UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andraž Kopač

NAČRTOVANJE MREŽE MERILNIKOV STOPAL IN POVEZANIH SISTEMOV V MALOPRODAJI OBUTVE

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Ljubljana, 2010

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: 01585/2009
Datum: 01.09.2009

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

Kandidat: **ANDRAŽ KOPAČ**

Naslov: **NAČRTOVANJE MREŽE MERILNIKOV STOPAL IN POVEZANIH SISTEMOV V MALOPRODAJI OBUTVE**

PLANNING A NETWORK OF FOOT SCANNERS AND RELATED SYSTEMS IN FOOTWEAR RETAIL

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

V sklopu ndiplomskega dela se posvetite načrtovanju skupin programskih komponent in povezav med njimi,ki bi v kombinaciji z laserskim merilnikom stopal lahko spremenili način nakupovanja obutve v maloprodaji. Sistem naj predstavlja mrežo merilnikov stopal in obsega uporabniški in programski vmesnik merilnika, povezavo merilnikov s centralno lokacijo, administracijo merilnikov in kolekcije,
zbiranje in obdelovanje opravljenih meritev ter spletno stran s predstavitvijo kolekcije in podporo ogleda meritev in nasveta o primerni velikosti obutve.

Specificirajte zahteve posameznih komponent mreže merilnikov in podajte natančnejši opis načrtovanja tudi s pomočjo tehnik UML in s primeri iz prakse, da dobite celovito podlago za implementacijo vseh komponent mreže merilnikov.

Mentor:

prof. dr. Saša Divjak

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Franc Solina

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andraž Kopač

NAČRTOVANJE MREŽE MERILNIKOV STOPAL IN POVEZANIH SISTEMOV V MALOPRODAJI OBUTVE

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Mentor: prof. dr. Saša Divjak

Ljubljana, 2010

Zahvala

Zahvaljujem se vsem udeleženim v izdelavi projekta, dr. Matiji Jezeršku, Juretu Miklavcu, Igorju Zupanu in Boštjanu Novaku, brez katerih diploma ne bi imela praktične podlage, na kateri temelji. Za mentorstvo se zahvaljujem prof. Saši Divjaku, za potrpežljivost pa mami in nadrejenim v službi.

Kazalo

1. Povzetek

Diplomsko delo se osredotoča na načrtovanje skupine programskih komponent in povezav med njimi, ki bi v kombinaciji z laserskim merilnikom stopal lahko spremenili način nakupovanja obutve v maloprodaji. Sistem poimenujemo mreža merilnikov stopal in obsega:

- grafični uporabniški vmesnik za merilnik,
- povezavo merilnikov s centralno lokacijo,
- programe za administracijo merilnikov in kolekcije izdelkov,
- zbiranje in obdelovanje opravljenih meritev ter
- spletno stran nove kolekcije s podporo ogleda meritev in nasveta o primerni velikosti obutve.

V podjetju Alpina, d. o. o. (v nadaljevanju Alpina) smo izvedli že več poskusov razvoja takšnega sistema, zato načrtovanje temelji na praktičnih primerih in izkušnjah.

V uvodu najprej spoznamo glavne posebnosti pri nakupu obutve in si ogledamo, kako lahko merilnik spremeni način nakupa obutve. Ugotovimo, da na trgu ni primernega merilnika, kar utemeljimo z naštevanjem lastnosti, ki bi jih moral imeti. Na hitro si ogledamo razvoj primernega merilnika v podjetju Alpina, ki je osnova za mrežo merilnikov.

Sledi specifikacija zahtev posameznih komponent mreže merilnikov in nato natančnejši opis načrtovanja. Pri načrtovanju izpostavljamo probleme in opisujemo možne metode reševanja. Pomagamo si s pomočjo tehnik UML in dodajamo rešitve uporabljene v praktični implementaciji. Tako pridemo do celovitega načrta za implementacijo vseh komponent mreže merilnikov.

V zaključku dobljeni načrt podpremo z rezultati in težavami, ki jih je prinesla implementacija v praksi.

Ključne besede

- merilnik stopal,
- grafični uporabniški vmesnik,
- sistem mrežnih komponent,
- spletna aplikacija
- načrtovanje programske opreme

1. Abstract

Diploma thesis focuses on planning and developing a group of software components and their relationships which combined with laser foot scanners could change the way footwear is bought in retail stores. We call this system a network of foot scanners and it consists of:

- Graphical user interface for the scanner
- Network connection between scanners and central location
- Administration software for scanners and articles,
- Gathering and processing measurements
- Website for new line of products with features like measurement viewing and shoe size advice.

In Alpina, d. o. o. (hereafter Alpina) we already tried to develop several such systems therefore planning can be based on real life examples and experiences.

Introduction starts by explaining the main peculiarities of footwear shopping and how a foot scanner can change the process. We fail to find satisfactory scanners on the market and try to justify that with listing their missing qualities. We also take a quick look in the development of suitable scanner in Alpina, which is used as basis for our network.

Next, requirements and specifications are gathered for every component in the scanner network and then followed with more detailed planning description. During planning we expose various problems and describe possible methods for solving them. We make use of UML techniques and include solutions from practical applications. All this brings us to complete implementation plan for every component in the scanner network.

Conclusion backs up the acquired plan with results and problems of real life implementation.

Keywords

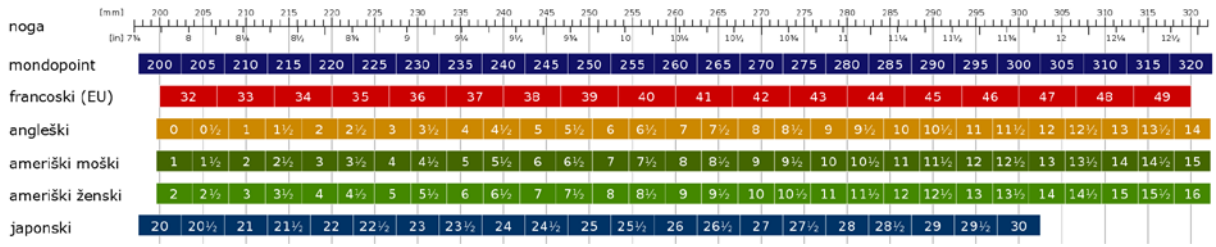
- Foot scanner
- Graphical user interface
- Components in a network system
- Web application
- Software planning

2. Uvod

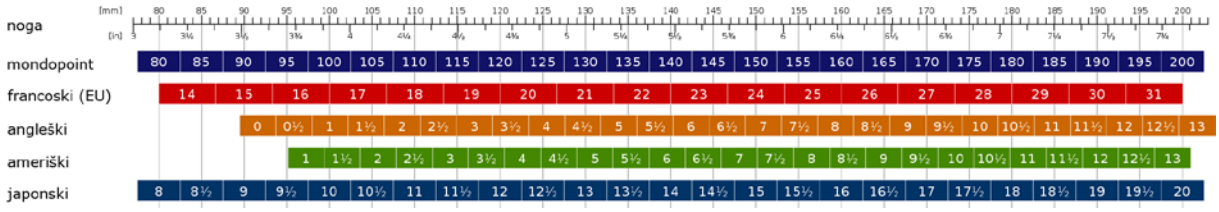
2.1. Nakupovanje obutve danes

Nakupovanje obutve ostaja enako že več desetletij. Prevladuje uveljavljena rutina obiska prodajalne, ogledovanja ponudbe, pomerjanja, izbiranja in zaključka z nakupom. Pomembnost posameznih korakov nakupa je odvisna predvsem od tipa obutve, ki jo iščemo. Pri nakupovanju modne obutve je bistveni korak vizualni pregled ponudbe, medtem ko izbira obutve za prosti čas temelji na udobnosti in prileganju, torej koraku pomerjanja. Oglaševanje je povzročilo napredek pri koraku vizualnega pregleda ponudbe. S pomočjo katalogov, oglasov in interneta lahko hitro najdemo podmnožico modelov, ki so nam vizualno všeč. Za manjši odstotek kupcev je to že dovolj za nakup preko teh kanalov, večina pa še vedno želi čevelj pred nakupom pomeriti. Nekateri ga radi tudi primejo v roke, da začutijo uporabljene materiale, trdoto ali težo. Pomerjanje pred nakupom je glavni razlog, zakaj prodaja obutve in na splošno izdelkov za tesno prileganje telesu preko interneta ne dohaja trendov internetne prodaje ostalih izdelkov, ki ne zahtevajo pomerjanja.

Nezaupanje v navedene velikostne številke proizvajalcev je upravičeno. Za začetek obstaja več sprejetih **velikostnih sistemov**, ki se prvotno delijo na celine in države, npr. francoski (evropski), angleški, ameriški, japonski, avstralski sistem. Večina ima ločene velikosti za žensko oziroma moško obutev, nekateri pa tudi za otroško [1]. Zaradi internacionalizacije se vedno več srečujemo s tujimi sistemi in približnimi pretvorbami med sistemi, ki samo pripomorejo k zmedi. Razumljivo je, da ob pomanjkanju standardov pride do odstopanj pri velikostnih številkah med proizvajalci. Poleg tega se tudi znotraj posameznega proizvajalca pojavijo razlike, predvsem zaradi različnih materialov, načinov izdelave in raznolikosti obutve. Tako sta lahko dva čevlja istega proizvajalca in iste velikostne številke različno ustrezna za določeno stopalo.



