UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Aleš Kalan

Razvoj uporabniškega vmesnika na dotik za aplikacije v zdravstvu

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: viš. pred. dr. Alenka Kavčič

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00031/2010
Datum: 04.10.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ALEŠ KALAN**

Naslov: **RAZVOJ UPORABNIŠKEGA VMESNIKA NA DOTIK ZA APLIKACIJE V ZDRAVSTVU**

DEVELOPMENT OF TOUCH USER INTERFACE FOR APPLICATIONS IN HEALTHCARE

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi proučite primernost uporabe zaslona, občutljivega za dotik, v zdravstvu.

Predlagajte primeren model razvoja uporabniškega vmesnika za aplikacije v zdravstvu, ki za komunikacijo z uporabnikom uporabljajo za dotik občutljiv zaslon. Pri tem upoštevajte vse posebnosti omenjenih namenskih aplikacij, pa tudi njihovih uporabnikov.

Predlagan model razvoja uporabite pri izdelavi novega uporabniškega vmesnika za zdravstveno aplikacijo, ki bo namesto miške in tipkovnice uporabljal za dotik občutljiv zaslon. Na koncu ocenite prednosti realiziranega uporabniškega vmesnika ter podajte kritično presojo njegovega delovanja. Naredite primerjavo aplikacije, ki deluje preko miške in tipkovnice, z novo aplikacijo, ki deluje preko zaslona, občutljivega za dotik. Primerjava naj zajema tako proces razvoja uporabniškega vmesnika aplikacije kot tudi uporabo aplikacije in odziv uporabnikov.

Mentor:
viš. pred. dr. Alenka Kavčič

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a _____Aleš Kalan_____,

z vpisno številko _____63060144_____,

sem avtor/-ica diplomskega dela z naslovom:

Razvoj uporabniškega vmesnika na dotik za aplikacije v zdravstvu_____

_____

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)
  Mentor: viš. pred. dr. Alenka Kavčič_____

  in somentorstvom (naziv, ime in priimek)

  _____/_____

- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela

- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne _____8.4.2011_____ Podpis avtorja/-ice: _____

Zahvala

Iskreno se zahvaljujem mentorici viš. pred. dr. Alenki Kavčič za izredno odzivnost ter izčrpno in hitro pomoč pri realizaciji mojega diplomskega dela.

Prav tako bi se rad zahvalil sodelavcem, ki so mi pomagali pri oblikovanju ideje in iskanju virov za diplomsko nalogo.

Zahvala gre tudi moji družini in prijateljem, ki so mi stali ob strani in me podpirali na študijski poti od začetka do konca.

Kazalo

Seznam uporabljenih kratic

CLI - command line interface - vmesnik z ukazno vrstico

GUI - graphical user interface - grafični uporabniški vmesnik

PDF - Portable Document Format - tip datoteke za shranjevanje dokumentov

ROP - računalnik ob postelji

TTL - terapevtsko temperaturni list

UI - user interface - uporabniški vmesnik

Povzetek

Zaradi velikega razmaha uporabe zaslonov, občutljivih na dotik, se je začel pospešeno razvijati tudi uporabniški vmesnik za te naprave. Uporabniški vmesnik je namreč glavno sredstvo za komunikacijo uporabnika z napravo, zato je bil tak razvoj nujno potreben. Glavni cilj razvoja je enostavnost, saj želimo uporabo teh zaslonov omogočiti tudi manj veščim uporabnikom, ki jih šele spoznavajo.

Osrednja tema tega diplomskega dela je razvoj uporabniškega vmesnika za zaslone na dotik od sprejema uporabniških zahtev do končne različice. Posebej je poudarjen postopek testiranja prototipa in žive različice uporabniškega vmesnika s končnimi uporabniki. Vsi postopki so opisani s primeri iz prakse razvoja takega uporabniškega vmesnika.

Pristop k razvoju uporabniškega vmesnika, ki sem ga opisal v diplomski nalogi, smo uporabili pri razvoju uporabniškega vmesnika za zaslone na dotik za aplikacijo, ki se uporablja v zdravstvenih ustanovah.

Ključne besede:

grafični uporabniški vmesnik, zaslon na dotik, test uporabnosti, prototip, proces razvoja uporabniškega vmesnika za zaslone na dotik

Abstract

Because of drastically increasing use of touchscreen devices the need for development of their user interface has increased accordingly. Graphical user interface is the main means of communication with the device. Hence, its development has become absolutely vital. The main goal of such development is simplicity, which enables less adept users to use these screens.

The main topic of this thesis is the development of graphical user interface from user demands to finished user interface. Emphasis is put on testing mockups and the live version of the user interface with final users. All processes are documented with examples from practice.

The approach for the development of an user interface, which I described in this thesis, has been used in the development of a graphical user interface for touch screens for an application, that is in use in medical institutions.

Keywords:

graphical user interface, touchscreen, usability testing, mockup, development of touchscreen user interface

1 Uvod

V diplomski nalogi bom prikazal razvoj uporabniškega vmesnika za zaslone, občutljive na dotik. To tematiko sem si izbral zaradi svojih delovnih izkušenj pri podjetju, ki razvija informacijske rešitve za zdravstvo. V podjetju smo želeli razviti nov izdelek, s katerim bi pokrili čim več zahtev uporabnikov, zato smo pričeli tudi z razvojem uporabniškega vmesnika zanj. Želeli smo, da bi bilo naš izdelek mogoče uporabljati na oba načina: z uporabo miške in tipkovnice in z uporabo zaslona na dotik. Morali smo torej razviti uporabniški vmesnik, ki omogoča tudi uporabo te nove tehnologije. Zaslon, občutljiv na dotik, smo izbrali zaradi potrebe po večji mobilnosti računalnika (od pacienta do pacienta). S tem smo zmanjšali težo in velikost strojne opreme, poleg tega so taki zasloni tudi bolj higienični.

1.1 Motivacija

Za opis postopka razvoja grafičnega uporabniškega vmesnika sem se odločil zato, ker so zasloni na dotik dokaj nova tehnologija. Temu sledi tudi nov način razvoja uporabniškega vmesnika, prilagojen posebej tem napravam. Uporabniški vmesnik za zaslone na dotik bi bil eden prvih pri nas v zdravstvenih aplikacijah, zato bi v podjetju lahko razširili ponudbo takih aplikacij in s tem pokrili večino zahtev na trgu.

1.2 Namen

V diplomski nalogi želim predstaviti:

- zaslone na dotik
- primernost njihove uporabe v zdravstvu
- zbiranje potrebnih podatkov od strank
- razvoj uporabniškega vmesnika za zaslone na dotik
- prednosti uporabe zaslonov na dotik
- razlike (na konkretnem primeru) med aplikacijo, ki uporablja miško in tipkovnico, in aplikacijo, namenjeno zaslonom na dotik

1.3 Cilji

Cilji diplomske naloge so:

- prikazati model razvoja uporabniškega vmesnika za zaslone na dotik, ki se uporablja v zdravstvenih ustanovah
- ugotoviti prednosti in slabosti takega uporabniškega vmesnika v primerjavi s klasičnim, ki uporablja miško in tipkovnico
- predstaviti posebnosti pri razvoju uporabniškega vmesnika za zdravstvene ustanove

2 Grafični uporabniški vmesnik in računalniške aplikacije v zdravstvu

Grafični uporabniški vmesnik omogoča komunikacijo med človekom in računalnikom oz. napravo, ki uporablja grafične elemente, kot so npr. ikone, okna in meniji, ki jih je možno nadzorovati z napravo za kazanje. Za interakcijo z gradniki uporabniškega vmesnika se pogosto uporabljata miška in tipkovnica [3, 4]. Razlika med takim vmesnikom in predhodnikom, ki je omogočal samo vnos tekstovnih ukazov, je velika.

Prednost GUI (*graphical user interface*) pred CLI (*command line interface*; vmesnik z ukazno vrstico) je predvsem v tem, da ga lahko uporabljajo tudi uporabniki z manj računalniškega znanja, saj so akcije veliko bolj intuitivne. Ljudje si namreč hitreje zapomnimo podobe kot pa golo besedilo, zato se tak vmesnik hitreje naučimo uporabljati. Ob izvedbi akcije se pri grafičnem uporabniškem vmesniku v trenutku prikaže rezultat oziroma povratna informacija in tako čutimo, da smo izvedli želeno akcijo [2].

Grafični uporabniški vmesniki se pogosto uporabljajo v mnogih elektronskih napravah, kot so na primer mobiteli, glasbeni predvajalniki, dlančniki, tablični računalniki in računalniki nasploh.

2.1 Grafični uporabniški vmesnik za zaslone na dotik

Pri razvoju uporabniškega vmesnika za zaslone, občutljive na dotik, moramo biti posebno pozorni, saj se nekoliko razlikuje od razvoja drugih uporabniških vmesnikov.

Glavne lastnosti, ki razlikujejo GUI za zaslone na dotik od običajnega uporabniškega vmesnika, ki uporablja tipkovnico in miško, so:

- uporaba prsta zmanjša natančnost izbire, zato moramo vse elemente prilagoditi prstom
- bližnjic na tipkovnici ni, ker je pri napravah z zasloni na dotik navadno sploh nimamo
- zasloni na dotik so po navadi manjši, zato je potrebno vse elemente primerno povečati
- zaradi velikosti zaslona je prikaz podatkov drugačen – prikažemo samo tisto, kar trenutno potrebujemo
- zaradi odsotnosti miške funkcionalnosti lebdenja odpadejo
- namesto fizične tipkovnice se uporablja programska tipkovnica, ki se pojavi na zaslonu pod elementom, v katerega vnašamo podatke
- zaradi počasnosti pisanja z uporabo programske tipkovnice se poskušamo izogniti vnašanju podatkov preko nje
- pisanje ni vezano samo na programsko tipkovnico, ampak je možno tudi s posebno programsko opremo, ki prepoznava pisavo: s posebnim pisalom pišemo po zaslonu in napisano se pretvori v standardne znake oz. besede
- če zaslon, občutljiv na dotik, podpira večtočkovno zaznavanje, je možna tudi uporaba posebnih kretenj s prsti

2.2 Zaslon, občutljiv na dotik

Zaslon, občutljiv na dotik, je elektronski zaslon, ki lahko zazna mesto, kjer se ga dotaknemo. Dotikamo se ga lahko na dva glavna načina: s prstom ali s posebnim pisalom. Slika 1 prikazuje primer dveh izdelkov, ki uporabljata zaslon na dotik, ter njuna uporabniška vmesnika [2, 16].

Slika 1: Primer uporabniškega vmesnika za iOS in konkurenčni izdelek Android

2.2.1 Prednosti zaslonov na dotik

Prednosti zaslonov, občutljivih na dotik, so sledeče:

- taki zasloni omogočajo neposredno interakcijo s prikazano vsebino brez uporabe dodatnega pripomočka, kot je na primer miška
- uporaba takega zaslona je veliko bolj intuitivna kot uporaba klasičnega grafičnega uporabniškega vmesnika, ki uporablja miško, saj nadzorujemo sistem neposredno preko zaslona [20]

2.2.2 Zgodovina

Prvi senzorji, občutljivi na dotik, so bili predstavljeni že leta 1971, vendar ti še niso bili prosojni. Prvi prosojni senzor so razvili v podjetju Elographics tri leta kasneje. V letu 1977 so pri istem podjetju razvili najpopularnejše senzorje, ki delujejo na tehnologiji električnega upora (*five-wire resistive technology*) in so v uporabi še danes.

