

1. Papir proti digitalnemu orodju. *Isti zaslon najprej skicirajte na papirju, nato izdelajte wireframe v izbranem programu. Primerjajte čas, število alternativ in vrsto odločitev, ki ste jih sprejemali.*

2. Primerjava dveh orodij. *Isti wireframe izdelajte v dveh različnih okoljih. Ne ocenjujte samo hitrosti, temveč tudi, kako orodje vpliva na vaš način razmišljanja.*

3. Odskeleta do prototipa. *Tri povezane zaslone pretvorite v interaktivni tok in ga preskusite z drugim študentom. Zabeležite vsako mesto oklevanja.*

4. AI kot pomočnik. *Z AI ustvarite predlog zaslona iz istega scenarija, ki ste ga prej načrtovali ročno. Poiščite najmanj pet odločitev AI in presodite, ali so utemeljene glede na persono.*

5. Funkcionalni prototip. *En majhen del interaktivnega prototipa – npr. izbiro in shranjevanje termina – izdelajte kot funkcionalni prototip. Ugotovite, katere nove težave postanejo vidne šele z dejanskimi podatki.*

A.15 Sklep

Dobro orodje ne nadomesti dobrega postopka. Z njim lahko hitreje izrazimo zamisel, jo pokažemo drugim in jo preverimo, vendar kakovost uporabniškega vmesnika še vedno izhaja iz razumevanja uporabnikov, pravilne izbire problema, iterativnega prototipiranja in vrednotenja. Orodja se bodo spreminjala; ta načela bodo ostala precej stabilnejša.

Zato naj študent ne postane »uporabnik Figme«, »uporabnik Penpota« ali »uporabnik FlutterFlowa«, temveč načrtovalec uporabniških vmesnikov, ki zna glede na namen izbrati primerno orodje in ga po potrebi zamenjati.

A.16 Spletni viri za posodabljanje dodatka

- **Figma – prototipiranje:** https://www.figma.com/prototyping/
- **Penpot – uradna stran in funkcije:** https://penpot.app/
- **Balsamiq – kaj je Balsamiq:** https://balsamiq.com/support/getting-started/what-is-balsamiq/
- **Axure RP – prototipiranje:** https://www.axure.com/axure-rp
- **FlutterFlow – dokumentacija:** https://docs.flutterflow.io/
- **Bubble – funkcije platforme:** https://bubble.io/features

Opomba: funkcije, licenčni modeli in ponudba posameznih izdelkov se spreminjajo. Pred uporabo pri pouku preverimo aktualne uradne strani.

Dodatek B: Kontrolni seznam za načrtovanje in vrednotenje uporabniškega vmesnika

Kontrolni seznam ni nadomestilo za raziskovanje uporabnikov ali testiranje uporabnosti. Je pripomoček, s katerim pred oddajo zasnove, prototipa ali izdelka sistematično preverimo, ali smo spregledali katero od pogostih težav. Posamezne postavke niso absolutna pravila; vedno jih presojamo glede na uporabnike, nalogo, napravo in okoliščine uporabe.

> **Kako uporabljamo seznam**
> Pri vsaki postavki označimo: izpolnjeno, ni izpolnjeno ali ni relevantno. Pri pomembnih neizpolnjenih postavkah zapišemo, kaj bomo spremenili in kako bomo spremembo preverili.

B.1 Uporabniki, cilji in kontekst

☐ Vemo, kdo so primarne skupine uporabnikov in na katerih podatkih temelji ta opredelitev.

☐ Poznamo glavne cilje in naloge uporabnikov, ne le funkcij sistema.

☐ Upoštevali smo razlike v znanju, izkušnjah, sposobnostih in digitalni pismenosti.

☐ Upoštevali smo naprave, okolje, pogostost uporabe, časovne pritiske in morebitne prekinitve.

☐ Če uporabljamo persone, te temeljijo na podatkih in ne na stereotipih.

☐ Za ključne naloge imamo scenarije oziroma uporabniške poti.