Slika 1 Primerjava velikostnih številk za odrasla stopala [1]



Slika 2 Primerjava velikostnih številk za otroška stopala [1]

Večina kupcev obutve preko interneta se trenutno zateka k velikostim, ki so navedene v milimetrih ali enakovrednemu ISO standardu **mondopoint** [2]. Za primerjavo čevljev s podobno strukturo je to dokaj dober pokazatelj. Zaupanje v internetni nakup pa dvigujejo še uveljavljene možnosti vračila nakupa, ki so pri internetni prodaji tudi zakonsko določene.

2.2. Možnosti za prihodnost

Za dvig procenta prodaje obutve preko interneta se bo moral zgoditi napredek predvsem na koraku pomerjanja, možnosti pa so odprte tudi pri vizualni predstavitvi izdelkov. Pri razvoju slednje obutvena industrija ne bo igrala glavne vloge, saj bodo nove možnosti za pristnejšo predstavitev izdelka na daljavo iskale prav vsi. Pri rešitvah za pomerjanje obutve pa je obutvena industrija prepuščena sama sebi in delno močnejšim preprodajalcem njihovih izdelkov.

Za reševanje tega problema sta trenutno na obzorju dve možnosti. Prva je globalni dogovor in standardizacija velikostnih številk. Te po možnosti ne bi obsegale samo dolžine, temveč še parametre kot so širina, višina narta in zelo pomembni obseg metatarzusa (del stopala med nartom in prsti) [3] ter še druge natančnejše kazalce. Takšen pristop lahko pričakujemo predvsem od velikih preprodajalcev obutve, ki ponujajo različne znamke, ali pa od večje skupine močnejših proizvajalcev, ki bi se povezali s tem razlogom.

Druga možnost je izdelava internih standardov ali merilne naprave, ki bi po opravljenem merjenju zagotavljala potrošniku točno prileganje, oziroma ustrezno izbiro velikostne številke. Ta pristop je pisan na kožo razvojno usmerjenim proizvajalcem obutve, ki lahko takšne aktivnosti izkoristijo tudi pri oglaševanju in večanju prepoznavnosti. To diplomsko delo se bo ukvarjalo predvsem z načrtovanjem programskih rešitev takšnega pristopa skupaj s primeri izdelave.

2.3. Merjenje stopal

Napravi, ki bi bila osnova za točno prileganje obutve kupcu, lahko rečemo merilnik stopal. Njegova naloga je dvojna. Najprej mora biti sposoben zadovoljivo natančno izmeriti velikost in obliko stopala (**merjenje, measuring**), nato pa dobljene rezultate povezati z bazo obutve in izbrati primerne izdelke (**primerjanje, matching**). Prvi del podjetju prinaša dragocene statistične podatke oblik nog, vendar sam po sebi ne bo ustvaril zanimanja za merjenje stopal v splošni javnosti. Za pravo dodano vrednost nakupa je potreben drugi del, ki pa za sabo potegne tako pozitivne kot negativne dejavnike merjenja stopal.

2.3.1. Prednosti merjenja stopal

Vzemimo za primer hipotetično podjetje, ki ima postavljeno mrežo idealnih merilnikov po svojih trgovinah, in si oglejmo situacije, ko bi merilniki skupaj z ustreznim pristopom oglaševanja lahko prinesli podjetju prednost pred konkurenco.

2.3.1.1. Gradnja podatkovne baze oblik stopal

Za razvoj obutve v podjetju velika baza izmerjenih stopal pride zelo prav. S statističnimi operacijami se lahko iz takšne baze izvleče povprečne oblike nog glede na:

- dolžino,
- starost,
- spol in
- različne regijske dejavnike.

Ugotovi se lahko, kakšna odstopanja se še splača pokrivati, da zaobjamemo optimalno število strank, hkrati pa se lahko odkrije zanimive manjše ciljne skupine potrošnikov s skupnimi značilnostmi stopal, za katere se lahko oblikuje posebne vrste obutve. Ti podatki se potem uporabijo pri izdelavi kopit, pri načrtovanju nove obutve in pri izbiri materialov.

Poleg meritev lahko taka baza obsega tudi povratno informacijo nakupa. To pomeni, da imajo meritve tudi podatek o pomerjenem, izbranem in kupljenem izdelku. Takšno bazo je težje ustvariti, obsega pa poleg geometrijskih lastnosti še človeški faktor izbire. Če je dovolj velika, se jo lahko s pomočjo strojnega učenja uporabi za natančnejšo izdelavo predloga ožjega izbora izdelkov na podlagi prejšnjih nakupov podobnih nog.

2.3.1.2. Nakup otroške obutve

Pri nakupovanju obutve za otroke lahko merilnik stopal pomeni zelo veliko dodano vrednost za stranko. Otroci ne znajo pravilno oceniti prileganja čevlja, zato z nakupom vedno rahlo ugibamo. Merilnik stopal bi lahko natančno ocenil primerno velikost posameznih otroških modelov in s tem zmanjšal tveganje nakupa, ki lahko vodi tudi do napačnega razvoja otroške noge. Hkrati bi pri otrocih to precej pogosto opravilo nakupovanja obutve postalo bolj raznoliko in zanimivo [4], povečal pa bi se tudi faktor vračanja strank.

2.3.1.3. Nakup preko interneta

Pri spletni ponudbi se nam odpre povsem nov nivo zaupanja v nakup, če nam podjetje zagotovi prileganje izbranega modela na našo prej izmerjeno nogo. Postopek nakupa bi izpustil zamudno ugotavljanje pretvorb velikostnih številk in bi temeljil samo še na pisani izbiri kolekcije in načinu plačila. Lastništvo strankine meritve pomeni tudi veliko verjetnost za prihodnje nakupe, s tem pa dobimo tudi močno CRM bazo, ki obsega zaporedja nakupov za posamezne stranke, hkrati pa nam povezuje izdelke v ponudbi k izmerjenim oblikam stopal.

2.3.1.4. Ljudje s posebnimi stopali

Ljudje imamo zelo raznoliko obliko stopal. Merilnike stopal bi se lahko izkoristilo za ugotavljanje problematičnih točk na stopalih in svetovanje takšnim nogam primerne obutve. Najpreprostejši primer nepravilnosti je razlika dolžine med levo in desno nogo. Z merilnikom podjetja Alpina lahko iz zbranih meritev različnih oseb ugotovimo, da ima le četrtina ljudi enako dolgi stopali (razlika manj kot 2mm). Kar 13% ljudi pa ima eno stopalo za več kot 6 mm daljše kot drugo. Če vemo, da je pri francoskem sortimentu razlika med celimi številkami 6,6 mm, ugotovimo, da dobra desetina ljudi potrebuje različno velikostno številko za levo in desno stopalo [5]. Ista ugotovitev velja tudi za obseg preko narta. Proizvajalec obutve ali celo večji preprodajalec bi lahko omogočil nakup para različnih velikostnih številk ali kakšen drugačen sistem prilagoditve volumna obutve ter tako pridobil stranke s posebnimi oblikami stopal.

2.3.1.5. Uporaba v povezavi z medicino

Merilnik bi proizvajalcu obutve omogočal sodelovanje z zdravstvenim sektorjem, kjer bi pacientom omogočali nakup posebne ortopedske obutve. Tako bi pridobili še en manjši, a stalen in zanesljiv trg obutve, ki omogoča tudi višjo ceno prilagojenih izdelkov.

2.3.1.6. Dviganje prepoznavnosti

Vse omenjene prednosti lahko spretna ekipa za tržno komuniciranje izkorišča ob številnih priložnostih za dviganje prepoznavnosti podjetja ali posamezne blagovne znamke. Cilja se lahko na posamezne stranke, na katere se nanaša prednost, ali pa se poudarja močne razvojne sposobnosti podjetja.

2.3.2. Slabosti merjenja stopal

V merjenju stopal bi težko našli kakšno slabost. V primeru napak in slabe izvedbe merilnika lahko pride do napačnih zaključkov, vendar tu gre za izvedbeno napako opreme, ne za problem koncepta merjenja stopal. Največja težava je, ker samo merjenje stopal za stranko

nima nobene dodane vrednosti. Da je merjenje zanimivo, mora vsebovati predlog velikosti in izdelka za nakup. Tukaj pa se stvari zapletejo zaradi natančnosti merilnika, algoritmov za izvedbo prileganja ter faktorja osebnega občutka. Kako se bo stranka odzvala na predlog merilnika, ki se ji ne bo zdel udoben?

Dokler gre le za predlog, ki ga stranka še vseeno pomeri v trgovini, je odločitev o nakupu in s tem odgovornost še vedno na strani stranke. V primeru internetne prodaje ali prodaje brez pomerjanja pa prodajalec stranki zagotavlja ustrezno velikost in s tem udobnost. Prodajalec prevzame odgovornost za odločitev o velikostni številki.

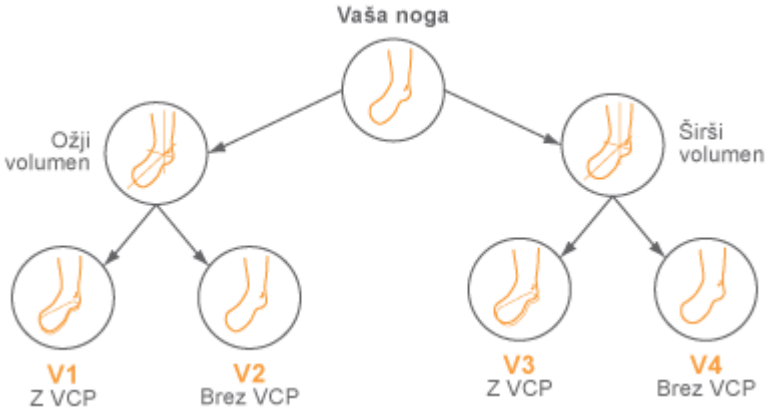
Z dobrim sistemom ugotavljanja prileganja naj bi bilo takšnih napak malo, pa vendar se mora prodajalec zavedati, da je zanje sedaj odgovoren sam. Stranki mora zato omogočati brezplačno vračanje ali menjavanje kupljene obutve, ki ne bi bila ustrezna. Tako se ne razlikuje več veliko od spletnih prodajaln brez merilnikov, ki tudi vse po vrsti zagotavljajo brezplačno vračanje. Prodajalec pa se lahko zaveda tudi tega, da bo zanesljivost računalniškega primerjanja v povprečju zagotovo večja od zanesljivosti strank s prevajalnimi tabelami velikostnih številk. Posledično lahko pričakuje manj vrnjene obutve. Z večanjem baze meritev in povezav dejanskih nakupov z njimi pa se ta zanesljivost samo še povečuje.