Zaslone na dotik uporabljamo predvsem na mestih, kjer tipkovnica in miška nista dovolj hitri in intuitivni, npr. v kioskih, na informacijskih točkah, v prodajnih centrih, v težki industriji,

itn. Vedno pogosteje se pojavljajo tudi v vsakodnevnih elektronskih napravah, kot so mobilni telefoni, tablični računalniki, igralne konzole in podobne naprave [1] .

V zadnjem času se razvoj usmerja v sisteme, ki so sposobni zaznavati več dotikov hkrati (*multitouch*). Za take zaslone je mogoče razviti še bolj intuitivne aplikacije oz. uporabniške vmesnike.

2.2.3 Tehnologija

Z leti se je razvilo (in se še vedno razvija) veliko različnih tipov zaslonov na dotik. Tehnologija upornosti (*resistive technology*) deluje tako, da na mestu dotika staknemo dve prevodni plasti in tako spremenimo prevodnost. Zanimiva je tudi rešitev z ultrazvočnimi valovi, ki potujejo nad zaslonom, in ko jih s prstom ali peresom prekinemo, se absorbira nekaj valovanja in tako sistem zazna kraj dotika. Taka tehnologija je občutljiva na zunanje vplive, kot so praske in umazanija.

Vse več novejših elektronskih naprav uporablja kapacitivne (*capacitive*) zaslone. Ti so namreč zanesljivejši, odzivnejši, fleksibilnejši in omogočajo večtočkovno zaznavanje [1]. Za določanje mesta dotika uporabljajo električno prevodnost človeškega prsta ali posebej za to prilagojenega pisala. Ob dotiku točke na zaslonu se spremenijo lastnosti elektrostatike, iz katerih algoritem nato izračuna koordinate dotika. Natančnost takega zaslona je določena z gostoto mreže točk na njem [3, 20]; mrežo točk na zaslonu prikazuje slika 2.

Slika 2: Mreža točk na kapacitivnem zaslonu

2.2.4 Pomanjkljivosti

Največja pomanjkljivost zaslonov na dotik je slaba ergonomičnost, saj človeško telo ni prilagojeno na daljše držanje rok pred seboj. Poleg tega trpi tudi vrat, saj so ti zasloni pogosto

nameščeni na višini rok, kar povzroči nižji kot gledanja, kot je udobno [1] . Ob daljšem času uporabe s prstom lahko pride do neprijetne bolečine, ker je na nekaterih zaslonih treba močneje pritisniti, da zaznajo dotik. Zato nekateri proizvajalci ta neprijetni občutek odpravljajo s posebnimi pisali, ki so tudi znatno natančnejša kot prsti.

2.2.5 Uporabljena tehnologija

Pri razvoju našega uporabniškega vmesnika za zaslon, občutljiv na dotik, smo se nekoliko omejili glede tehnologije zaslonov. Zasloni z večtočkovnim zaznavanjem so dražji in težje dostopni, zato smo se usmerili v razvoj uporabniškega vmesnika, ki ne uporablja takih funkcionalnosti. Cena nakupa strojne opreme bo tako nižja in posledično bo prodaja končnega izdelka večja.

Za testiranje uporabniškega vmesnika smo uporabili prenosni računalnik HP EliteBook 2730p [7]. Njegov zaslon na dotik temelji na uporniško pasivni tehnologiji, ki omogoča uporabo s pisalom ali s prstom. Zaradi majhnosti zaslona (12,1") in možnosti vrtenja za 180 stopinj je izredno prijeten za uporabo. Večino funkcij lahko izvedemo z uporabo prstov, za večjo natančnost pa imamo na voljo pisalo, ki se skriva v ohišju

2.3 Uporaba računalnikov in programske opreme v zdravstvu

Uporaba računalnikov in programske opreme se je tako kot v vseh panogah razširila tudi na področju zdravstva. To je zelo občutljivo področje, saj je od programske in strojne opreme odvisno zdravje in včasih tudi življenje bolnika. Prisotne morajo biti redundantne funkcije, ki skrbijo, da se morebitne napake v sistemu hitro odkrijejo in odpravijo, da ne morejo škodovati pacientom.

Poleg zdravja pacientov so zelo občutljivi tudi podatki o pacientih in njihovih posegih. Zavoljo varovanja teh podatkov morajo biti sistemi za prenos zelo kompleksni in dobro kodirani.

Razvoj in trženje zdravstvenih aplikacij je torej zelo odgovorno in zahtevno opravilo. Ob vsakršni nepazljivosti se lahko hitro kaj zalomi, vsaka napaka pa lahko pacienta stane življenja ali povzroči, da osebni podatki pridejo v roke tretji osebi, ki bi jih lahko uporabila pacientu v škodo.

2.4 Aplikacija Računalnik ob postelji – ROP

Program ROP se bo uporabljal v bolnišnicah kot nadomestek terapevtsko temperaturnega lista. To je list velikosti A3, na katerem so zapisani vsi podatki o pacientu. Nanj se vsak dan beležijo npr. podatki o tem, katera zdravila je prejel in katera še mora, razna opažanja, pacientove alergije, diete, rezultati testiranj iz laboratorijev, njegovo stanje, razne osnovne

meritve, kot so pritisk, temperatura, kisik v pljučih … Aplikacija ROP bo vse te podatke prikazala urejeno, pregledno in enostavno za hitro vnašanje v sistem.

2.4.1 Funkcionalnost

Terapevtsko temperaturni list se v bolnišnicah uporablja že dolgo, vendar njegova učinkovitost v današnjem času postaja vprašljiva. S programom ROP bomo namreč iste podatke lahko prikazali urejeno in imeli nadzor nad tem, kdo jih lahko vnaša in pregleduje, pa tudi nad tem, kdo lahko predpisuje razna zdravila in terapije. Glavna funkcionalnost programa bo vnos in pregled podatkov o pacientu.

Program uporablja gradnike (*widget*), zato ga lahko enostavno prilagajamo. Postavitev gradnikov prilagodimo željam posameznih bolnišnic in zdravstvenega osebja. Pomembna funkcionalnost programa je grafični prikaz izvajanja vseh pomembnih nalog, kot so delitev zdravil, menjava infuzije, dieta itd. Sistem nas obvešča, kdaj so te naloge potrebne, poleg tega pa sledi tudi porabi zdravil, da imajo bolnišnice večji nadzor nad njo.

Program ima še veliko možnosti za razširitev, ki se bodo razvijale v skladu z željami strank.

2.4.2 Namen

Namen programa ROP je pridobiti večjo preglednost in sledljivost. Ko bo zdravnik predpisal neko terapijo oz. zdravilo, bo takoj razvidno, kdo je odgovoren za ta predpis, kdo ga mora izvesti in kdaj. Tako bo prišlo do manj napak in bo zato zdravstvena oskrba boljša.

Program bo znal grafično prikazati natančne podatke. Ti so bili do sedaj zelo približni ali pa jih sploh ni bilo. Zdravniki bodo lahko izvedli hitri pregled svojih pacientov kar iz pisarne, zato bodo manj obremenjeni in bodo svoje delo lahko opravljali učinkoviteje. Podobno bodo lahko sestre od daleč preverile, kdaj morajo opraviti kako nalogo po navodilu zdravnika.

2.4.3 Delovanje

Program bo naložen na prenosnih ali tabličnih računalnikih, ki podpirajo tehnologijo zaslonov, občutljivih na dotik. Deloval bo v operacijskih sistemih Windows XP, Windows Vista in Windows 7. Dostop do podatkov bo urejen preko spletnega servisa, ki bo naložen na ločenem strežniku. Potrebna bo uporaba intraneta preko brezžičnega omrežja ali s pomočjo podatkovnega kabla.

Vsaka skupina uporabnikov bo imela svoje pravice za upravljanje z aplikacijo, kot so na primer branje, pisanje in urejanje podatkov. Imeli bomo tri glavne skupine uporabnikov: medicinske sestre, zdravnike in administratorje. Slednji bodo imeli nalogo izdelati postavitev gradnikov glede na potrebe ustanove, v kateri bo deloval program.

3 Razvoj grafičnega uporabniškega vmesnika

Uporabniški vmesnik je zelo pomemben in zapleten del računalniškega programa, zato je zelo pomembno, da ga dobro načrtujemo in razvijemo. Najpomembnejši del razvoja je učinkovito zbiranje uporabniških zahtev in njihova implementacija.

3.1 Standardni proces razvoja uporabniškega vmesnika

Proces razvoja oziroma izdelave uporabniškega vmesnika je za vse aplikacije, spletne strani, računalnike, mobilne komunikacijske naprave, itd. podoben ali celo enak. Cilj je izdelati enostaven, funkcionalen in učinkovit uporabniški vmesnik v smislu izvedbe uporabniških nalog – temu pogosto pravimo »k uporabniku usmerjeno načrtovanje«. Pri načrtovanju moramo skrbeti za ravnotežje med tehnično funkcionalnostjo in vidnimi elementi, edino tako nam bo uspelo izdelati uporabniški vmesnik, ki ni samo funkcionalen, ampak tudi uporaben in prilagodljiv [18] .

Poznamo več faz in procesov v razvoju dobrega uporabniškega vmesnika:

- zbiranje zahtev – naredimo seznam funkcionalnosti, ki jih mora program podpirati, in zahtev uporabnikov programa
- analiza uporabnikov – analiziramo, kateri in kakšni uporabniki bodo uporabljali program, in ga primerno prilagodimo; pogosto si zastavimo naslednja vprašanja:
    - kaj naj program dela, da bo zadovoljil uporabnikove želje?
    - kako se bo program prilagodil uporabnikovemu delovnemu procesu?
    - kako zahtevna bo uporaba programa za konkretne uporabnike?
    - kakšen izgled in občutek privlači uporabnika?
- informacijska shema – odločimo se, kako bomo pregledno prikazali podatke
- test uporabnosti – končnemu uporabniku prikažemo prototip in ga prosimo, da na glas govori o svojih izkušnjah z njim (temu rečemo »protokol misli na glas«)
- oblikovanje in izdelava grafičnih elementov – končna oblika grafičnih elementov se pogosto določi na podlagi testa uporabnosti, ki pokaže, kakšen izgled privlači končnega uporabnika; možna je izdelava večjega števila predlog, za vsakega uporabnika posebej [18]

Pri našem razvoju uporabniškega vmesnika za zaslone na dotik smo se okvirno držali zgornjih alinej. Največji poudarek smo dali zbiranju uporabniških zahtev in testiranju uporabnosti pri končnem uporabniku. Ta dva procesa sta zelo pomembna, saj sta neposredno povezana z osebo, ki bo izdelek uporabljala dnevno na svojem delovnem mestu. Rezultat tega bo aplikacija, ki bo prijetna, funkcionalna in hitra za uporabo. Če končna aplikacija ni prijetna za vsakodnevno uporabo, imajo uporabniki naporno delo, temu pa se želimo izogniti.