B.2 Informacijska arhitektura in navigacija

☐ Uporabnik lahko hitro prepozna, kje je in kaj lahko naredi.

☐ Najpomembnejše funkcije in vsebine so dovolj vidne in logično razvrščene.

☐ Poimenovanja kategorij, menijev in ukazov uporabljajo jezik uporabnika.

☐ Navigacija je dosledna med zasloni.

☐ Uporabnik se lahko vrne iz napačne poti brez izgube dela.

☐ Globina in kompleksnost navigacije sta sorazmerni z nalogo.

B.3 Razumljivost in miselni model

☐ Vmesnik uporablja pojme, ki so uporabniku razumljivi.

☐ Razmerje med dejanjem in rezultatom je jasno.

☐ Ikone, kadar niso splošno razumljive, spremlja besedilo ali druga razlaga.

☐ Vmesnik ne zahteva poznavanja notranje tehnične strukture sistema.

☐ Podobne funkcije se obnašajo podobno.

☐ Uporabnik lahko iz prikaza pravilno sklepa o trenutnem stanju sistema.

B.4 Vidnost stanja in povratna informacija

☐ Sistem po vsakem pomembnem dejanju pravočasno pokaže, da je dejanje zaznal.

☐ Pri daljšem postopku je uporabniku jasno, da sistem še deluje.

☐ Po uspešnem zaključku je rezultat nedvoumen.

☐ Napake in opozorila so vidna, razumljiva in postavljena ob problem.

☐ Uporabnik lahko razlikuje med izbranim, neizbranim, omogočenim in onemogočenim stanjem.

B.5 Nadzor uporabnika in obravnava napak

☐ Uporabnik lahko prekliče ali razveljavi pomembna dejanja, kjer je to smiselno.

☐ Destruktivna ali težko popravljiva dejanja so ustrezno zaščitena.

☐ Sistem preprečuje napake, kadar jih je mogoče zanesljivo predvideti.

☐ Sporočilo o napaki pove, kaj se je zgodilo in kaj lahko uporabnik naredi.

☐ Uporabnik zaradi manjše napake ne izgubi po nepotrebnem že vnesenih podatkov.

☐ Obnovitev po napaki je preprostejša od ponovnega začetka celotne naloge.

B.6 Vizualna hierarhija in čitljivost

☐ Najpomembnejši elementi so vizualno dovolj poudarjeni.

☐ Besedilo je dovolj veliko in čitljivo na ciljni napravi.

☐ Kontrast med besedilom, ozadjem in pomembnimi elementi je ustrezen.

☐ Barva ni edini način posredovanja pomembne informacije.

☐ Razmiki, poravnave in skupine pomagajo razumeti strukturo.

☐ Vmesnik ni po nepotrebnem vizualno nasičen.

B.7 Vnos podatkov in obrazci

☐ Zahtevamo samo podatke, ki jih za nalogo res potrebujemo.

☐ Oznake polj so jasne in ostanejo razumljive tudi med vnosom.

☐ Format pričakovanega vnosa je razviden pred napako.

☐ Privzete vrednosti so varne in smiselne.

☐ Zaporedje polj sledi naravnemu poteku naloge.

☐ Preverjanje vnosa uporabniku pomaga, namesto da ga kaznuje.

B.8 Dostopnost in vključujoče načrtovanje

☐ Vmesnik je uporaben tudi brez zanašanja samo na barvo, zvok ali natančen kazalec.

☐ Interaktivni elementi so dovolj veliki in imajo dovolj prostora okoli sebe.

☐ Vmesnik podpira povečavo besedila in različne velikosti zaslona, kjer je to relevantno.

☐ Zaporedje fokusa in upravljanje s tipkovnico sta smiselna pri vmesnikih, kjer se uporabljata.

☐ Slike in netekstovne informacije imajo ustrezne besedilne alternative, kjer so potrebne.

☐ Animacije in časovne omejitve ne ustvarjajo nepotrebnih ovir.

B.9 Prototipiranje

☐ Pred podrobno izdelavo smo raziskali več kot eno možno rešitev.