2.4. Primera dveh različnih pristopov izdelave prilagojene obutve

Alpina je razvila dve liniji izdelkov, ki vsaka po svoje poskušata rešiti prilagajanje obutve stranki. Razlikujeta se predvsem v stopnji prilagojenosti posamezniku.

Kolekcija ACS je skušala zelo natančno zagotoviti prileganje stopalu stranke z ločeno prodajo leve in desne številke, vsake v treh možnih širinskih različicah. Izdelke se je izdelalo po naročilu po opravljeni meritvi in niso bili na zalogi v prodajalnah.

Kolekcija Binom skuša združiti prilagodljivost s fleksibilnostjo prodaje ter zmanjšati stroške izdelave po naročilu. Pri izbiri zato uvede nekaj kompromisov. Obutev lahko kupimo v dveh širinskih različicah, ki ju lahko še dodatno prilagajamo z vstavljanjem VCP vložkov (plošče za nadzor volumna) za levo ali desno stopalo in tako dobimo 4 različne volumne (Slika 3). Ne moremo pa več kupiti različni dolžinski številki za levo in desno stopalo.



Slika 3 Širinske kombinacije kolekcije Binom [6]

3. Merilnik – osnova za mrežo merilnikov

Osnova za gradnjo mreže merilnikov stopal in povezanih sistemov je torej delujoč merilnik stopal. Več takšnih merilnikov bi postavili na izbrana prodajna mesta v maloprodajni mreži, kjer bi omogočali preprosto izvajanje meritev stopal obiskovalcev. Najprej si oglejmo, ali lahko na trgu najdemo primerni merilnik, ki bi ga lahko uporabili pri izdelavi mreže.

3.1. Kratek pregled trga obstoječih rešitev

Na trgu je na voljo precej 3D merilnikov stopal, ki pa niso namenjeni končnim kupcem obutve. Takšni merilniki so dragi, pogosto veliki, nepripravni, počasni, za uporabo pa potrebujejo najmanj kratko uvajanje, če že ne strokovnjaka. Njihova prvotna naloga je natančno merjenje za raziskave in razvoj obutve ali v medicini za izdelavo ortopedske obutve. Za te naloge so postali nepogrešljivi in upravičijo praviloma visoko ceno, saj se največkrat potrebuje le enega.

V podjetju Alpina se za natančno raziskavo stopal in kopit uporablja predvsem naslednji produkt podjetja Vorum Research Corporation:

3.1.1. YETI 3D Foot Scanner [7]

Merilnik YETI (Slika 4) lahko meri tako stopala kot kopita in meri oblike za uporabo pri načrtovanju in izdelavi ortopedske in posebne obutve. Sistem obsega ojačeno ohišje s kamerami in programsko opremo za računalnik.

Meritev poteka vzdolž stopala z merjenjem prečnih presekov. Štirje laserji osvetlijo črto trenutno merjenega preseka, nato pa osem kamer v ohišju izračuna njegovo lokacijo v prostoru. Laserji in kamere se mehansko pomikajo vzdolž stopala in merijo presek za presekom. Postopek traja nekaj sekund, odvisno od nastavitve natančnosti. Največja dosežena natančnost je +-0.5 mm, zanjo pa moramo obuti bele nogavice [8]. Če želimo, da programska oprema izračuna standardne dolžine, širine in obsege stopala, moramo na bele nogavice nalepiti štiri črne markerje, ki označujejo najbolj izpostavljene točke stopala (1. in 5. zunanji metatarzalni sklep ter notranji in zunanji malleolus) [9]

Slika 4 Vorum YETI™ Foot Scanner [7]

Slika 5 INFOOT scanner [10]

3.1.2. INFOOT scanner [10]

Merilnik japonskega proizvajalca i-Ware laboratory (Slika 5) je zasnovan zelo podobno kot YETI. Meritev izvaja vzdolž stopala z laserjem in sistemom kamer. Razvoj so nadaljevali skupaj z angleškim podjetjem CSM 3D, zdaj del skupine Torielli, ki za obutveno industrijo razvijajo tudi laserske in druge rezalnike.

3.2. Lastnosti merilnika za maloprodajno mrežo

Omenjeni produkti niso primerni za gradnjo mreže merilnikov za končne kupce zaradi prej omenjenih slabosti. Manjkajo jim lastnosti, ki bi jih takšen merilnik moral imeti:

- **Robustnost** – merilnik je namenjen vsem obiskovalcem trgovine, ki niso pod stalnim nadzorom. Preživeti mora srečanja z radovednimi otroki ter nerodnimi, nepremišljenimi in kdaj tudi zlobnimi strankami. Občasni tresljaji ne smejo pokvariti kalibracije, prav tako ne različni pogoji okolja, na primer svetlobni.
- **Hitrost in preprostost** – meritev mora biti opravljena hitro in preprosto, saj se strankine radovednosti, da poskusi, ne sme kaznovati z zapletenim in dolgotrajnim postopkom.
- **Rezultat** – stranka mora od meritve nekaj pridobiti. Za začetek bi to lahko bili podatki o njenih nogah, kot so dolžina, širina ali slika. Nadaljevanje je lahko prikaz kolekcije z rezultatom prileganja strankini nogi. Če je stranki kakšen čevelj všeč, bi ga lahko pomerila takoj v najustreznejši številki.
- **Cena** – pri izdelavi mreže merilnikov govorimo o večjem številu naprav. Cena posameznega merilnika in njegovega vzdrževanja mora biti čim nižja, saj želimo pokriti čim večje število prodajnih mest. Poleg tega bodo merilniki večino časa prepuščeni obiskovalcem, zato so poškodbe neizogibne, njihova odprava pa ne sme biti predraga.
- **Centraliziranost in celovitost** – ko govorimo o mreži merilnikov, morajo ti biti povezani na neko centralno točko za upravljanje. Tam bi se zbirale vse meritve in rezultati, hkrati pa bi točka služila za nadzor vseh merilnikov, njihovega stanja in delovanja, urejanje prikaznih podatkov, kalibracije in ostale administracijske posege.

Po naštetih lastnostih lahko vidimo, da je prepad med dobrim merilnikom in njegovo uporabo v maloprodaji še precej velik. Na trgu sicer najdemo tudi manjše merilnike [11,12,13], tudi takšne s prilagojenim vmesnikom za merjenje in nekaterimi naštetimi lastnostmi. Vseeno pa celovite rešitve mreže merilnikov za enkrat še ne bomo našli, predvsem zaradi specifičnosti posameznih prodajalcev, ki postane ovira pri povezovanju golega merilnika v mrežo in poslovni model prodajalca.

Obstajajo podjetja, ki so poskušala strankam zagotoviti prijazno merjenje stopal, vendar se je vse končalo v obliki enega (testnega) merilnika v neki trgovini, ki je znal izmeriti nogo. Tukaj je še veliko prostora za izboljšave.

3.3. Poskusi podjetja Alpina v preteklosti

Prvi poskus Alpine na področju merilnikov izvira iz leta 2000, ko so s pomočjo Fakultete za strojništvo razvili precej velik merilnik za obutev po meri, na katerem je bilo mogoče kupiti posebno **ACS** obutev v treh različnih širinah in različne dolžine za levo in desno nogo. Sledila je izboljšana in manjša različica merilnika, ki pa je pripeljala tudi nestrinjanje Alpine s finančnimi pogoji proizvajalca merilnika (UCS), zato se je sodelovanje prekinilo. Alpina je

želela preprostejšo in cenejšo rešitev merjenja stopal v maloprodajni mreži, zato so se vrnili k izdelavi lastne rešitve v hiši.

Poizkusili so na dva različna načina, ampak se je pri obeh ustavilo že zaradi nenatančne meritve, še preden pridemo do problemov mreže merilnikov.

Oba merilnika sta temeljila na pridobivanju dvodimenzionalne slike stopala od spodaj in iz strani. Pri prvem smo to skušali doseči z vzporednim snopom svetlobe, sistemom zrcal in digitalnim fotoaparatom. Izkazalo se je, da je vzporedni snop svetlobe zelo težko doseči, prav tako je sistem zrcal v končni konstrukciji bil prevelik. Problematični so bili tudi svetlobni pogoji v okolici merilnika. Edini del projekta, ki ni predstavljal težave, je bila zelo dobra podpora programskih orodij (SDK) za vodenje digitalnih fotoaparatov [14].

V drugem poskusu smo skušali sistem zrcal nadomestiti z navadnimi namiznimi optičnimi čitalci, ohranili pa smo sistem vzporednega snopa svetlobe. Rezultat je bil nekaj manjši merilnik, ki pa je bil tudi počasnejši. Tudi ta prototip ni dosegal želene natančnosti in fleksibilnosti.