3.2 Uporabniške zahteve

Razvoj uporabniškega vmesnika je zelo pomemben korak v izdelavi programske opreme za masovno uporabo. Celotna uporabnost programa je namreč odvisna od izdelave dobrega uporabniškega vmesnika, tj. če je vmesnik slab, bo tudi uporaba programa otežena in

neprijetna. Hitrost in enostavnost sta v zdravstvenih aplikacijah zelo pomembni, zato programska oprema ne sme povzročati nepotrebnih napak in časovnih zaostankov.

3.2.1 Pridobivanje uporabniških zahtev

Uporabniške zahteve lahko pridobimo na več načinov. Najpogostejša načina sta uporaba različnih vprašalnikov, ki so hitri za izpolnitev in enostavni za analizo, ter organizacija uporabniških delavnic, ki so zahtevnejše za izvedbo in analizo, ampak dajo natančnejše končne rezultate. Slednje smo uporabili tudi v našem podjetju.

Na uporabniške delavnice povabimo uporabnike, ki bodo uporabljali našo aplikacijo. S prisotnimi se pogovarjamo in tako nadgrajujemo funkcionalnost in uporabnost programa. Na vsaki delavnici obdelamo posamezne sklope vprašanj, ki se tičejo uporabniškega vmesnika. Odgovori na ta vprašanja nam pomagajo izboljšati način varovanja pomembnih podatkov, sledenje njihovim spremembam, pa tudi izgled, funkcionalnost, uporabnost in dinamičnost uporabniškega vmesnika. Poleg tega nam pomagajo ugotoviti, kakšne možnosti nadgradnje programa potrebujemo, katere funkcije so pomembne za kako ustanovo itn.

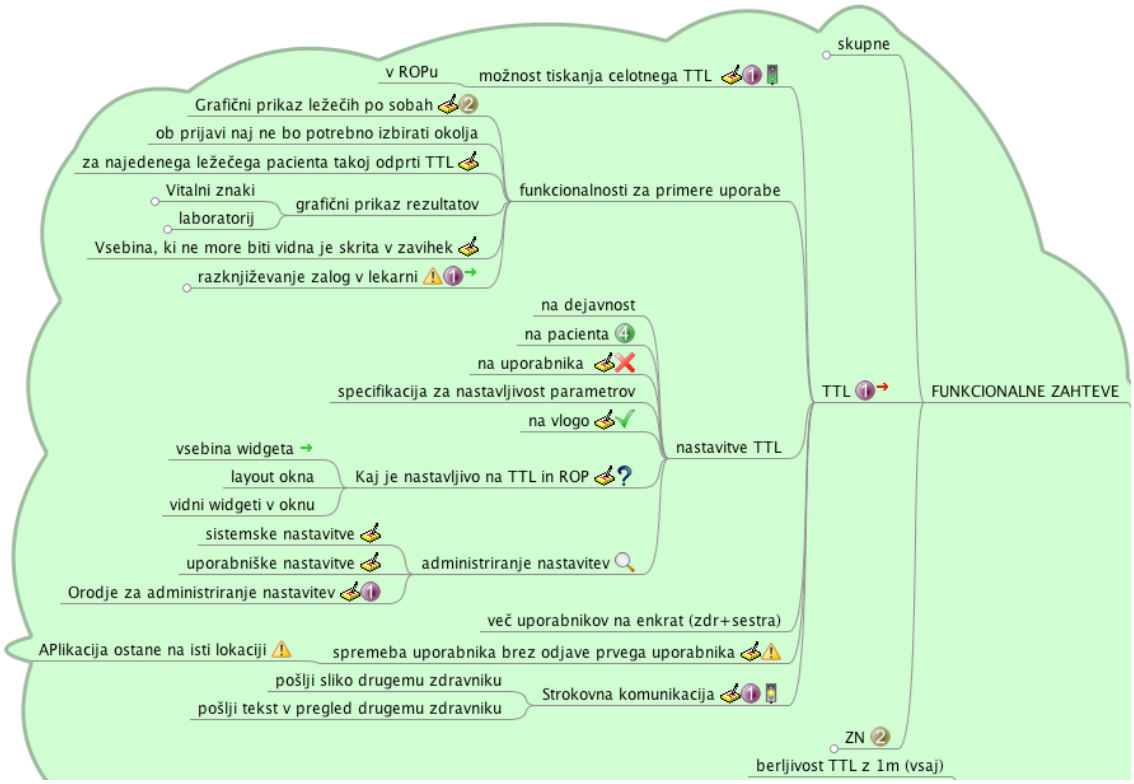
Na vsaki delavnici prisotni prejmejo nalogo, ki jo rešijo in predstavijo na naslednji delavnici.

Pridobivanje uporabniških zahtev na tak način je možno tudi pri drugih uporabniških vmesnikih.

3.2.2 Rezultati delavnic

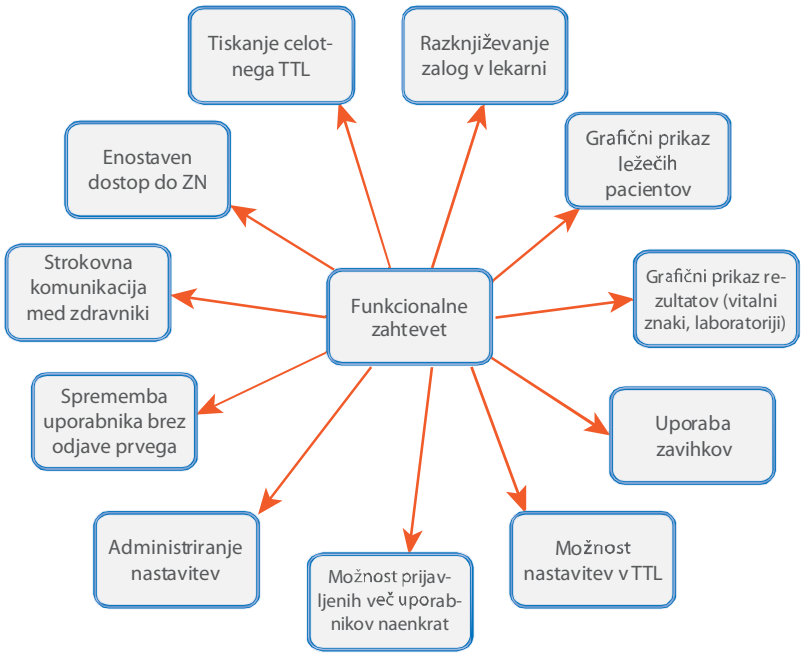
Rezultate posameznih delavnic zberemo in analiziramo. Iz njih poskušamo izluščiti najpomembnejše zahteve in ideje, ki jih bomo implementirali v naš uporabniški vmesnik. Na sliki 3 vidimo primer rezultata uporabniških delavnic v obliki miselnega vzorca, in sicer za program ROP [9]. Glavni veji zahtev sta: funkcionalne zahteve (slika 4) in nefunkcionalne zahteve (slika 5).

Pri razvoju programa ROP smo imeli že po prvih nekaj delavnicah na voljo dovolj podatkov o najpomembnejših funkcijah, ki naj bi jih program podpiral, da zadovolji večino potreb zdravstvenih ustanov. Ugotovili smo tudi, da se bodo morali uporabniki prilagoditi uporabi računalnika, kajti do sedaj so uporabljali samo papir. Največja razlika med tema dvema medijema je, da je papir bolj fleksibilen in omogoča hitrejše in enostavnejše vnašanje podatkov. Slabost papirja pa je težje preverjanje pravilnosti podatkov in slabša organiziranost.

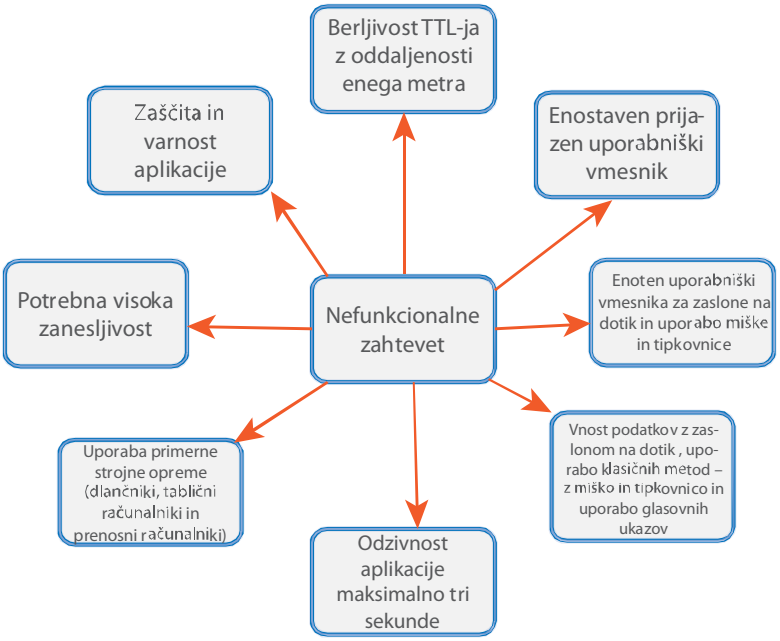


Slika 3: Miselni vzorec funkcionalnih uporabniških zahtev uporabniškega vmesnika za program ROP, ki je bil sestavljen v programu FreeMind

Uporabniške zahteve in druge pomembne informacije smo zbrali v obliki miselnega vzorca v programu FreeMind (slika 3). Uporabniške zahteve smo razdelili v dve skupini: funkcionalne in nefunkcionalne zahteve.



Slika 4: Funkcionalne zahteve



Slika 5: Nefunkcionalne zahteve

3.3 Izdelava uporabniškega vmesnika po uporabniških zahtevah

Po popisu uporabniških zahtev za uporabniški vmesnik smo se lotili izdelave osnovnih gradnikov in njihove postavitve. Pričeli smo z izbiro primernih barv, ki so zelo pomemben dejavnik pri uporabniškem vmesniku.

Grafične elemente smo izdelali v programu Adobe Illustrator, ki je program za izdelavo vektorske grafike. Zanj smo se odločili zato, ker je eden boljših programov za vektorsko grafiko na trgu.

Za osnovo grafičnih gradnikov smo izbrali vektorsko grafiko, saj je ta prilagodljiva in jo je mogoče neposredno prevesti v format, ki ga lahko kasneje programerji uporabijo v svojem orodju. Tako se izognemo večkratnemu delu.

Po končani izdelavi glavnih gradnikov (npr. vnosno polje, spustni seznam, gumbi, okna, drsniki, modalna okna, ikone itd.) smo se lotili izdelave pogledov, ki so bili zapisani v uporabniških zahtevah. Pri izdelavi uporabniškega vmesnika je potrebno paziti na ravnovesje med uporabnostjo, prilagodljivostjo in estetiko, saj so ti kriteriji bistvenega pomena za izdelavo kvalitetnega uporabniškega vmesnika.

Pri izdelavi uporabniškega vmesnika je pomembna stalna komunikacija z ljudmi, ki so zadolženi za vsebinski del, in ljudmi, ki so zadolženi za arhitekturo in funkcionalnost izdelka.