☐ Stopnja realističnosti prototipa ustreza vprašanju, ki ga želimo preveriti.

☐ Strukture nismo preskočili tako, da bi takoj začeli z vizualnimi podrobnostmi.

☐ Interaktivni prototip vključuje ključne poti, povratne informacije in vsaj pomembne napake.

☐ Vemo, kaj v prototipu deluje zares in kaj je samo simulirano.

☐ Po ugotovitvah testiranja prototip dejansko spreminjamo.

B.10 Vrednotenje

☐ Pred testiranjem smo določili naloge in vprašanja, na katera želimo odgovoriti.

☐ Pri testiranju opazujemo vedenje uporabnika, ne le njegovih mnenj.

☐ Moderator uporabniku ne razlaga vmesnika, če želimo preveriti njegovo razumljivost.

☐ Zabeležimo mesta oklevanja, napake, vračanja in nepričakovane poti.

☐ Težave razvrstimo po pomembnosti in pogostosti.

☐ Po spremembah ponovno preverimo vsaj ključne naloge.

B.11 Pred uvedbo in po njej

☐ Preverili smo delovanje na dejanskih ciljnih napravah in velikostih zaslona.

☐ Preverili smo pomembne robne primere, prazna stanja in napake povezave.

☐ Varnostna in zasebnostna opozorila so razumljiva uporabniku.

☐ Uporabnik lahko najde pomoč, kadar jo potrebuje.

☐ Po uvedbi zbiramo povratne informacije in spremljamo težave v resnični uporabi.

☐ Končni sistem obravnavamo kot začetek novega cikla izboljšav, ne kot konec HCI procesa.

B.12 Hiter pregled pred oddajo

Vprašanje	Da / Ne / N. r.	Opomba
Ali uporabnik razume, kaj je glavni cilj zaslona?	☐ Da ☐ Ne ☐ N. r.	
Ali je naslednji korak dovolj očiten?	☐ Da ☐ Ne ☐ N. r.	
Ali sistem jasno pokaže rezultat dejanja?	☐ Da ☐ Ne ☐ N. r.	
Ali se uporabnik lahko vrne ali popravi napako?	☐ Da ☐ Ne ☐ N. r.	
Ali so besedilo, kontrast in velikost elementov primerni?	☐ Da ☐ Ne ☐ N. r.	
Ali smo preverili uporabo brez naših razlag?	☐ Da ☐ Ne ☐ N. r.	

Ali smo vključili pomembna stanja napak in prazne rezultate?	☐ Da ☐ Ne ☐ N. r.	
Ali rešitev deluje za različne relevantne uporabnike?	☐ Da ☐ Ne ☐ N. r.	
Ali smo ugotovitve testiranja pretvorili v spremembe?	☐ Da ☐ Ne ☐ N. r.	
Ali lahko vsako pomembno oblikovalsko odločitev utemeljimo z uporabnikom, nalogo ali dokazom?	☐ Da ☐ Ne ☐ N. r.	

Dodatek C: Slovarček izrazov HCI

Slovarček navaja slovenske izraze skupaj z najpogostejšimi angleškimi ustreznicami. Pri nekaterih izrazih se v slovenski strokovni rabi uporablja tudi angleški izraz; zato je angleško poimenovanje navedeno posebej. Definicije so namenoma kratke in usmerjene v pomen izraza v komunikaciji človek–računalnik.

Slovenski izraz	**Angleški izraz**	**Pomen**
Dopustnost / zaznana možnost delovanja	affordance / perceived affordance	Lastnost oziroma zaznani namig, ki uporabniku pomaga razumeti, kako lahko z objektom ali elementom ravna.
AR / obogatena resničnost	augmented reality, AR	Tehnologija, ki resničnemu okolju dodaja digitalne informacije ali objekte.
oblikovalski sistem	design system	Usklajen nabor pravil, komponent, slogov in vzorcev za dosledno oblikovanje digitalnega izdelka.
Oblikovalski žeton	Design token	Oblikovalski žetoni so nedeljivi deli oblikovalskega sistema, kot so barve, razmik, tipografska lestvica.
direktna manipulacija	direct manipulation	Način interakcije, pri katerem uporabnik neposredno deluje na vidne objekte in takoj vidi rezultat dejanja.