3.4. Primer trenutnega merilnika podjetja Alpina

Po spodletelih poskusih izdelave merilnika se je Alpina odločila za novo sodelovanje z Ljubljansko fakulteto za strojništvo. Dr. Matija Jezeršek je izdelal 3D laserski merilnik in konstrukcijo zanj, ki obeta dobro osnovo za mrežo merilnikov. Sledi kratek opis tega merilnika, ki je osnova za mrežo merilnikov in s tem za moje diplomsko delo.

3.4.1. Zgradba merilnika

Merilnik je sestavljen iz treh osnovnih gradnikov:

- okrogla plošča, na kateri stranka stoji,
- premična glava z dvema kamerama in laserjem,
- pokončno ogrodje z vgrajenim na dotik občutljivim zaslonom in računalnikom v notranjosti.

Celotni merilnik je precej manjši od prvotnih poskusov podjetja Alpina in je po velikosti precej optimalen glede na stabilnost, prenosljivost, porabo prostora in tudi ceno izdelave konstrukcije.

3.4.2. Dva tipa merilnika

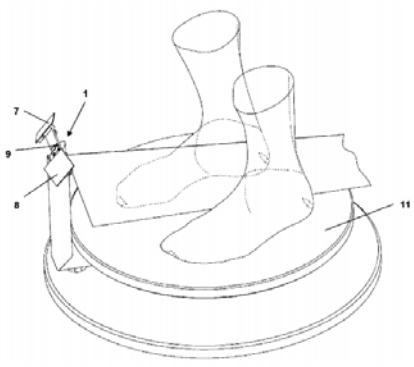
Poleg že opisane zgradbe merilnika za v trgovine se je pojavila še potreba po manjšem in bolj prenosnem merilniku, ki bi se uporabljal ob različnih akcijah merjenja stopal, npr. v vojski, zdravstvenih domovih, na letališčih, prireditvah, itn. Pri takšnih akcijah merilnik ne bi bil brez nadzora na voljo obiskovalcem, zato je lahko manj robusten in tudi manj samoumeven.

Od osnovne zgradbe se je ohranilo okroglo ploščo in premično glavo, medtem ko je pokončno ogrodje zamenjal prenosni računalnik. Takšen merilnik se lahko premika v večjem potovalnem kovčku.

3.4.3. Način merjenja

Merilni modul je pritrjen na premično glavo in vsebuje dve kameri in linijski laserski projektor, ki so med sabo zamaknjeni. Premična glava se ob meritvi zapelje okoli stopal in obe kameri zabeležita položaj projecirane laserske črte (Slika 6). Ob kalibriranem sistemu se s pomočjo triangulacije iz zajetih odbojev svetlobe na stopalih lahko izračuna 3D obliko.

Podrobnosti o uporabljenih metodah dr. Matije Jezerška niso del te diplomske naloge, so pa na voljo v njegovi disertaciji [15]. Dejanski sistem uporabljen v merilniku za Alpino pa je patentiran v patentu [16] iz dne 15.5.2008.



Slika 6 3D merjenje stopal [16]

3.4.4. Programski vmesnik merilnika podjetja Alpina

Podatki od zajema do prikaza na zaslonu potujejo preko ločenih komponent, od katerih vsaka skrbi za del obdelave. Prva komponenta skrbi za zajem 3D oblike v surovi obliki in za osnovno obdelavo, ki vrne podatke o dolžinah, širinah in obsegih stopal. Druga komponenta prevzame te podatke in jih primerja z vnaprej pripravljeno bazo modelov obutve. Za vsak model vrne optimalno velikostno številko ter različne parametre ujemanja. Obe komponenti sta del zgradbe merilnika in sta delo Dr. Matija Jezerška. Napisani sta v jeziku C++ za in prevedeni v obliko Windows izvršilne datoteke (.exe).

4. Zahteve mreže merilnikov

4.1. Zahteve programske opreme merilnika

4.1.1. Grafični vmesnik

Glavna zahteva pri grafičnem vmesniku je preprostost. Celotna naprava mora delovati tako samoumevno, da lahko naključni obiskovalec brez pomoči v prvem poskusu izmeri nogo in pridobi podatke o meritvi. Vmesnik mora biti lep na pogled in mora delovati hitro in tekoče.

Merilnik za pravilno merjenje od uporabnika zahteva pripravo v obliki vihanja hlačnice, obuvanja priloženih belih nogavic ter postavljanja na pravo mesto. Naloga vmesnika je, da uporabniku brez pomoči prodajalca dopove, kaj mora storiti.

Po opravljenem merjenju naj se zaslon izpišejo naslednji podatki o meritvi:

- sploščen 2D prikaz stopal s pogledom z vrha in od strani,
- izmerjeno dolžino, širino in obseg ter
- širina noge glede na povprečje.

V primeru ponesrečene meritve naj merilnik omogoča ponovno izvajanje koraka meritve. Če se iz podatkov lahko ugotovi, da je meritev ponesrečena, pa naj se ponovi avtomatično.

Poleg podatkov o meritvi želimo obiskovalcu omogočiti še hiter in preprost pregled kolekcije čevljev primernih za njegovo nogo, izmed katerih bi si jih lahko nekaj izbral v ožji izbor. Za lažjo navigacijo morajo čevlji biti združeni v skupine.

Obiskovalec mora po končanem pregledu kolekcije imeti možnost natisniti rezultat meritve, na katerem so podatki o stopalih in seznam čevljev iz ožjega izbora. Izpis vsebuje tudi kodo meritve, katero lahko uporabi na spletni strani za ponovni ogled opravljene meritve.

Na koncu se obiskovalcu ponudi še možnost reševanja kratke ankete na temo merjenja stopal.

Če obiskovalec sredi poteka odide, se mora merilnik čez nekaj časa avtomatično postaviti v začetno stanje in čakati naslednjega obiskovalca. Če je bila meritev pred tem že uspešno opravljena, se mora zaključiti in shraniti.

Celotni uporabniški vmesnik mora biti pripravljen na večjezično uporabo, saj je prvi pogoj za preprostost uporabniški vmesnik v domačem jeziku obiskovalca. Izbira jezika naj bo del zagonske nastavitve, lahko pa se uporabniku omogoči intuitivno menjavo jezika tudi med delovanjem, na primer na začetnem zaslonu.

Med postopkom merjenja želimo od obiskovalca izvedeti starost in spol. Starost bi se uporabljala predvsem v statistične namene pri analizi obiskovalcev, spol pa je pomemben pri izbiri izdelkov, ki jih ponudimo obiskovalcu.

4.1.2. Programski vmesnik

V ozadju grafičnega vmesnika mora aplikacija opravljati še nekaj pomembnih operacij, ki pripomorejo k bolj avtonomnem delovanju merilnikov ter olajšajo upravljanje v prihodnosti. Pod takšne operacije spadajo:

- nastavitvena datoteka merilnika, ki jo lahko administrator preprosto popravlja tako na daljavo kot na lokaciji,
- pisanje dnevnika dogodkov, ki olajša razhroščevanje v primeru napak,
- sprotno pošiljanje zajetih meritev na centralno lokacijo za prikaz na internetu in za kasnejše obdelave,
- periodično posodabljanje merilnika iz centralne lokacije, ko je na voljo mrežna povezava. Posodabljati je potrebno:
    - kolekcijo izdelkov,
    - slike izdelkov,
    - datoteke za izvajanje primerjanja (matching).

4.2. Zahteve programske opreme za administracijo merilnikov

4.2.1. Administracija merilnikov

Za nadzor nad delovanjem merilnikov v mreži se potrebuje posebno aplikacijo, s katero je mogoče iz centralne lokacije preveriti delovanje merilnikov. Funkcionalnosti, ki jih mora omogočati so pregled dnevnika dogodkov, pregled trenutnega stanja, nadgradnja merilnika, spreminjanje nastavitev merilnika in zahteva za ponovni zagon. Z njo se bo nadzorovalo večje število merilnikov, ki morajo biti hitro dostopni iz shranjenega seznama.

4.2.2. Administracija kolekcije

Za izdelavo kolekcije izdelkov, ki bo na voljo na posameznem merilniku, se prav tako potrebuje aplikacijo, ki bo olajšala to delo. Aplikacija mora znati prevesti podatke iz prodajalčevega sistema izdelkov v podatke potrebne za merilnik. Omogočati mora tudi njihovo dodajanje ali spreminjanje v primeru, da podatki iz prodajalčevega sistema o izdelkih ne bi zadoščali za prikaz na merilniku. Urejanje celotne kolekcije želimo imeti na enem mestu. Program naj omogoča ustvarjanje skupin merilnikov, ki imajo enako podmnožico izdelkov v kolekciji. Potrebujemo torej vmesnik za ustvarjanja povezav med izdelki in skupinami.

4.3. Zahteve programske opreme za obdelavo zbranih meritev

Med obratovanjem mreže merilnikov se nam bo na centralni lokaciji nabralo večje število meritev. Za njihovo obdelavo potrebujemo fleksibilne aplikacije, saj je nemogoče vnaprej predvideti vse načine obdelave. Za surovo matematično procesiranje se potrebuje programske vmesnike, s pomočjo katerih se lahko hitro napiše aplikacije za specifične obdelave. Bistvo takšnih aplikacij je samo koda za izračune in obdelavo, za čim več ostalih pomožnih operacij pa naj poskrbijo že pripravljeni razredi.

Poleg matematične obdelave se potrebuje tudi splošna poročila o meritvah, s pomočjo katerih bi s čim manj truda izdelovali mesečne povzetke. Takšna poročila naj bodo primerna tudi za tehnično nepodkovane bralce in naj bodo v standardnih formatih, npr. PDF.