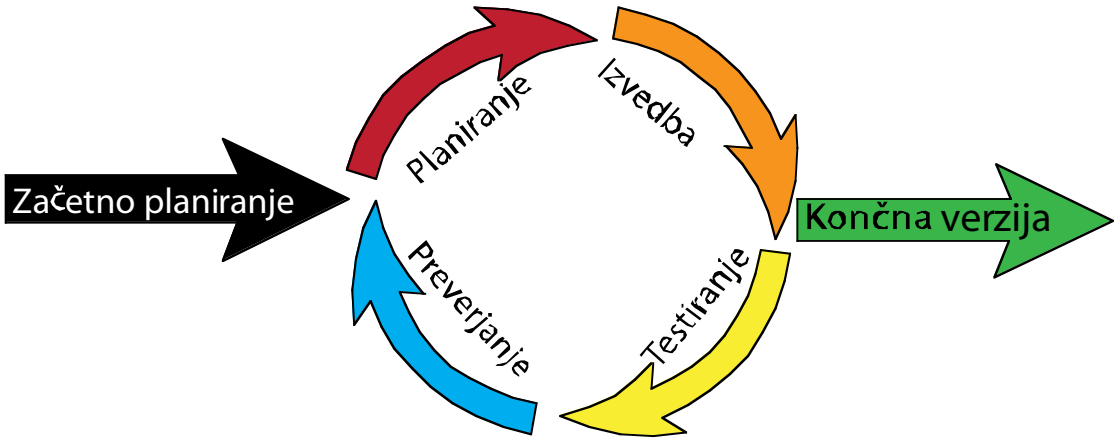
Komunikacija z ekipo, zadolženo za vsebinski del, je pomembna predvsem za boljši način prikaza in vnosa podatkov, saj ta ekipa ve, kako moramo vsebino prikazati, da se bodo uporabniki dobro počutili.

Nasveti ekipe, ki skrbi za arhitekturo in funkcionalnost, pripomorejo k odločitvam, kaj naj bi uporabniški vmesnik sploh omogočal oz. kakšne funkcionalnosti naj vnesemo vanj.

Zahvaljujoč tesnemu sodelovanju z obema ekipama smo prišli do prve različice uporabniškega vmesnika, ki pa je bila še daleč od končne oblike, saj specifikacija še ni bila končana.

3.3.1 Iterativni razvoj

Za uporabniški vmesnik smo uporabili iterativni način razvoja [8]. Tak razvoj se prične z začetnim načrtovanjem in zaključi s končno različico programa. Med tema dvema točkama razvoja pa potekajo iteracije načrtovanja, izvedbe, testiranja in preverjanja. Slika 6 prikazuje shematičen potek iterativnega razvoja. Razvoj torej poteka v manjših smiselno razbitih enotah, ki jih izvedemo v obliki prej omenjenega cikla in tako dopolnjujemo program. Na ta način hitro odkrijemo svoje napake in se sproti učimo iz že nastalega dela programa. Za dober nadzor nad takim razvojem je dobro imeti pripravljen seznam akcij in funkcij, ki jih je še potrebno implementirati v kakem izmed ciklov.



Slika 6: Model iterativnega razvoja

Tak način razvoja uporabniškega vmesnika omogoča ponovno snovanje določenih delov aplikacije, ki se mogoče ne skladajo s končno različico programa in jih je potrebno popraviti pred zaključkom implementacije. Cilj vsake iteracije je enostavnost, modularnost in možnost ponovne zasnove.

3.3.2 Izdelava specifikacije za uporabniški vmesnik

Specifikacija je obvezen dokument pri razvoju dobre in stabilne aplikacije. Napisati jo je treba zato, da zunanji izvajalci vidijo in razumejo, kakšno obliko in funkcionalnost mora imeti uporabniški vmesnik, ki ga bodo izdelali. Uporabili smo dve vrsti specifikacije. Prva je bila napisana v programu Enterprise Architect in je predstavila podrobno funkcionalnost vsakega gumba na obrazcu in njegovo povezavo z arhitekturo programa. Drugi dokument pa je bil

napisan v programu Microsoft Word in je opisoval izključno vizualne elemente in obrazce. Šele na podlagi tega dokumenta so lahko programerji zgradili uporabniški vmesnik.

Ko smo končali z risanjem prve različice osnovnih gradnikov in pogledov, smo v posebnem dokumentu, ki mu pravimo »UI specifikacija« in je namenjen izključno opisu uporabniškega vmesnika, za vsak element temeljito opisali obnašanje in parametre. Zapisali smo, katere barve vsebuje, kakšna je njegova velikost, kako se odziva na različne resolucije in velikosti zaslonov ... Opisali smo tudi čas, ki je potreben, da se aplikacija odzove na uporabnikovo akcijo, obnašanje desnega in levega miškinega gumba, dvojnega klika, lebdenja z miško … Opisali smo tudi povezavo med akcijami miške in akcijami na zaslonih na dotik: dvojnemu kliku na miški smo npr. priredili enojni dotik zaslona. Predpisali smo tudi minimalne sistemske zahteve – npr. resolucijo in gostoto pik zaslona. Na podlagi teh podatkov so lahko programerji začeli z razvojem osnovnih gradnikov, ki smo jih kasneje uporabili v aplikaciji.

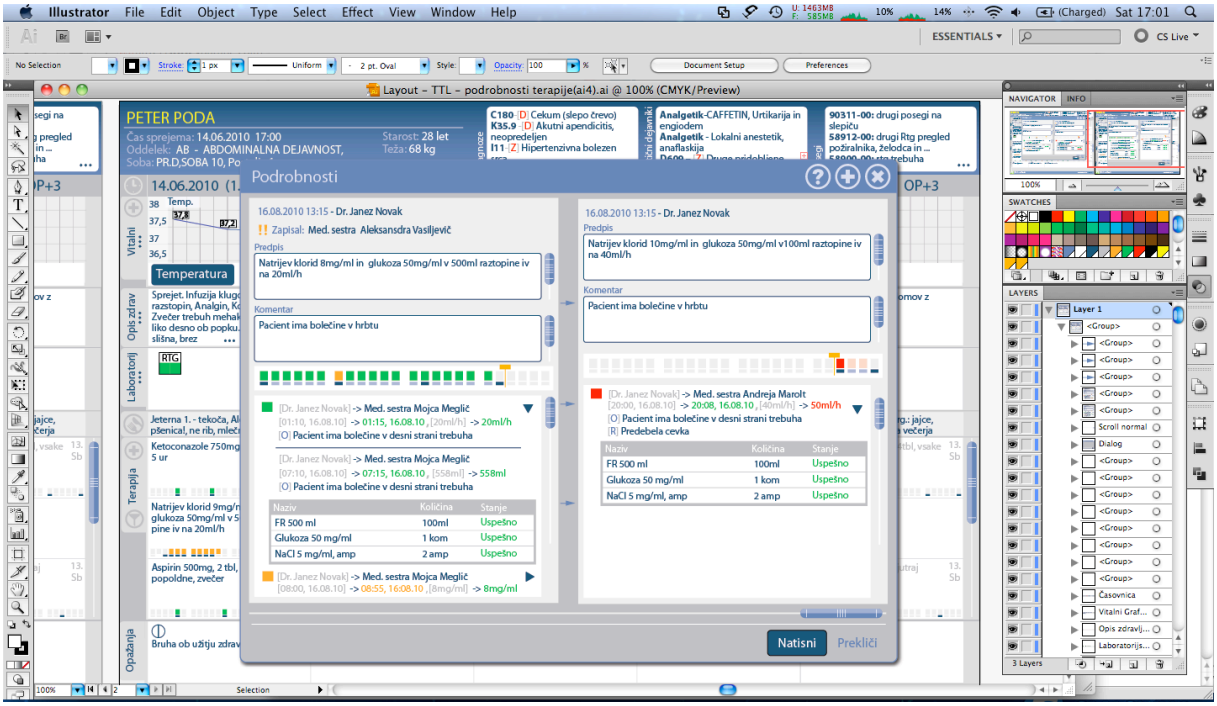
Dokumente smo sproti dopolnjevali in popravljali, saj se je nekaj problemov in sprememb pojavilo že med samim razvojem programa in uporabniškega vmesnika zanj. Idealno bi bilo sicer najprej napisati specifikacijo in na podlagi te narediti končni program, a v praksi se pogosto dogajajo nepričakovane spremembe.

3.4 Izdelava prototipa

Prototip je približek končnega izdelka v polni ali pomanjšani velikosti, izdelan za namene preučevanja, testiranja, učenja, prikazovanja in promocije [10]. Prototip lahko implementira tudi nekatere funkcije končnega izdelka.

Prototipi pri razvoju programske opreme, predvsem pri razvoju uporabniških vmesnikov, pogosto izgledajo kot končni izdelek, ampak so brez funkcionalnosti. Lahko so narisani s svinčnikom na papir, s pomočjo posebnih orodij, ki vsebujejo vnaprej izrisane komponente, ali pa v obliki računalniških slik, ki že predstavljajo končno obliko uporabniškega vmesnika za dano aplikacijo.

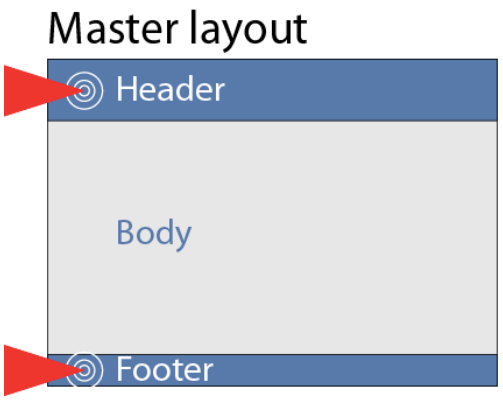
Prototip uporabniškega vmesnika smo se odločili narisati v programu Adobe Illustrator, in sicer zaradi možnosti enostavnega urejanja vsebine, prilagodljivosti in prenosljivosti. Slika 7 prikazuje primer prototipa osnovne strani aplikacije s prikazanim temperaturnim listom in podrobnostmi o prejetih zdravilih.



Slika 7: Izdelava prototipa v programu Adobe Illustrator

3.4.1 Izdelava osnovnih gradnikov

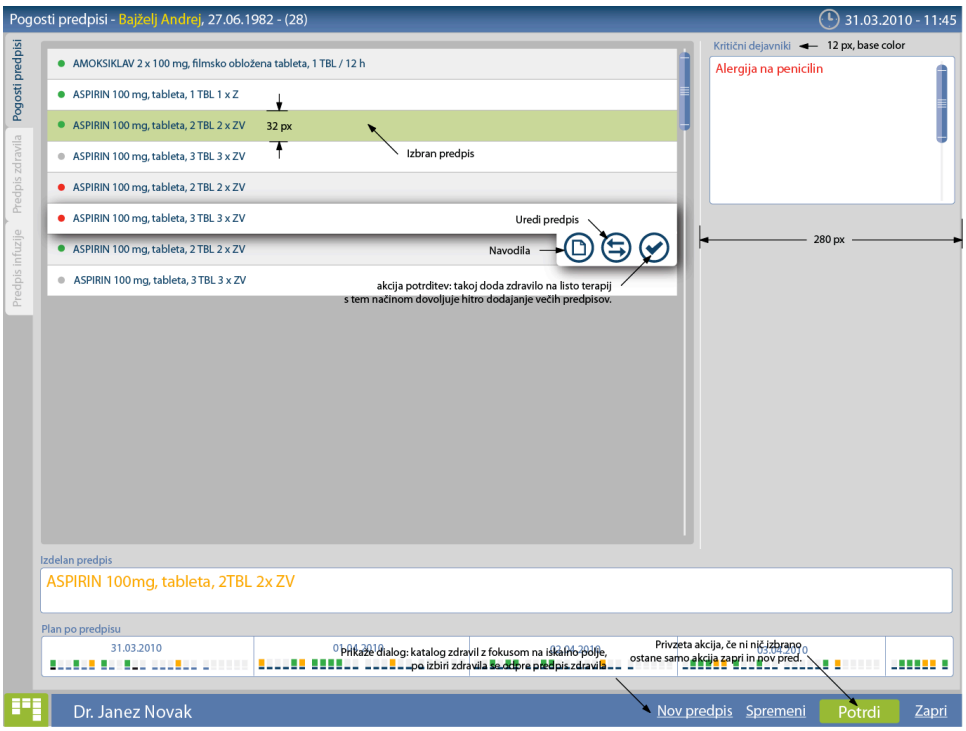
Delo v Illustratorju smo začeli z izdelavo osnovnih gradnikov in pogledov (primer take osnovne postavitve je prikazan na sliki 8), iz katerih smo kasneje sestavili zaslonske slike, ki jih je program potreboval. Vsem gradnikom smo določili skupne globalne barve, saj smo tako lahko vsem naenkrat spreminjali vrednosti. Ta funkcionalnost nam je prišla tekom razvoja uporabniškega vmesnika večkrat prav, saj je nekatere stvari mogoče videti šele, ko uporabniški vmesnik pogledamo kot celoto.