dostopnost	accessibility	Lastnost sistema, da ga lahko uporabljajo ljudje z različnimi sposobnostmi in oviranostmi; vključuje zaznavne, motorične in kognitivne vidike.
KČR / komunikacija človek–računalnik	HCI / human–computer interaction	Področje, ki proučuje načrtovanje, uporabo in vrednotenje interaktivnih računalniških sistemov v odnosu do ljudi.
heuristična evalvacija	heuristic evaluation	Strokovni pregled vmesnika glede na uveljavljena načela oziroma hevristike uporabnosti.
Prototip visoke vernosti	high-fidelity prototype	Prototip, ki je po videzu in/ali vedenju razmeroma blizu končnemu sistemu.
horizontalni prototip	horizontal prototype	Prototip, ki površinsko zajame širok nabor funkcij, ne da bi jih v globino implementiral.
informacijska arhitektura	information architecture	Organizacija, poimenovanje in povezovanje informacij ter funkcij tako, da jih uporabnik lahko najde in razume.
interakcija	interaction	Izmenjava dejanj in odzivov med uporabnikom in sistemom.
interaktivni prototip	interactive prototype	Prototip, ki omogoča klike, dotike, prehode in druge simulirane odzive ter s tem preverjanje uporabniškega toka.
kognitivna obremenitev	cognitive load	Količina miselnega napora, ki ga naloga ali vmesnik zahteva od uporabnika.
konceptualni model	conceptual model	Načrtovalčeva organizacija ključnih pojmov, objektov, dejanj in odnosov, na katerih temelji sistem.
kontekst uporabe	context of use	Uporabniki, naloge, oprema ter fizično, družbeno in organizacijsko okolje, v katerem se sistem uporablja.
Prototip nizke vernosti	low-fidelity prototype	Hiter in grob prototip z malo vizualnimi podrobnostmi, namenjen zgodnjemu

		raziskovanju strukture in poteka.
miselni model	mental model	Uporabnikova notranja predstava o tem, kako sistem deluje in kako se bo odzval na dejanja.
maketa	mockup	Statičen, vizualno razmeroma natančen prikaz vmesnika, namenjen preverjanju videza, hierarhije in čitljivosti.
modalnost	modality	Način, po katerem sistem sprejema ali posreduje informacije, npr. vid, zvok, dotik ali govor.
naravno preslikavanje	natural mapping	Razmerje med kontrolami in učinki, ki je uporabniku razumljivo zaradi prostorske, kulturne ali druge naravne povezave.
odzivnost	responsiveness	Sposobnost sistema, da se na uporabnikova dejanja odziva dovolj hitro in predvidljivo; pri spletnih vmesnikih tudi prilagajanje različnim velikostim zaslona.
Persona uporabnika	persona	Na podatkih temelječ reprezentativen opis značilnega uporabnika ali skupine uporabnikov, uporabljen pri načrtovanju.
Poosebljenje	personalization	Sistem samodejno prilagodi svojo vsebino, videz ali funkcionalnost tako, da ustrezajo željam, vedenju ali vlogi posameznika.
povratna informacija	feedback	Informacija sistema, ki uporabniku pokaže, da je bilo dejanje zaznano, kakšno je stanje in kakšen je rezultat.
prototip	prototype	Namerno nepopolna predstavitev prihodnjega sistema, izdelana zato, da lahko zamisel preverimo in spreminjamo pred dokončno implementacijo.
scenarij uporabe	usage scenario	Kratek opis uporabnika, njegovega cilja, okoliščin in poteka naloge brez nepotrebnega predpisovanja tehnične