4.4. Zahteve spletne strani za prikaz meritev

Najbolj osnovna izvedba takšne spletne aplikacije bi obsegala samo obrazec za vnos kode meritve in bi nato prikazala rezultate meritve. Obiskovalec si s takšno stranjo ne more kaj prida pomagati. Največ, kar lahko stori je, da pokaže svoja stopala prijateljem.

Lahko pa to spletno stran razvijemo v poljubno kompleksno spletno trgovino. Z združevanjem podatkov o meritvah stopal obiskovalcev in ponudbo spletne prodaje obutve lahko izdelamo močno in ta trenutek unikatno spletno prodajalno. V podjetju Alpina smo to možnost prestavili za naslednjo stopnjo projekta merilnikov. Za začetek smo izbrali nekakšno vmesno pot v obliki spletne strani za predstavitev nove linije izdelkov, ki vsebuje tudi vse dodatne možnosti, ki jih prinese merilnik stopal.

Spletna stran mora omogočati vnos kode meritve in s tem ogled stopal in izmerjenih podatkov. Ogled meritve naj vsebuje iste podatke, kot smo jih videli na merilniku. Dodatno se prikaže še datum in čas izvedbe meritve ter vneseno starost in spol. Doseči želimo tudi, da obiskovalec lahko pošlje povezavo do svoje meritve prijateljem (permalink).

Obiskovalca ne bi obremenjevali z registracijo, saj stran ni na nivoju spletne trgovine, kjer bi bil uporabniški račun z zgodovino nujen. Radi pa bi vseeno zbrali nekaj statistike in elektronskih naslovov. Potrebujemo torej preprost sistem uporabnikov, ki ne zahteva registracije.

Pomembno je tudi, da preko spletne strani ne bi bilo mogoče preprosto priti do vseh opravljenih meritev. Takšna baza namreč podjetju pomeni veliko in nočemo, da bi konkurenčno podjetje preprosto prišlo do vseh zajetih meritev.

Predstavitev nove linije izdelkov naj bo običajen prikaz izdelkov in njihovih lastnosti. Posebnosti se pojavijo, če ima obiskovalec vneseno meritev. V tem primeru naj pri vsakem izdelku sistem izpiše tudi predlagano velikostno številko.

Stran lahko opremimo še s seznamom prodajnih mest in zalogo posameznih modelov po teh prodajnih mestih. Ob vsakem izdelku bo torej možen ogled vseh prodajnih mest, ki imajo izmerjeno velikost na zalogi. Za izvedbo te funkcionalnosti bo potrebno prenašati aktualne podatke o zalogah iz centralnega sistema proizvajalca.

5. Načrtovanje mreže merilnikov

5.1. Načrtovanje strojne in programske opreme merilnika

5.1.1. Strojna oprema

Izbiro strojne in programske opreme določajo končne aplikacije. V njih je potrebno poiskati zahtevne odseke in določiti želeno odzivnost sistema ob predvidenih neugodnih situacijah.

Če bi imeli popolnoma neodvisen merilnik, ki je ločen od programskega vmesnika, bi breme zajema 3D podatkov prepustili strojni opremi merilnika. Koncept merilnika iz Alpine pa predvideva skupno strojno opremo za vse potrebne operacije. Kritični del je zajem velikega števila slik iz kamer med merjenjem v realnem času ob premikanju merilnega modula, ki zahteva moč povprečnega domačega računalnika oziroma delovne postaje. To prepreči uporabo vgradnega sistema ampak vseeno predstavlja cenovno ugodno strojno opremo. Izhodiščni merilnik Alpine je dovolj velik, da vanj brez težav vgradimo PC računalnik standardne velikosti, oziroma uporabimo malce zmanjšan tip ležečega ohišja za primernejšo razporeditev vhodno izhodnih priključkov.

Izbira najbolj razširjene arhitekture ima tudi druge prednost. Na njej teče največja izbira operacijskih sistemov in aplikacij ter hkrati ponuja veliko razširitvenih naprav. PC arhitektura sicer ni najbolj optimalna rešitev, je pa najmanj omejujoča. Format oblike (form factor) in poraba postajata vedno manjša, zmogljivost pa se stalno povečuje.

Takšna strojna oprema bi morala zadoščati tudi za vse zgoraj naštete zahteve programskega vmesnika, od katerih je najzahtevnejši grafični vmesnik. Animiran grafični vmesnik s prijetno uporabniško izkušnjo je lahko velik porabnik resursov, predvsem če izbiramo grafiki neprijazne programske jezike oziroma se nismo pripravljeni posvetiti optimizaciji grafičnega vmesnika.

Pri izbiri strojne opreme moramo biti pozorni tudi na možnost priklopa vseh želenih vhodno izhodnih naprav. Za merilnik je zelo verjetna uporaba tiskalnika, potrebuje se mrežno kartico, možna je uporaba zvočne kartice, pomisliti moramo na način rokovanja z merilnikom, itn. V primeru Alpinskega merilnika se uporablja na dotik občutljiv zaslon, za krmiljenje merilnih komponent pa se potrebuje serijski in firewire vmesnik. Za odpravo zahteve po serijskem vmesniku, ki zna na novejših matičnih ploščah majhnega formata biti redkost, smo uporabili pretvornik med RS232 in USB. Pri slednjih moramo paziti, saj ne delujejo vse naprave z vsakim pretvornikom. Ko najdemo delujoč izdelek, se ga je pametno držati.

5.1.2. Programska oprema

5.1.2.1. Operacijski sistem

Osnova programske opreme je operacijski sistem. Koncept merilnika sam po sebi nikjer ne omejuje njegove izbire. Težave lahko nastanejo ob uporabi specifične strojne opreme, kjer nas lahko gonilniki ali razvojna okolja za uporabo naprave (SDK) omejijo. Drugi večji dejavnik je razpoložljivo znanje razvojnih strokovnjakov, saj se vedno izplača izbrati sisteme, ki jih razvojna ekipa pozna in ima z njimi že izkušnje. Glede na razširjenost sta glavna kandidata za osnovo merilnika Windows in Linux.

V primeru merilnika podjetja Alpina je dr. Jezeršek je za zajem iz firewire kamer uporabil jezik C++ in Windows knjižnice, kar je narekovalo tudi izbiro operacijskega sistema Windows, in sicer stare in preizkušene različice XP. Tudi administratorji, ki bodo upravljali osnovne operacije na merilnikih so veliko bolj vajeni Windows okolja. Seveda pa je cena posameznega merilnika zato dražja za Windows licenco.

5.1.2.2. Spremljevalna programska oprema

Poleg operacijskega sistema potrebujemo še nekaj programov, ki dopolnijo osnovno inštalacijo operacijskega sistema. Iskali smo predvsem programe, ki so brezplačni in tako ne večajo stroška merilnika.

Prva naloga je njegovo zakrivanje, da uporabnik dobi občutek, da je sistem popolnoma namenski merilnik z eno samo nalogo. Za dosego tega cilja mora osnovna aplikacija teči čez celotni zaslon in se pognati skupaj z zagonom sistema. Onemogočiti moramo procese v ozadju, ki bi lahko povzročili prikaz neželenih pogovornih oken. Primeri takšnih procesov so na primer obvestila o novejših različicah programov. Kljub temu se lahko hitro zgodi, da se vseeno kdaj prikrade kakšno obvestilo med delovanjem aplikacije. Za bolj zanesljivo zakrivanje moramo aplikacijo narediti tako, da je vedno na vrhu vseh ostalih oken (always on top). V operacijskem sistemu Windows se skupaj s pogovornim oknom pogosto v ospredje postavi tudi opravilna vrstica, ki pa ne upošteva pravila vedno na vrhu. Zato je priporočljivo, da opravilno vrstico v celoti skrijemo. Za to opravilo smo v Alpini uporabili brezplačni program **ObjectDock** podjetja Stardock [17], ki zanesljivo in popolnoma skrije opravilno vrstico.

Za občasne posege na merilniku potrebujemo možnost normalnega dela preko zaslona na dotik brez tipkovnice. USB vtičnice na ohišje merilnika ne smemo postaviti, ker bi omogočala preveč preprosto interakcijo s sistemom tudi nepooblaščenim osebam. Lahko bi jo skrili pod ključ, ampak za kratke posege obstaja še ena možnost. Windows ima že vgrajeno tipkovnico na zaslonu, ampak je njena uporaba zelo omejena. Za boljšo izbiro se je izkazal program **Touch-It Virtual Keyboard** podjetja Chessware SA [18]. Zaradi zaslona na dotik je priporočljivo tudi izklopiti miškin kazalec, saj je ta v takšnem uporabniškem vmesniku odveč. To operacijo naj bi omogočal gonilnik zaslona na dotik.

Za oddaljene posege na merilniku potrebujemo zanesljivo rešitev oddaljenega dostopa. Predvsem je pomembno, da je rešitev sposobna obiti požarne zidove in NAT usmerjevalnike, saj nikoli ne vemo, kje se bo nahajal prenosni merilnik. Ustrezne so torej samo rešitve, ki delujejo preko posredniških strežnikov, najbolje HTTP(S) protokola. Primer takšnega produkta je **LogMeIn** [19], ki z brezplačno različico pokrije osnovne zahteve oddaljenega dostopa za Windows. Za Linux je takšnih rešitev manj, še posebej brezplačnih, vendar če imamo zagotovljen VPN ali kontrolo nad požarnimi zidovi na poti, lahko uporabimo tudi običajne programe za oddaljen dostop, kot so različne verzije VNC. Pri slednjih lahko uporabimo možnost vzpostavitve povezave s strani merilnika, vendar mora za to posredovati prodajalec na lokaciji merilnika.