Slika 8: Prikaz osnovne postavitve uporabniškega vmesnika

3.4.2 Izdelava posameznih oken

Po končani izdelavi osnovnih gradnikov smo izdelali poglede (slika 9) za vse vnaprej specificirane zahteve in jih nato opisali v dokumentaciji za uporabniški vmesnik. Tekom izdelave uporabniškega vmesnika smo se pogosto vračali k urejanju že izdelanih pogledov. Sama izdelava osnovnih gradnikov ni bila tako zahtevna kot njihova postavitev v okno, kjer naj bi služili vsak svoji funkciji.



Slika 9: Končni izgled okna »Pogosti predpisi« z dodatnimi opisi in napotki programerjem

4 Test uporabnosti

Test uporabnosti se izvaja zato, da dobimo podatke o odzivu uporabnikov na trenutni izgled in funkcionalnost uporabniškega vmesnika, še preden gre v razvoj. Popravek na sliki je veliko hitrejši in cenejši kot popravki v programski kodi, zato smo izvedli test uporabnosti večkrat in zelo temeljito z vsemi poglavitnimi uporabniki [14, 6, 5].

4.1 Test uporabnosti

Test uporabnosti je tehnika testiranja prototipa s pomočjo uporabnikov [17]. Na ta način preizkusimo uporabniški vmesnik programa pri končnih uporabnikih. Glavni cilj testa uporabnosti je ugotoviti, če izdelani program oz. uporabniški vmesnik implementira vse funkcije, ki jih potrebuje.

Tehnika testa uporabnosti je tehnika »črne škatle«. To pomeni, da uporabnik, ki testira, pripravi veljavne in neveljavne vhodne podatke ter vnaprej določi želeni rezultat teh podatkov. Uporabnik torej ve, kakšne izhodne podatke mora dobiti. Cilj testa je opazovanje

uporabnika med interakcijo s testiranim prototipom. S testom uporabnosti dobimo podatke za naslednja štiri področja:

- učinkovitost – koliko časa in korakov je bilo potrebno za izvedbo neke osnovne akcije
- natančnost – koliko napak so naredili uporabniki
- spomin – kako dobro se uporabnik spomni delovanja programa po določenem času
- čustveni odziv – kako se uporabnik počuti, ko zaključi neko akcijo

Poglavitna značilnost testa je, da uporabnik sam brez večje pomoči moderatorjev uporablja prototip za izvedbo osnovnih funkcij, ki bodo na voljo v končni različici programa. Najbolj priljubljene metode za izvajanje testa uporabnosti so: protokol misli na glas, učenje z raziskovanjem in sledenje očem [12]. Pri našem testu smo uporabili t.i. "protokol misli na glas". Pri tem načinu prosimo uporabnika, naj na glas komentira in razmišlja o uporabi prototipa. Vse komentarje si moderatorji zapišejo in jih upoštevajo pri razvoju prototipa in pri naslednjih testiranjih uporabnosti.

4.2 Scenariji

Pred izvedbo testa uporabnosti moramo najprej sestaviti ključna vprašanja, na katera bi radi dobili zadovoljive odgovore. V našem primeru so vprašanja zajemala vsa področja uporabe aplikacije. Predvsem smo se osredotočili na uporabnost, funkcionalnost in smiselnost posameznih dnevnih opravil, ki jih bodo izvajali uporabniki. Vprašanja smo razdelili na smiselne sklope. Slika 10 prikazuje primer vprašanj, ki smo jih zastavili uporabniku za sklop »Dekurzus« [12].

Dekurzus=Opis zdravljenja (IP)	- Ali v osnovnem pogledu (base view) uporabnik vidi dovolj informacij? Ali jih potrebuje? Kdaj? - Ali je PopUp view primeren? (mogoče bi raje celo okno samo za to ali kaj drugega…) - Ali so v PopUp view prikazane vse informacije, ki jih uporabnik potrebuje? Kdaj sploh potrebuje kronološki pregled vseh zapisov? - Ali je dodajanje novega zapisa dovolj enostavno? (Ali si res to želijo početi skozi okno za prikaz vseh dekurzusov? Mogoče bi raje nekakšen neposreden vnos na TTL? Ali pač naj bo v PopUp view vedno že tako na razpolago dodajanje novega zapisa?) - Ali je prednastavljen čas/datum vnosa uporaben? - Ali lastni zapisi morajo biti izpostavljeni?

Slika 10: Primer zastavljenih vprašanj za sklop »Dekurzus« v programu ROP

Ko smo sestavili vsa pomembna vprašanja, smo se lotili načrtovanja scenarijev za vsako med njimi. Za vsako vprašanje smo si izmislili scenarij, iz katerega smo želeli dobiti odgovor nanj.

Vsak scenarij (primer enega scenarija iz [15] je na sliki 11) je bil namenjen določenemu sklopu uporabnikov, ki bodo uporabljali naš izdelek. Scenarij je sosledje dogodkov, katerega rezultat je zaključena akcija. Scenarij je sestavljen iz konkretnih akcij uporabnika, kot so na primer:

- izbere oddelek

- zamenja oddelek
- preklopi na okno iskalnika
- odpre pacientov TTL
- se prijavi v okolje

Posamezne testne scenarije za naše uporabnike smo najprej opisali s seznami, kot je ta zgoraj. Poleg seznama vseh korakov, ki so potrebni za izvedbo scenarija, smo napisali tudi določene kriterije. Če je rezultat posameznega koraka pokazal, da scenarij ustreza našim kriterijem, je bil scenarij sprejemljiv in ni bilo potrebe po spremembah.

Scenarij	Koraki	Kriteriji
U7. Urejanje dekurzusov	1. Za izbranega pacienta za dva dni nazaj poišče/pogleda "Dekurzus" 2. Na tri pikice pogleda bolj detajlen pregled dekurzusov (popup) 3. Izbere dodajanje novega zapisa 4. Vnese tekst in spremeni uro na katero se zapis nanaša (na 17:00) 5. Shrani spremembe 6. Gre nazaj na TTL 7. Opazi napako v zadnjem zapisu (vidnem v pregledu dekurzusov) in prikliče popup z detajli 8. Za želeni (zadnji) zapis prikliče urejevalnik. 9. Spremeni nekaj v tekstu. 10. Shrani in gre nazaj na TTL	1. Dodajanje novega zapisa je dovolj enostavno in na primernem mestu (okno za prikaz vseh dekurzusov) 2. Ni potrebe po neposrednem vnosu na TTL. 3. Ni treba da je v PopUp view vedno prisotno/vidno okno za dodajanje novega zapisa. 4. Prednastavljen čas/datum vnosa ni moteč.

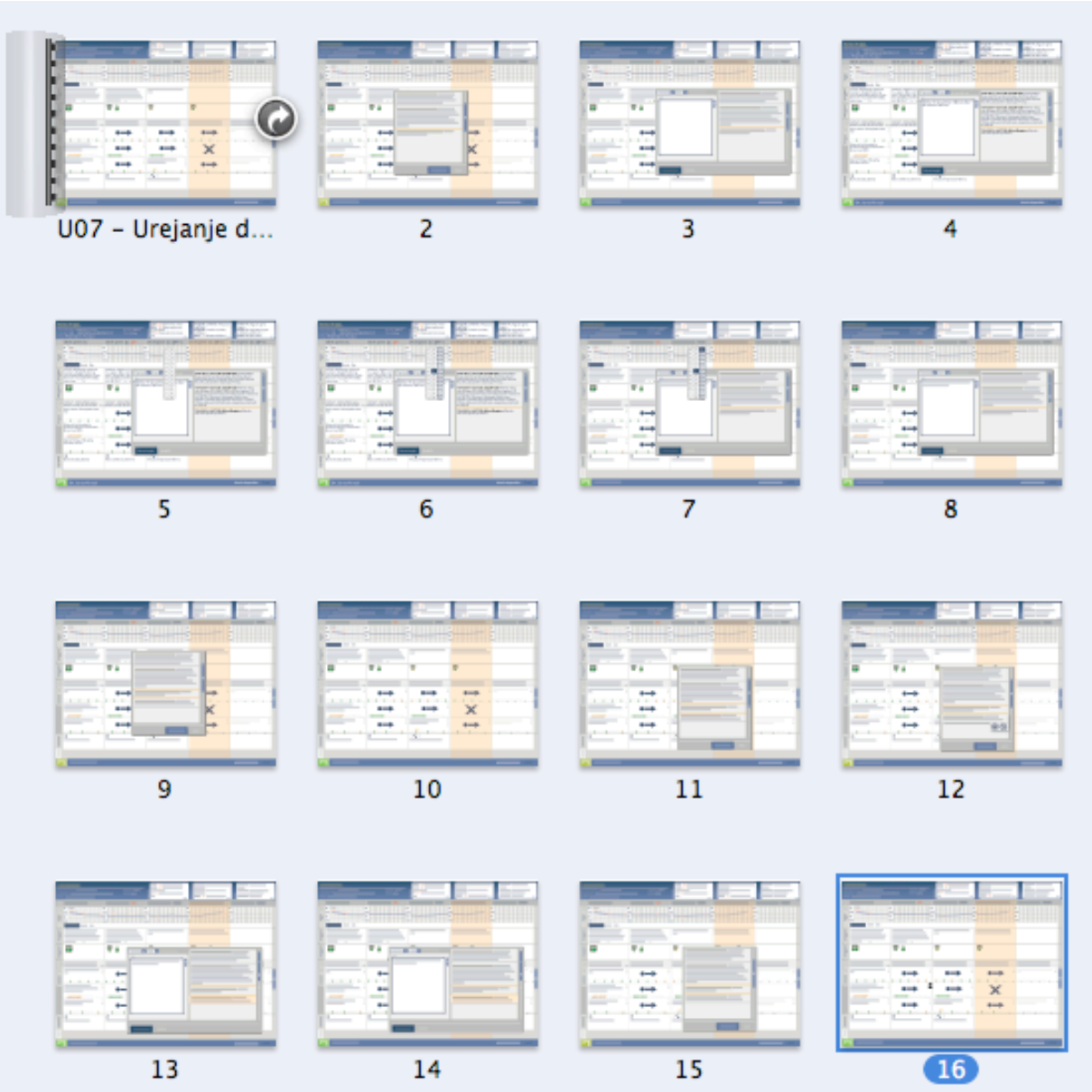
Slika 11: Primer scenarija za urejanje dekurzusov z vsemi koraki in kriteriji zanj

4.2.1 Risanje scenarijev

Ko smo končali z načrtovanjem scenarijev, smo se lotili njihovega risanja. Scenarije smo izrisali v istem programu kot osnovne gradnike (Adobe Illustrator CS5). Pri izdelavi smo sledili posameznim korakom, ki so bili zapisani v načrtu za dani scenarij. Z izrisanimi scenariji smo želeli upodobiti končni izgled programa, kot ga bodo uporabljali zdravniki in medicinske sestre. Po končanem izrisu scenarija smo slike zložili skupaj v PDF dokument, ki je obsegal tudi do 20 strani. Primer enega izmed scenarijev, sestavljenega iz 16 slik (zaslonskih posnetkov), je prikazan na sliki 12. Vsaka slika prikazuje program, odprt v celozaslonskem načinu. Ob kliku nanjo se prikaže naslednja slika v vrsti – na ta način simuliramo klike in odzive aplikacije.