		rešitve.
Zaznavni znak	signifier	Vidni, zvočni ali drug znak, ki uporabniku sporoča, kje in kako je mogoče izvesti dejanje.
skica	sketch	Zelo hiter, grob prikaz zamisli za vmesnik, pogosto izdelan na papirju ali tabli.
uporabniška izkušnja	user experience, UX	Celota uporabnikovih zaznav, odzivov in izkušenj pred, med in po uporabi izdelka ali storitve.
uporabniška pot	user journey	Zaporedje korakov, skozi katere uporabnik prehaja od potrebe do cilja, skupaj z dejanji, vprašanji, težavami in občutki.
uporabniški tok	user flow	Predstavitev poti oziroma zaporedja zaslonov in odločitev, skozi katere uporabnik opravi določeno nalogo.
uporabniški vmesnik	user interface, UI	Del interaktivnega sistema, prek katerega uporabnik zaznava stanje sistema in izvaja dejanja.
uporabniško usmerjeno načrtovanje	human-centred design / user-centred design	Iterativen pristop, pri katerem načrtovanje temelji na razumevanju uporabnikov, njihovih nalog in konteksta ter na vrednotenju rešitev z uporabniki.
uporabljivost	usability	Stopnja, do katere lahko določeni uporabniki v določenem kontekstu učinkovito, uspešno in zadovoljivo dosežejo svoje cilje.
vertikalni prototip	vertical prototype	Prototip, ki podrobno izdela ozek del sistema ali eno funkcionalno pot.
vidnost stanja sistema	visibility of system status	Načelo, da mora sistem uporabniku pravočasno in razumljivo sporočati svoje trenutno stanje.
VR / navidezna resničnost	virtual reality, VR	Interaktivno računalniško ustvarjeno okolje, ki uporabniku omogoča občutek

		prisotnosti v virtualnem prostoru.
vrednotenje uporabnosti	usability evaluation	Sistematično preverjanje, kako dobro vmesnik podpira uporabnike pri njihovih nalogah.
Skelet	wireframe	Strukturiran, navadno nizko podroben načrt zaslona, ki prikazuje razporeditev, hierarhijo in navigacijo brez dokončne grafične podobe.
XR / razširjena resničnost	extended reality, XR	Krovni izraz za tehnologije navidezne, obogatene in mešane resničnosti.

Terminološka opomba
Pri nekaterih izrazih slovenska terminologija ni povsem poenotena. V učbeniku je zato smiselno ob prvi uporabi navesti slovenski izraz in angleški ustreznik, nato pa dosledno uporabljati izbrano obliko.

Dodatek D: Literatura in priporočeni viri

Spodnji izbor je namenjen študentu in učitelju kot izhodišče za nadaljnji študij. Literatura je razdeljena na temeljna dela, sodobne učbenike, uporabniško usmerjeno načrtovanje in UX, dostopnost in standarde ter specializirana področja. Pri spletnih virih in standardih je pred uporabo priporočljivo preveriti najnovejšo različico.

D.1 Temeljna dela HCI in oblikovanja interakcije

- Norman, D. A. (2013). *The Design of Everyday Things: Revised and Expanded. Basic Books.*
- Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (6th ed.). Pearson.
- Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). *Human-Computer Interaction* (3rd ed.). Pearson.
- Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates.

D.2 Sodobni učbeniki in širši pregledi

- Rogers, Y., Sharp, H., & Preece, J. (2023). *Interaction Design: Beyond Human-Computer Interaction* (6th ed.). Wiley.
- Benyon, D. (2019). *Designing User Experience: A Guide to HCI, UX and Interaction Design* (4th ed.). Pearson.

- Lazar, J., Feng, J. H., & Hochheiser, H. (2017). *Research Methods in Human-Computer Interaction* (2nd ed.). Morgan Kaufmann.