Za pravilno delovanje posameznega merilnika v celotni mreži merilnikov je pomembno tudi, da ima točno uro. Ura v današnjih računalnikih ne teče pretirano točno, poleg tega pa se dogaja, da se zaradi izpraznjene baterije na matični plošči ponastavi BIOS. Za zagotavljanje točne ure smo v osnovno inštalacijo merilnika vključili program, ki deluje s pomočjo protokola **SNTP** (Simlpe Network Time Protocol) [20]. Osnovne možnosti podpira že Windows, ampak je zanesljivost in nastavljivost preveč omejena. Primer programa, ki ga lahko uporabimo, je **Automachron** [21], katerega razvoj se je sicer ustavil, vendar zelo dobro služi namenu, dobimo pa ga v zastonjskih zbirkah programov na internetu. V operacijskem

sistemu Linux lahko nastavimo servis **ntpd**, ki ga imajo vse distribucije že vgrajenega, ali pa ga ponujajo kot dodatni paket.

Čas neaktivnosti merilnika lahko izkoristimo za predvajanje predstavitvenega videa merilnika ali podjetja. Za to lahko uporabimo kar ohranjevalnik zaslona operacijskega sistema. Pri Windows je za to potrebno v naprej iz videa ustvariti .scr datoteko ohranjevalnika zaslona. Lahko pa uporabimo aplikacijo **AVISS** [22], ki ustvari ohranjevalnik zaslona z možnostjo predvajanja video datotek. Tudi ta program nima več izhodiščne spletne strani, še vedno pa ga dobimo v zbirkah zastonjskih programov na internetu.

5.1.3. Izbira programskega jezika

Za programski in grafični vmesnik merilnika potrebujemo programski jezik z dobro podporo za delo z grafičnimi elementi in z omrežnimi povezavami ter podporo večnitnosti. Ker je pri industrijskih projektih vedno pomemben tudi čas razvoja, moramo izbrati jezik, ki ga čim bolje poznamo in omogoča hitro izdelavo aplikacije. Pri izbiri igra vlogo tudi moč razvojnih okolij, ki so na voljo, ter cena njihovih licenc.

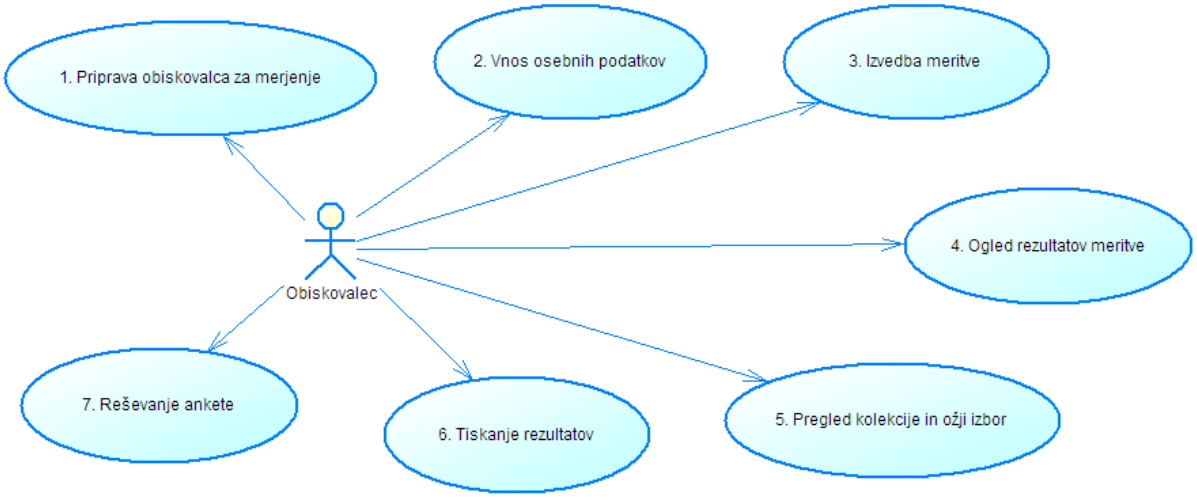
Pri izbiri si lahko pomagamo na primer z metodo SAESA (Selecting Appropriate Software Engineering Assets) [23], po kateri najprej ocenimo velikost in zapletenost projekta. Celotni sistem lahko glede na metodo ocenimo kot srednje velik in zapleten (med 10.000 in 50.000 vrstic kode), medtem ko bodo posamezni deli najverjetneje vsi majhni in nezapleteni (manj kot 10.000 vrstic kode). V uvodu smo že ugotovili, da želene rešitve na trgu še ni, pomagamo pa si lahko z manjšimi sestavnimi deli sistema. Ti deli so premajhni, da bi bistveno vplivali na izbiro, zato sistem obravnavamo brez njih. Metoda predlaga prepuščanje izdelave majhnih komponent posameznikom in uporaba jezika in metod izdelave iz virov, ki so nam trenutno na voljo. Če bi predvidevali večjo rast projekta, bi ga morali načrtovati kot velik in zapleten, za kar bi uporabili nadaljnje tabele SAESA metode.

Primer ustreznega jezika in tudi izbira za Alpinin merilnik je Java, ki je kos vsem zahtevam vmesnika. Gre sicer za enega bolj požrešnih programskih jezikov, vendar že za merilnik potrebujemo dovolj močno strojno opremo, ki bi morala brez težav gladko poganjati slikovit uporabniški vmesnik. Z izbiro Jave smo odprli vrata morebitnemu prehodu na cenejšo Linux platformo v prihodnosti. Prevajalniki so brezplačni, prav tako pa je brezplačno tudi močno razvojno okolje Eclipse [24]. Najpomembnejši razlog pri izbiri programskega jezika pa so izkušnje razvojne ekipe. Če hočemo v industriji z interno skupino ljudi, torej brez namenskega zaposlovanja ali zunanje izdelave (outsourcing), čim hitreje izdelati aplikacijo, moramo uporabiti orodja, ki jih ta skupina že pozna.

5.2. Načrtovanje grafičnega vmesnika

Sestavljanje načrta uporabniškega vmesnika je dober začetek takšnega projekta, znan tudi kot pristop »Najprej vmesnik« po knjigi Getting Real [25]. Najti je potrebno način, kako na najbolj preprost možen način uporabnika popeljati skozi postopek meritve stopal. Začnemo pri uporabniškem vidiku aplikacije in skušamo najti idealni način njegovega dela.

Z upoštevanjem zahtev grafičnega vmesnika dobimo skupino nalog, ki jih mora vmesnik pokriti (Slika 7). Na splošno velja, da če hočemo uporabnika brez predznanja popeljati skozi nek postopek, je najprimernejša uporaba čarovnikov. V primeru merjenja stopal na uporabo čarovnika namiguje že zaporednost postopka z več koraki. Posamezne korake moramo dovolj razdrobiti, da je besedila z navodili na posameznih zaslonih minimalno. Tudi navigacija po vmesniku v obliki čarovnika bo za uporabnika najbolj samoumevna in preprosta.



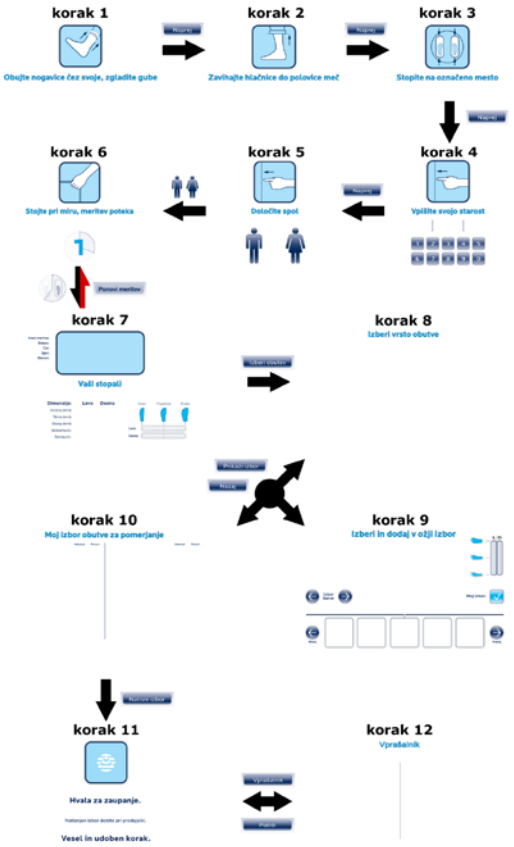
Slika 7 Primeri uporabe grafičnega vmesnika

V primeru Alpinskega merilnika se je določanje korakov in oblikovanje uporabniškega vmesnika izvajalo hkrati in izmenično, da smo si lahko bolje predstavljali končno podobo postopka. Grafično podobo vmesnika je oblikoval Jure Miklavc iz podjetja Jure Miklavc Studio. Rezultat si bomo kot primer podrobno ogledali po korakih.

5.2.1. Koraki grafičnega vmesnika

Postopek smo razstavili na 12 korakov (Slika 8):

- *koraki 1, 2 in 3 pripravijo obiskovalca v pravilni položaj za izvedbo meritve,*
- *koraka 4 in 5 od njega pridobita podatka o starosti in spolu,*
- *v koraku 6 se izvaja meritev, na zaslonu pa je aktivna animacija napredka,*
- *korak 7 vsebuje vse izmerjene podatke,*
- *v korakih 8, 9 in 10 obiskovalec pregleduje kolekcijo obutve in si sestavlja ožji izbor všečnih modelov,*
- *s prehodom v korak 11 se rezultati in ožji izbor natisnejo,*
- *če želi, lahko reši še anketo v koraku 12.*



Slika 8 Celotni načrt korakov izvedbe meritve [26]

5.2.1.1. Priprava obiskovalca za merjenje (Slika 9)

Za opravljanje meritve se mora obiskovalec sezuti, si nadeti priložene bele nogavice in zavihati hlačnice. Prvi trije koraki vmesnika so namenjeni preprostim navodilom in ikonam, ki pripravijo obiskovalca, da stori zahtevano pripravo pred merjenjem.