Med risanjem scenarijev je bila potrebna stalna povezava s sodelavci, zadolženimi za vsebinski del naloge, ki so edini poznali delovanje zdravnikov in sester.

Za format PDF smo se odločili zato, ker podpira vektorsko grafiko in je zato najbolj primeren za prikaz na različnih zaslonih in pri različnih resolucijah. PDF je tudi zelo prenosljiv format, saj si ga lahko ogledamo v vseh računalniških okoljih in v mnogih aplikacijah. Zelo priročno je tudi dejstvo, da Adobe Illustrator pozna format PDF in smo lahko nekatere spremembe hitro popravili kar v tem formatu.



Slika 12: Izrisani scenarij za »Urejanje dekurzusa« po posameznih korakih

4.3 Proces testiranja uporabnosti pri uporabniku

Po končanem izrisu scenarijev smo izvedli test uporabnosti pri naših strankah. Testiranje je potekalo v dveh dneh po 3 ure, in sicer z uporabo miške in tipkovnice. Za testiranje smo potrebovali le računalnik z naloženim programom za branje datotek PDF. Poleg uporabnika sta bila prisotna še moderator in moderatorjev pomočnik, ki sta si zapisovala opažanja in pomagala uporabniku. Uporabnik je imel najprej uvodno izobraževanje iz uporabe aplikacije, nato pa je dobil naloge, ki jih je moral izpeljati do konca. Pri tem je uporabljal vnaprej pripravljen prototip aplikacije v obliki PDF. Uporabnik je lahko zastavljal vprašanja, in če na kakega med njimi nismo znali odgovoriti, smo ga zapisali in poskušali nanj odgovoriti do naslednjega dne. Zabeležili smo tudi vse komentarje uporabnikov na predstavljene scenarije [11].

Testiranje je potekalo na naslednji način (uporabnika smo pri tem vodili skozi vsak scenarij posebej):

1. Moderator prebere korak.
2. Uporabnika prosimo, da med simulacijo uporabe programa na glas razmišlja.
3. Moderator glede na simulirane akcije uporabnika preklaplja slike v datoteki PDF.
4. Moderator in njegov pomočnik opazujeta, kako uporabnik rešuje situacijo, ter si zapisujeta njegova opažanja in komentarje.
5. Ob koncu vsakega scenarija uporabnika prosimo, da poda komentar nanj – če je rešitev primerna. Moderator si komentar zabeleži.
6. Ob koncu testiranja uporabnika prosimo, da komentira aplikacijo kot celoto in izvedbo testiranja. Moderator si komentar zabeleži.
7. Po prvem dnevu testiranja se na podlagi prejetih komentarjev pripravimo na naslednji dan testiranja.

Zbrane podatke in komentarje smo kasneje analizirali in upoštevali pri izdelavi aplikacije.

4.4 Rezultati testiranja

Testiranje nam je dalo predvsem vpogled v ideje in želje uporabnikov, kako bi lahko izboljšali uporabnost posameznega sklopa programa. Videnje uporabe programa skozi oči bodočega uporabnika močno pripomore k razvoju boljše aplikacije, saj jo lahko na podlagi komentarjev prilagodimo njemu v prid. Prvi del testa je tekel z uporabo miške in tipkovnice, drugi del pa z uporabo zaslona na dotik. Uporabniki so imeli manjše težave z uporabo zaslonov na dotik, ker so jih do sedaj redko uporabljali. Pravega testiranja za zaslone na dotik nismo mogli opraviti, saj je testiranje potekalo z zaslonskimi slikami, združenimi v en dokument PDF.

Rezultate smo zbrali v poročilu testiranja [13] in jih kasneje analizirali. Mnogo idej in komentarjev testiranih smo tudi vpeljali v naš izdelek, ki je bil tedaj še vedno v prvi fazi razvoja [19].

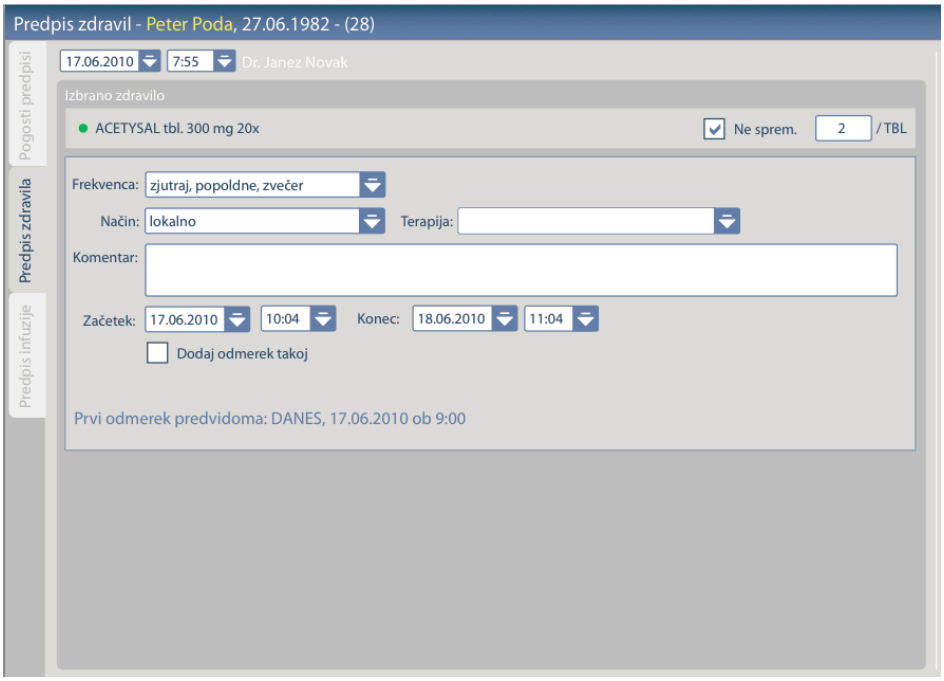
4.5 Popravki uporabniškega vmesnika na podlagi pridobljenih podatkov

Po analizi rezultatov testiranja smo se lotili popravkov na uporabniškem vmesniku in funkcionalnosti aplikacije. Z uporabniškim vmesnikom so bili uporabniki na splošno zadovoljni. Najbolj jih je motila odsotnost nekaterih parametrov, ki so jih potrebovali za izvedbo svojega dela. To smo ustrezno popravili.

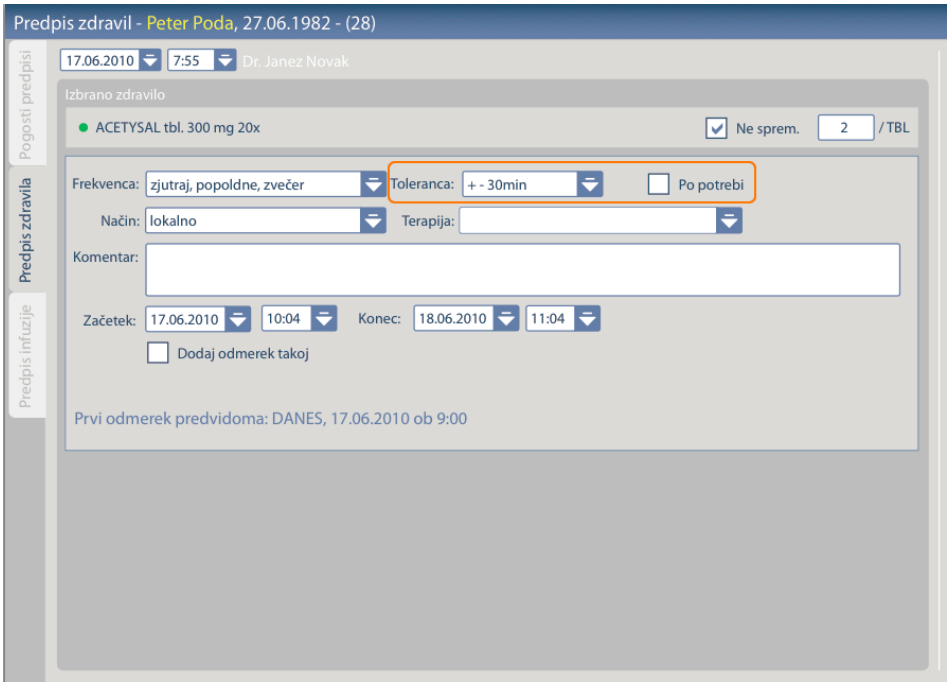
Dobili smo tudi veliko idej in pripomb glede stvari, ki bi jih uporabniki želeli imeti v naslednji različici programa. Mnogih izmed teh idej smo se spomnili že sami, ampak so bile namenjene implementaciji šele kasneje v razvoju.

4.5.1 Primer popravka

Med testiranjem uporabniškega vmesnika so uporabniki pri zavihku za predpis zdravila (prikazan na sliki 13) pripomnili, da potrebujejo možnost izbire časovne tolerance, ki je v prvi različici še nismo imeli. Poleg tega so želeli imeti tudi možnost izbire, da je predpisano zdravilo dozirano po potrebi ali pa samo po naročilu zdravnika. Vmesnik smo dopolnili v skladu z željami uporabnikov (dopolnjena različica je na sliki 14).