D.3 Uporabniško usmerjeno načrtovanje, UX in prototipiranje

- Krug, S. (2014). *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability* (3rd ed.). New Riders.
- Garrett, J. J. (2011). *The Elements of User Experience: User-Centered Design for the Web and Beyond* (2nd ed.). New Riders.
- Buxton, B. (2007). *Sketching User Experiences: Getting the Design Right and the Right Design.* Morgan Kaufmann.
- Goodwin, K. (2009). *Designing for the Digital Age: How to Create Human-Centered Products and Services*. Wiley.
- Snyder, C. (2003). *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces.* Morgan Kaufmann.
- Nielsen, J. (1993). *Usability Engineering*. Academic Press.

D.4 Kognitivni vidiki, vizualno zaznavanje in oblikovanje

- Johnson, J. (2020). *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. (3rd ed.). Morgan Kaufmann.
- Ware, C. (2021). *Information Visualization: Perception for Design* (4th ed.). Morgan Kaufmann.
- Tufte, E. R. (2001). *The Visual Display of Quantitative Information* (2nd ed.). Graphics Press.

D.5 Dostopnost in vključujoče načrtovanje

- World Wide Web Consortium (W3C). *Web Content Accessibility Guidelines (WCAG)* 2.2. W3C Recommendation.
- ISO 9241-210. *Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems.*
- ISO 9241-11. *Ergonomics of human-system interaction — Part 11: Usability: Definitions and concepts.*
- Clarkson, P. J., Coleman, R., Keates, S., & Lebbon, C. (eds.) (2003). *Inclusive Design: Design for the Whole Population*. Springer.

D.6 Mobilni, vseprisotni in novi načini interakcije

- Hoober, S., & Berkman, E. (2011). *Designing Mobile Interfaces*. O'Reilly Media.
- Jerald, J. (2015). *The VR Book: Human-Centered Design for Virtual Reality.* ACM Books / Morgan & Claypool.
- LaViola, J. J., Kruijff, E., McMahan, R. P., Bowman, D., & Poupyrev, I. (2017). *3D User Interfaces: Theory and Practice* (2nd ed.). Addison-Wesley.

D.7 Umetna inteligenca in HCI

Amershi, S., et al. (2019). *Guidelines for Human-AI Interaction.* Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. ACM.

D.8 Pomembni spletni in strokovni viri

Vir	Zakaj je uporaben	Naslov
W3C Web Accessibility Initiative (WAI)	Smernice, tehnike in izobraževalna gradiva o spletni dostopnosti.	https://www.w3.org/WAI/
Nielsen Norman Group	Članki in raziskovalni povzetki o uporabnosti, UX in interakcijskem oblikovanju.	https://www.nngroup.com/
Interaction Design Foundation	Pregledi pojmov, metod in tem HCI/UX; del vsebin je javno dostopen.	https://www.interaction-design.org/
ACM Digital Library	Primarni znanstveni vir za HCI; posebej konferenci CHI in UIST ter revije ACM.	https://dl.acm.org/
IEEE Xplore	Znanstveni članki in konferenčni prispevki s področij HCI, vizualizacije, VR/AR in sorodnih tehnologij.	https://ieeexplore.ieee.org/

D.9 Kako uporabljati literaturo pri študiju

Za osnovno razumevanje je smiselno kombinirati sodoben splošni učbenik z izbranimi klasičnimi deli. Starejša literatura pri HCI ni nujno zastarela: temeljna spoznanja o zaznavanju, povratni informaciji, miselnih modelih, napakah in uporabniško usmerjenem načrtovanju ostajajo relevantna, čeprav so se naprave in konkretni vmesniki spremenili.

Pri hitro spreminjajočih se področjih – generativni AI, XR, konkretna razvojna orodja, platformne smernice – naj učbenik ne bo edini vir. Študent naj preveri aktualno dokumentacijo, standarde in nove raziskovalne članke. Pri znanstvenih trditvah imajo prednost primarni raziskovalni viri pred povzetki in marketinškimi gradivi.

Predlog za študente
Pri seminarskem ali projektnem delu uporabite vsaj en splošni vir HCI, en vir o izbrani metodi ali problemu ter ustrezne primarne raziskovalne vire oziroma standarde. Pri hitro spreminjajočih se tehnologijah preverite tudi aktualnost uporabljenih virov.