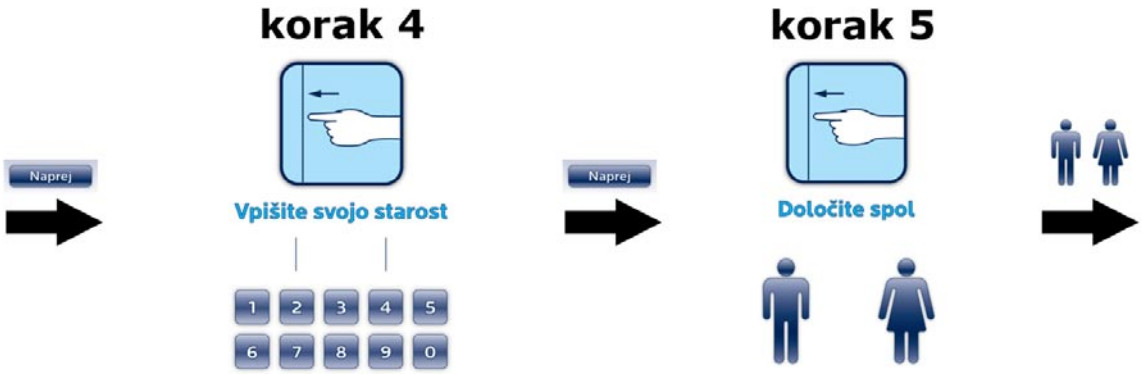
Slika 9 Priprava obiskovalca za merjenje [26]

5.2.1.2. Vnos osebnih podatkov (Slika 10)

Sledi vnos zahtevanih osebnih podatkov. Najprej od obiskovalca zahtevamo vnos starosti s preprosto številčnico. Najvišji možni vnos starosti omejimo, čeprav to ne pomaga veliko pri preprečevanju namernih napačnih vnosov.

Ta korak se zdi zelo primerno mesto za vnos komandnih kod v merilnik. Z vpisom dogovorjene »starosti« lahko izklopimo vmesnik, če hočemo servisirati merilnik preko namizja operacijskega sistema. Pametno je tudi dodati kodo za nadaljevanje meritve s testnim uporabnikom, katerega rezultat se ne bo shranil po zaključku meritve. To kodo bi uporabljali za testiranje pravilnosti delovanja merilnega procesa, brez da bi smetili po bazi pravih meritev.

V naslednjem koraku sledi izbor spola preko klika na ustrezni simbol.



Slika 10 Vnos osebnih podatkov [26]

5.2.1.3. Izvedba meritve (Slika 11)

Vmesnik požene zagonsko datoteko za izvedbo merjenja. V obliki parametra ji poda v prejšnjih korakih pridobljena podatka o starosti in spolu. Med izvajanjem meritve vmesnik prikazuje animacijo, ki je hkrati indikator napredka. Za indikator napredka potrebujemo od komponente povratno informacijo med merjenjem. To lahko dosežemo z branjem standardnega izhoda komponente, po katerem bi se prenašala sporočila o napredku. Še boljša metoda bi bila izdelava .dll datoteke, ki bi jo klicali z Javo z uporabo JNI (Java Native Interface) [27]. Če nimamo možnosti pridobivanja povratne informacije med merjenjem, moramo čas merjenja predvideti in indikator napredka ustrezno premikati. Možno je tudi, da se merilniki med sabo lahko rahlo razlikujejo v hitrosti merjenja, zato moramo v tem primeru hitrost animacije vključiti v nastavitveno datoteko merilnika.

Ko se proces merjenja zaključi lahko vmesnik preveri izhodno kodo za morebitne napake. Možne napake so fizična ovira na poti merilne glave okoli nog obiskovalca, težki svetlobni

pogoji, odsotnost obiskovalca ali njegov nepravilni položaj, premikanje med meritvijo, itn. V tem primeru mora merilnik v naslednjem koraku prikazati obvestilo o neuspešni meritvi. Tam bo obiskovalec imel tudi možnost ponoviti meritev.

V primeru uspešne meritve moramo prenesti izmerjene informacije v grafični vmesnik. To lahko storimo z besedilno datoteko, ki jo ustvari proces za merjenje. Alpinin merilnik poleg osnovnih podatkov, kot so dolžina in širina ter obsegi na različnih mestih stopala, poskrbi še za zaporedje točk, ki predstavljajo tloris in stranski ris izmerjenih stopal. Datoteko z rezultati ustvari komponenta `fs2.exe`, ki jo je izdelal dr. Matija Jezeršek. Poleg tekstovne datoteke s povzetki meritve `fs2.exe` ustvari še surovo datoteko izmerjenih 3D točk, ki se shrani za kasnejšo analizo v podjetju.

Slika 11 Izvedba meritve [26]

5.2.1.4. Ogled rezultata meritve (Slika 12)

Na rezultatu meritve si obiskovalec lahko ogleda podatke o dimenzijah svojih stopal. Obiskovalcu poleg numerične predstavitve prikažemo še tloris in stranski ris obeh stopal ter graf širine obeh stopal glede na povprečje.

Točke za tloris in stranski ris je potrebno normalizirati za prikaz v pripravljenem okvirčku. Obdržati moramo pravilno razmerje med stopali, hkrati pa jih želimo prikazati čim večje, da zapolnijo ves prostor, ki je na voljo. Za izris lika stopala lahko uporabimo kar vgrajeni funkciji razreda `Graphics`: `drawPolygon` in `fillPolygon`.

V primeru slabo zajetih podatkov, ki lahko nastanejo zaradi premikanja, lahko obiskovalec na tem zaslonu sproži ponovno merjenje.

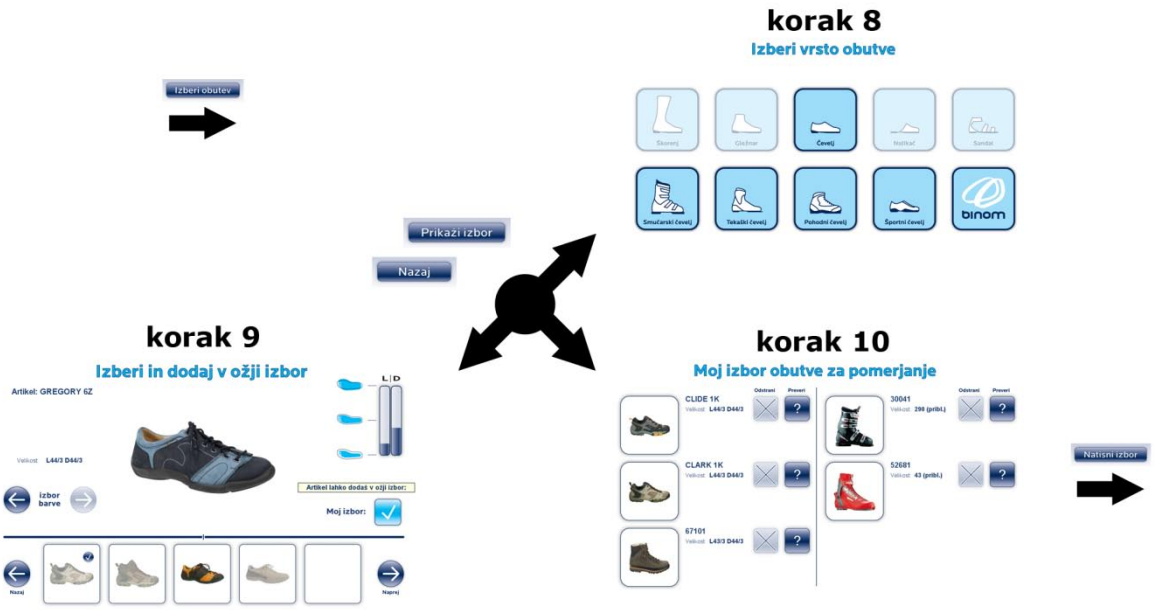
Čas, ki ga obiskovalec uporabi za ogled podatkov, izkoristimo za izvajanje parih operacij v ozadju. V tem trenutku imamo namreč vse podatke o obiskovalcu, zato moramo pripraviti prilagojene podatke o kolekciji. Najprej izločimo neustrezne izdelke, preostale pa primerjamo z izmerjenimi stopali, jih razporedimo v skupine in jih uredimo v pravilni vrstni red. Pripraviti je treba tudi slike za animacijo. Več o teh postopkih v poglavjih 5.2.4 in 5.2.5.

Slika 12 Ogled rezultata meritve [26]

5.2.1.5. Pregledovanje kolekcije (Slika 13)

Po opravljeni meritvi želimo obiskovalca popeljati skozi kolekcijo in ga navdušiti nad kakšnim modelom. Seznam izdelkov je lahko precej dolg, zato izdelke najprej razdelimo v skupine. Po izboru skupine sledi zaporedni pregled izdelkov iz te skupine, ki so delno razvrščeni glede na ustreznost za obiskovalca. Premikanje po izdelkih je animirano, saj takšna animacija pripomore k orientaciji uporabnika. Pri izdelku obiskovalec poleg slike vidi tudi predlagano velikostno številko in graf prileganja modela njegovim stopalom.