Slika 13: Zaslonska slika uporabniškega vmesnika izrisanega za test uporabnosti



Slika 14: Uporabniški vmesnik z vnesenim popravkom po želji uporabnikov – dodana možnost izbire časovne tolerance in možnost izbire atributa »Po potrebi«

Velik del popravkov se je nanašal na manjkajoča ali odvečna polja pri vnosu ali prikazu podatkov. Za te popravke so se uporabniki odločili šele, ko so aplikacijo videli in uporabili v simuliranih okoliščinah. Brez testa uporabnosti bi morali popravke v programsko kodo implementirati naknadno, kar bi predstavljalo veliko večji strošek. Popravkov nismo izvedli zgolj na uporabniškem vmesniku, ampak tudi na strani funkcionalnosti. Ta je opisana v ločenem dokumentu, ki je namenjen programerjem aplikacije. Ob vsaki spremembi uporabniškega vmesnika je namreč potrebno urediti tudi funkcije, ki se odzivajo na popravljeni element. Vse popravke na uporabniškem vmesniku smo opravili v kratkem času in brez večjih težav. Funkcionalne zahteve so potrebovale nekoliko več časa, saj je bilo vsako med njimi potrebno dobro pretehtati in premisliti, kako se mora izvajati, da bodo vse funkcionalnosti, povezane z njo, pravilno delovale.

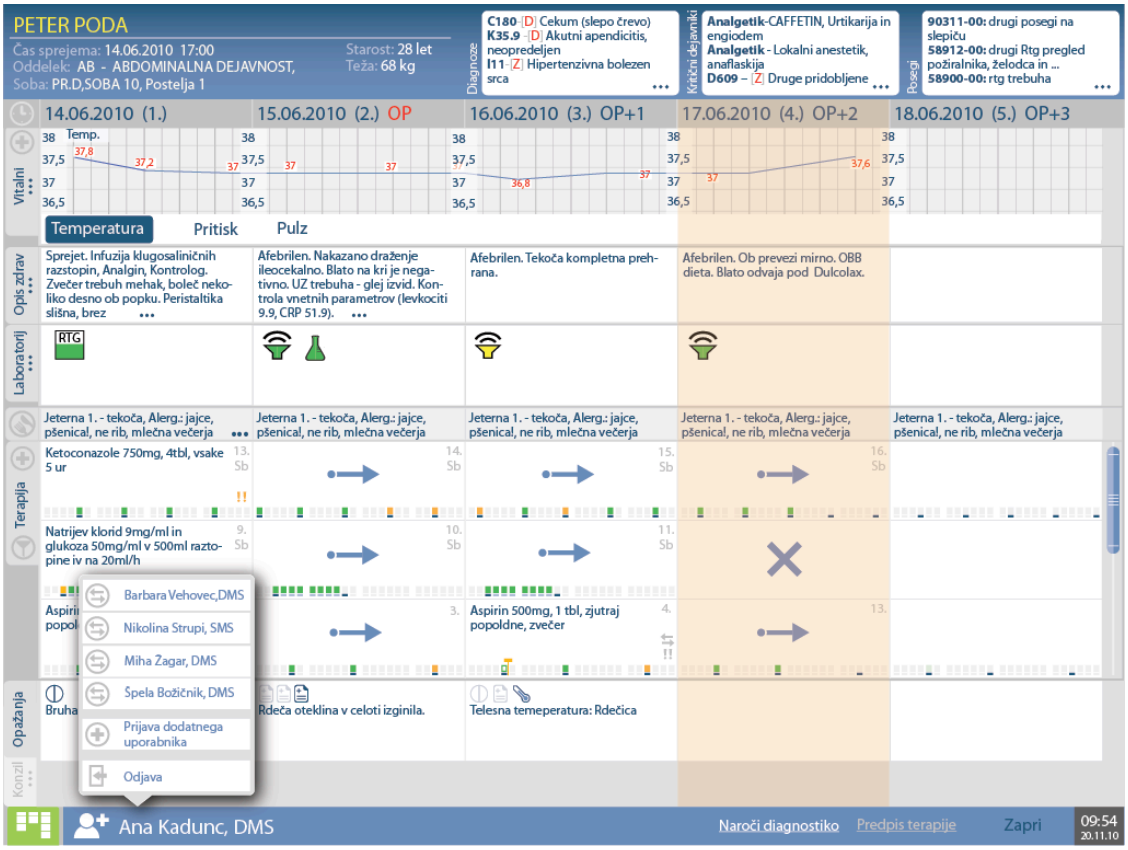
Po vnosu vseh popravkov lahko ponovno izvedemo test uporabnosti pri uporabnikih. Večkrat kot ponovimo ta cikel, manjša je verjetnost, da bodo potrebni popravki v kasnejših fazah, ki jih je težje in dražje izpeljati.

4.6 Razvoj izdelka

Ko smo opravili vse popravke na uporabniškem vmesniku, funkcionalnosti in v zalednih sistemih, smo specifikacijo in druge dokumente predali v razvoj, kjer so se lotili izdelave programa na iterativen način, podobno kot pri našem razvoju prototipa uporabniškega vmesnika. Med programerji in analitiki je potekala stalna komunikacija, saj vseh podrobnosti ni bilo mogoče zapisati v razvojne dokumente programa.

Med razvojem smo naleteli na težave z uporabniškim vmesnikom, saj nekaterih zahtev ni bilo možno implementirati tako, kot je bilo zapisano v dokumentu. Zato smo morali nekatere komponente uporabniškega vmesnika ponovno prilagajati. Imeli smo nekaj težav s kombiniranjem klasičnega načina uporabe (z miško in tipkovnico) in načina z zaslonom na dotik. Problem je tičal v velikosti posameznih gradnikov in njihovih odzivnih točk. Gradniki morajo biti namreč za uporabo z zasloni na dotik vsaj dvakrat večji, saj smo s prstom veliko manj natančni kot z miško. Spustni seznam ima npr. odzivno točko na gumbu s puščico, kar miški ne predstavlja nobenih težav, če želimo isti element aktivirati s prstom, pa se moramo precej bolj potruditi, saj s prstom nismo dovolj natančni in pogosto zgrešimo aktivni gumb elementa. Zato smo morali razširiti aktivni del čez cel element. Uporaba s prstom je tako olajšana in posledično je uporabnik manj obremenjen z neodzivnostjo oz. nenatančnostjo elementa.

Dobro premišljen in testiran uporabniški vmesnik za dano aplikacijo je zelo pomemben, saj se aplikacije s slabo uporabnostjo ne morajo kosati s ponudbo na današnjemu trgu, kjer je uporabnost zelo visoko cenjena. Poleg uporabnosti je vedno bolj pomembna tudi vizualna podoba aplikacije. Uporabniki raje uporabljajo aplikacije, ki so prijetne za oko, kot pa aplikacije z zastarelim izgledom. Zato smo veliko pozornost posvetili tudi dizajnu. Slika 15 prikazuje osnovni pogled uporabniškega vmesnika aplikacije ROP.



Slika 15: Končni izgled uporabniškega vmesnika za primarni pogled programa ROP

4.6.1 Pregled celotnega procesa razvoja aplikacije

Celoten proces razvoja aplikacije in njenega uporabniškega vmesnika povzema diagram na sliki 16. Proces vključuje več korakov, nekateri med njimi pa se lahko tudi ponavljajo.

Slika 16: Diagram procesa razvoja uporabniškega vmesnika

Prvi in poglavitni korak pri razvoju uporabniškega vmesnika je zbiranje uporabniških zahtev. Iz teh zahtev si lahko ustvarimo okvirno sliko aplikacije od tega, katere funkcije mora podpirati, do želene oblike in upoštevanja delovnega procesa zdravstvenih delavcev. Z dobro analizo in razumevanjem uporabniških zahtev si lahko prihranimo veliko odvečnega dela kasneje. Ko zberemo uporabniške zahteve, izdelamo prvi prototip programa od osnovnih gradnikov do pogledov za posamezne podprte funkcije. V tem koraku moramo sprejeti kompromis med uporabniškimi zahtevami in oblikovanjem, saj se vseh uporabniških zahtev ne da uresničiti, ker nekatere med njimi ne ustrezajo splošni podobi in uporabnosti programa. Naslednji korak je testiranje uporabnosti izrisanega uporabniškega vmesnika za posamezni sklop akcij. Pri testiranju je pomembno, da uporabimo metodo, ki ustreza tipu aplikacije. Med testiranjem si moramo komentarje in ideje uporabnika zapisati in jih kasneje tudi upoštevati.

Po končanem testiranju napišemo poročilo in zberemo podatke ter se lotimo popravkov na testiranem uporabniškem vmesniku. Popravki so lahko vizualne ali funkcionalne narave. Slednji zahtevajo več truda, ker je po navadi potrebnih več korakov, preden pridemo do cilja funkcije. Delo na uporabniškem vmesniku, narisanem v grafičnem programu, je zelo enostavno, zato so po navadi tudi napake hitro odpravljene. Po končani implementaciji popravkov lahko ponovimo prejšnji korak in ponovno izvedemo testiranje uporabnosti. Nato ponovno upoštevamo uporabnikove predloge in jih realiziramo v programu. Ta dva koraka lahko ponavljamo, dokler niso uporabniki popolnoma zadovoljni.

Ko je specifikacija vmesnika dokončana, sledi razvoj aplikacije. Med razvojem izdelka je pomembna komunikacija med uporabniki, analitiki in programerji, saj tako pogosto odkrijemo kakšno nekonsistentnost ali težavo pri realizaciji zahtev. Ko je aplikacija dokončana, njeno beta različico ponovno testirajo uporabniki ter ponovno podajo svoje komentarje. Testiranje naj bi potekalo tako na zaslonih na dotik kot z uporabo miške in tipkovnice, da pokrijemo celotno funkcionalnost izdelka. Po končanem beta testiranju vključimo v aplikacijo dobljene komentarje in ideje ter ponovno izvedemo testiranje. Tudi ta cikel lahko ponavljamo, dokler niso z izdelkom zadovoljni tako uporabniki kot tudi razvijalci. Po končanem zadnjem ciklu je pripravljena prva različica delujočega programa z dokončno razvitim uporabniškim vmesnikom.

5 Primerjava uporabniškega vmesnika za zaslone na dotik z običajnim vmesnikom

V okviru diplomskega dela sem pripravil tudi primerjavo klasičnega vmesnika, ki uporablja tipkovnico in miško, z vmesnikom za zaslone na dotik. Testiranje obeh sem opravil s pomočjo testnih smernic iz knjige »Handbook of Usability Testing« [14]. Peto poglavje knjige navaja tipična vprašanja, na katera odgovarjamo tekom uporabe posameznega uporabniškega vmesnika oz. programa. Izpostavimo lahko nekaj najpomembnejših vprašanj:

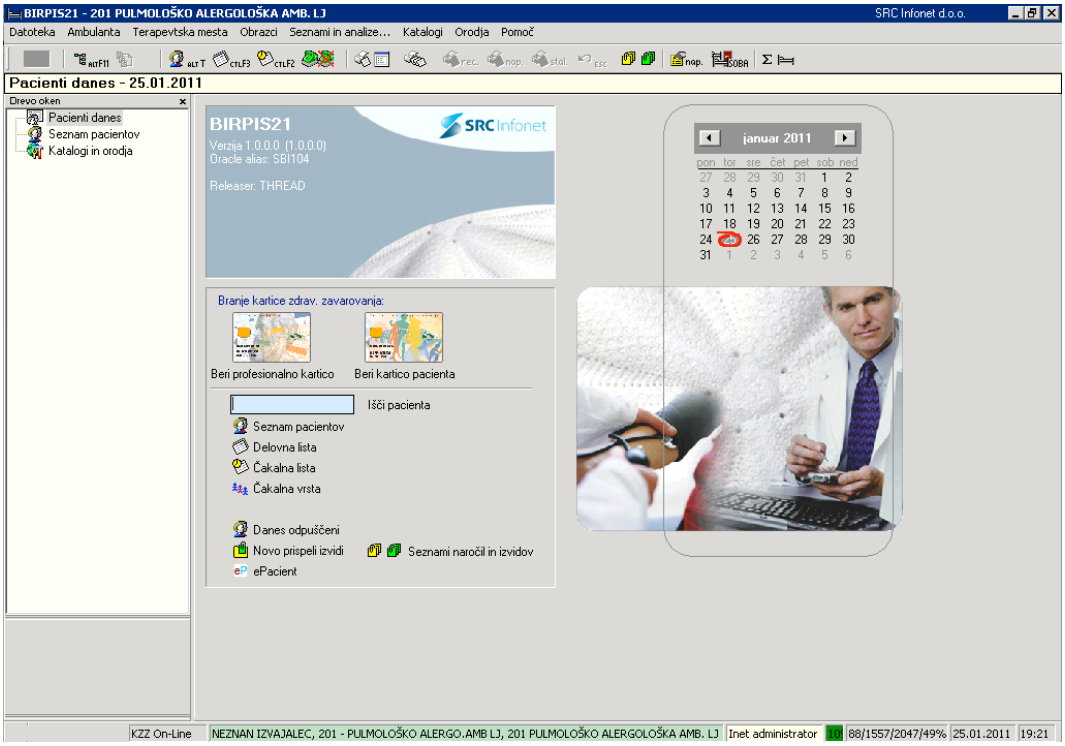
- Kako dobro uporabnik razume ikone in simbole?

- Katere ikone in simboli so problematični? Zakaj?
- Kako hitro lahko uporabnik izvede pogosto potrebne akcije?
- Kako se potek dela v aplikaciji ujema z miselnim procesom uporabnika?
- Kako hitro in uspešno uporabnik najde potrebna orodja ali nastavitve?
- Katere uporabnostne težave omejujejo izvedbo osnovnih nalog?

Na podlagi zgornjih vprašanj sem pripravil primerjavo obeh uporabniških vmesnikov [14] .

Uporabniški vmesnik za zaslone na dotik lahko primerjamo z običajnim vmesnikom, ki uporablja miško in tipkovnico (tega prikazuje slika 17), na več na načinov. Glavna področja primerjanja so:

- uporabnost aplikacije
- hitrost privajanja na aplikacijo oz. uporabniški vmesnik
- delovni tok



Slika 17: Primer uporabniškega vmesnika za uporabo z miško in tipkovnico

Primerjal sem dve aplikaciji za zdravstvo, BIRPIS21, ki je starejše izdelave in ima razvit uporabniški vmesnik za uporabo z miško in tipkovnico (slika 17), in ROP, ki ima razvit uporabniški vmesnik za zaslone, občutljive na dotik.

Testiranja so pokazala, da je delo z aplikacijami, ki uporabljajo zaslone na dotik, hitrejše od dela z aplikacijami, ki uporabljajo miško in tipkovnico. Največja prednost tipkovnice pri uporabnosti aplikacije je možnost uporabe bližnjic s kombiniranjem različnih funkcijskih tipk. Osebe, vešče uporabe bližnjic, lahko hitrost dela močno povečajo in na ta način delajo

učinkoviteje kot z zasloni na dotik. Hitrost uporabe posameznega načina je torej odvisna od izkušenj, posebej zato, ker so zasloni na dotik še dokaj nova tehnologija, ki je mnogi uporabniki še niso popolnoma navajeni uporabljati.

Hitrost privajanja na aplikacijo za zaslon na dotik je večja, ker so akcije bolj intuitivno predstavljene. Zato je uporaba programa poenostavljena in bolj intuitivna. Največji problem zaslonov na dotik je uporaba programske tipkovnice, ki je manj natančna in jo je težje uporabljati kot fizično. Po drugi strani je izvajanje posameznih akcij v aplikaciji, ki uporablja miško in tipkovnico, nekoliko bolj zapleteno, kar pa je verjetno posledica starejšega načina izdelave te aplikacije.

Elementi uporabniškega vmesnika za zaslon na dotik se morajo odzivati na večjih površinah kot pri klasičnem grafičnem uporabniškem vmesniku, saj človeški prsti niso tako natančni kot računalniška miška. Večja velikost posameznih elementov pripomore tudi k temu, da je lažje razbrati njihov pomen oziroma funkcijo, ki jo predstavljajo. Posledica večjih elementov uporabniškega vmesnika je zato večja uporabnost same aplikacije.

Uporaba zaslonov na dotik je v mnogih primerih hitrejša in učinkovitejša od uporabe miške in tipkovnice. Zaradi odsotnosti miške in tipkovnice se večina funkcij poenostavi, saj ne potrebujemo več dveh naprav za izvajanje ukazov.

Odsotnost tipkovnice in miške poveča tudi higieno in čistočo računalnika, kar je v zdravstvu še posebej pomembno. Tipkovnica namreč lahko vsebuje veliko bakterij in mikrobov, ki se skrivajo med tipkami in na njih. Tudi miška ni zelo higiensko izpopolnjena, pri računalniku z zaslonom na dotik pa se bakterije in mikrobi lahko skrivajo samo na zaslonu, ki ga je relativno enostavno temeljito očistiti.

5.1 Primerjava pristopa k razvoju obeh uporabniških vmesnikov

Razvoj uporabniškega vmesnika za zaslone na dotik se ne razlikuje veliko od razvoja običajnih uporabniških vmesnikov. Glavna posebnost razvoja prvega je skrb za primerno velikost elementov, da jih lahko aktiviramo s povprečnim prstom. Običajni uporabniški vmesnik omogoča uporabo fizične tipkovnice in s tem uporabo bližnjic na njej. Pri zaslonih na dotik dodatno težavo predstavlja dejstvo, da je težko na eni zaslonski sliki prikazati veliko količino podatkov, saj smo omejeni ne samo z majhnostjo zaslona, ampak tudi s potrebo po širših in večjih elementih, ki morajo biti primerni za izbiro s prstom. Paziti moramo, da čim bolje izkoristimo prostor. Za razvoj uporabniškega vmesnika za zaslone na dotik zaradi teh omejitev potrebujemo nekoliko več časa.

5.2 Odziv uporabnikov

Uporabniki so se na zaslone na dotik odzvali zelo pozitivno. Vsi so pripravljeni na izobraževanje in njihovo uporabo v vsakdanjem delu po bolnišnicah. Pri uporabi zaslonov na

dotik je uporabnike motila odsotnost fizične tipkovnice ter miške, ampak jim je nova tehnologija zanimiva in vabljiva za uporabo.

6 Zaključek

V diplomski nalogi sem opisal postopek razvoja uporabniškega vmesnika za zaslone na dotik. Kvaliteta uporabniškega vmesnika je nujna za dober in konkurenčen uporabniški vmesnik, kaže pa se tudi v zadovoljnih uporabnikih in hitrem delovanju. S pogostim testiranjem prototipa pri uporabnikih smo zagotovili kvaliteten in uporabniku prilagojen uporabniški vmesnik.

Postopek razvoja grafičnega uporabniškega vmesnika za zaslone, občutljive na dotik, se začne s pridobitvijo uporabniških zahtev. Brez teh podatkov razvoja ne moremo pričeti. Po analizi pridobljenih uporabniških zahtev izdelamo prvi prototip, ki ga uporabimo za izvedbo testa uporabnosti. Po dobljenih pozitivnih in negativnih rezultatih prototip popravimo in ponovno izvedemo test uporabnosti. Ta cikel ponavljamo, dokler ne ugotovimo, da je prototip zrel za naslednji korak. Prototip skupaj z dokumentacijo predamo v razvoj in tako dobimo živo različico uporabniškega vmesnika. To lahko ponovno testiramo pri uporabnikih. Po zadnjih popravkih žive različice uporabniškega vmesnika je stvar zaključena.

S pomočjo te metodologije smo razvili tudi uporabniški vmesnik za program ROP, ki ima še veliko prostora za nadaljnji razvoj in razširitev funkcionalnosti. Opazili smo, da je tak način odličen za razvoj dobrega uporabniškega vmesnika, potrebuje pa tudi veliko časa, ki ga pogosto žal nimamo na pretek. Več razpoložljivega časa tako po navadi pomeni boljši končni uporabniški vmesnik.

Uporabniški vmesnik za zaslone na dotik bi lahko dopolnili oz. izboljšali z uporabo animacij posameznih elementov, kar da uporabniku občutek večjega nadzora nad samim programom. Več izboljšav je možnih tudi pri uporabi in izgledu tabel za prikaz podatkov. Sam postopek razvoja uporabniškega vmesnika pa bi lahko izvedli v več iteracijah, kar bi pripomoglo k še boljšemu končnemu izdelku.

Kazalo slik

Viri in literatura

[1] *Capacitive sensing.* Dostopno na: http://en.wikipedia.org/wiki/Capacitive_sensing

[2] *Comparison of Samsung Galaxy Tab vs Apple iPad.* Dostopno na : http://www.sizlopedia.com/2010/09/04/comparison-of-samsung-galaxy-tab-vs-apple-ipad/

[3] *Graphical user interface.* Dostopno na: http://en.wikipedia.org/wiki/Graphical_user_interface

[4] *GUI Definition.* Dostopno na: http://www.linfo.org/gui.html

[5] *How to Implement a Software Development Process.* Dostopno na: http://www.codeproject.com/KB/testing/DevelopSoftwareStepByStep.aspx

[6] *How to Perform 100% Testing on Gui Applications.* Dostopno na: http://www.wikihow.com/Perform-100%25-Testing-on-Gui-Applications

[7] *HP EliteBook 2730p Tablet Performance.* Dostopno na: http://www.laptopmag.com/review/laptop/hp-elitebook-2730p.aspx?page=2#axzz15TII0dN0

[8] *Iterative and incremental development.* Dostopno na: http://en.wikipedia.org/wiki/Iterative_and_incremental_development

[9] Miselni vzorec zahtev in aspektov, SRC Infonet interna dokumentacija, 2010

[10] *Mockup.* Dostopno na: http://en.wikipedia.org/wiki/Mockup

[11] I. Pavlović, Metodologija testiranja uporabnosti, SRC Infonet interna dokumentacija, 2010

[12] I. Pavlović, Z. Trenz, Testiranje uporabnosti, SRC Infonet interna dokumentacija, 2010

[13] Poročilo testiranja – medicinska sestra, SRC Infonet interna dokumentacija, 2010

[14] J. Rubin, D. Chisnell, Handbook of Usability Testing: Howto Plan, Design, and Conduct Effective Tests, Indianapolis: Wiley Publishing, Inc., 2008

[15] Test uporabnosti – vsi scenariji, SRC Infonet interna dokumentacija, 2010

[16] *Touchscreen.* Dostopno na: http://en.wikipedia.org/wiki/Touchscreen

[17] *Usability testing.* Dostopno na: http://en.wikipedia.org/wiki/Usability_testing

[18] *User interface design.* Dostopno na:
http://en.wikipedia.org/wiki/User_interface_design

[19] Zapisnik testiranja SBCE, SRC Infonet interna dokumentacija, 2010

[20] *Zaslon na dotik.* Dostopno na:
http://sl.wikipedia.org/wiki/Zaslon_na_dotik