Obiskovalec se lahko hitro vrne na izbiro skupine, če si želi ogledati drugačen tip izdelkov.



Slika 13 Pregledovanje kolekcije [26]

Vsak izdelek, ki mu je všeč, lahko s klikom na gumb postavi v ožji izbor. Med izbiranjem lahko skoči na ogled izbranih izdelkov, kjer si jih ogleda ali izbriše. Da bi bil vmesnik čim bolj preprost, omejimo ožji izbor na en zaslon, na katerega gre do 6 izdelkov.

Ogled kolekcije in izbiranje izdelkov torej poteka z izmenjavo teh treh zaslonov. Ko se obiskovalec naveliča pregledovati izdelke, lahko zaključi ogled s tiskanjem podatkov o meritvi in izdelkov v ožjem izboru.

5.2.1.6. Zaključek in anketa (Slika 14)

S tiskanjem podatkov se postopek obiska merilnika lahko zaključi. Obiskovalec ima možnost izpolniti še anketni vprašalnik, če mu to ni odveč.

Na zaključnem izpisu na papirju vidi imena izbranih modelov in njihove velikosti, ki jih lahko v prodajalni takoj preizkusi ali kupi. Hkrati ima napisano tudi kodo meritve, ki mu omogoča ogled njegovih stopal na spletni strani.

Slika 14 Zaključek in anketa [26]

5.2.2. Programska struktura čarovnika

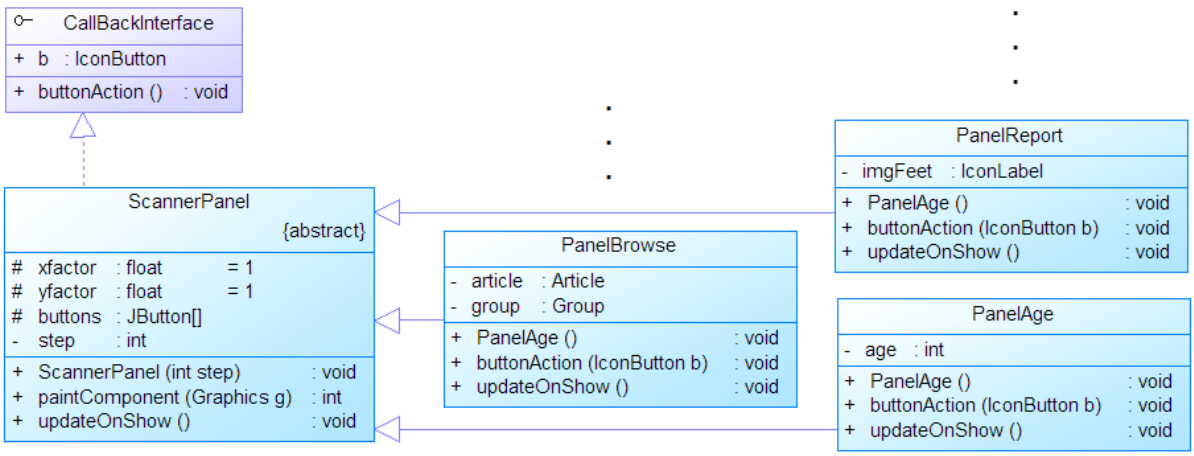
Za izdelavo čarovnika moramo najprej poiskati skupne elemente vseh korakov. Tipični skupni gradniki čarovnika so oblika okvirja čarovnika z ikono ali logotipom, prostor za gumbe za navigacijo in status položaja v postopku. Z definiranimi skupnimi gradniki lahko postavimo osnovni abstraktni razred generične podstrani, katerega bodo dedovale vse podstrani postopka (Slika 15).

Pri izbiri funkcionalnosti osnovnega objekta je pomembno ravnovesje med poenotenjem in omogočanjem izjem. Če bomo preveč poenotili podstrani, pridemo do težave pri podstraneh, ki potrebujejo kakšno posebnost. V nasprotnem primeru bomo podvajali kodo in s tem vnesli možnosti za napake.

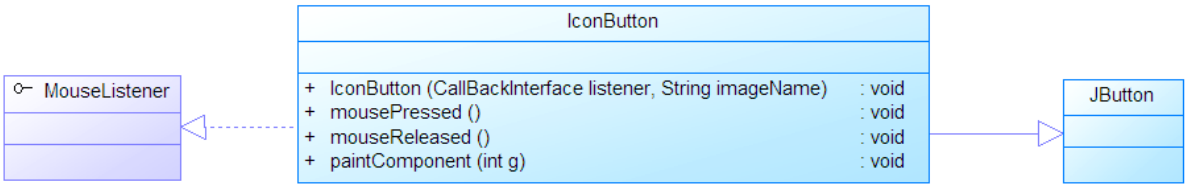
Abstraktni razred vseh podstrani čarovnika naj vsebuje metode za nalaganje slik vmesnika, za izris okvirja in abstraktne metode za lovljenje dogodkov (`updateOnShow`). Ob aktivaciji podstrani ta razred avtomatično izriše ozadje okvirja in nanj doda sliko trenutnega koraka. Konstruktor klicane podstrani doda na ekran vse potrebne gumbe in ostale kontrole, ki jih potrebuje za prikaz svoje informacije. To naj se zgodi ob zagonu merilnika za vse korake, saj se bodo podstrani po zagonu prikazale velikokrat in se želimo izogniti ustvarjanja podstrani za vsak prikaz. Dovolj je, da se ob prikazu podstrani sproži ustrezna metoda, ki v komponente naloži aktualno vsebino. Abstraktni razred hrani še podatke o velikosti aplikacijskega okna, faktorjih resolucijskega raztega in podobne spremenljivke in konstante skupne vsem podstranem.

Poenotimo lahko tudi izdelavo gumbov z novim razredom `IconButton` (Slika 16). Ob kreiranju mu podamo objekt z metodo, ki se bo klicala ob dogodkih na gumbu, ter ime datoteke s sliko, ki jo bo gumb prikazoval. Metodo zahtevamo s preprostim vmesnikom, ki ga mora podan razred implementirati. Gumb potem poskrbi za nalaganje ustreznih slik za

sproščeno in pritisnjeno stanje v pravem jeziku ter za ustrezno odzivanje na prestrežene dogodke.



Slika 15 Dedovanje podstrani



Slika 16 `IconButton`

5.2.3. Resolucija zaslona

Pisani grafični vmesniki so velikokrat sestavljeni iz bitne grafike. Za razliko od vektorske grafike je bitna grafika vedno v naprej pripravljena v določeni velikosti, točkah, in jo pri povečevanju/manjšanju popačimo.

Stoječi merilniki za prodajalne imajo trenutno vsi enako resolucijo, težko pa predvidimo, da bo tudi v prihodnosti tako. Hkrati mora aplikacija delovati tudi kot prenosna različica merilnika, ki za strojno platformo uporablja poljubni prenosnik. Če želimo aplikacijo imeti čez celotni zaslon, lahko pričakujemo različne resolucije in razmerja velikosti stranic zaslona.

Različna razmerja stranic najpreprosteje rešimo z vpeljavo črnih robov na ustrezni strani. Namesto črnine bi lahko tudi podaljšali del vmesnika, ki nima uporabne funkcije. V primeru Alpinskega merilnika smo se odločili za pripravo vsega materiala za zaslone na končnih merilnikih v prodajalnah. Na teh mestih se bo z napravo srečalo največ obiskovalcev, zato je to naša primarna platforma. Za uporabo na prenosnikih s širokim zaslonom dodamo črna robova levo in desno, ki sta se izkazala za zelo nemoteča.

Tudi resolucijsko bodo slike pripravljene samo za primarno platformo merilnika v prodajalni. Na ostalih resolucijah se bodo slike dinamično povečevale med izrisovanjem. Java omogoča precej nastavljivo kvaliteto tega procesa preko `RenderingHints` [28] parametrov objekta `Graphics2D`. Povprečni računalnik je lahko uporabil natančno dinamično povečavo z bilinearno [29] interpolacijo, ki je dala zelo dobre rezultate. V primerjavi s še boljšo bicubic [30] interpolacijo je bila razlika skoraj neopazna. Za zelo počasne računalnike dodamo še nastavitveno možnost interpolacije najbližjega soseda (nearest-neighbor) [31], ki vmesnik vidno popači, ampak omogoča gladko delovanje na poljubnem zaslonu.

Problem bi lahko reševali tudi s pripravo bitnih grafik za več predvidenih resolucij, vendar ima velika večina merilnikov enako velikost zaslona in resolucijo. Vse druge velikosti se bodo uporabljale samo začasno in tako ne upravičijo dodatnega časa potrebnega za razvoj prikaza in izdelavo grafik.

5.2.4. Optimizacija izbiranja modelov za prikaz

Preprost vmesnik z velikimi ikonami in omejeno količino informacij je idealen za neukega uporabnika z na dotik občutljivim zaslonom. Ni pa to dobra izbira za prikaz velikega števila izdelkov. Neustrezne modele je pametno izločiti, da ne zasedajo dragocenega prostora na zaslonu. Možnih filtrov za izločanje in urejanje je več, morajo pa biti fleksibilni, saj se morajo podrediti dinamičnim zahtevam prodajne strategije podjetja.

Postopka se lotimo z branjem vseh izdelkov v seznam iz katerega potem izločamo neprimerne z več prehodi različnih kriterijev. Glavne operacije, ki jih potrebujemo so:

- dodajanje na konec seznama,
- listanje seznama od začetka do konca,
- listanje seznama od začetka do konca z vmesnim brisanjem ustreznih elementov in
- urejanje preostalih elementov po poljubnih kriterijih.

Najboljša izbira strukture za te postopke se izkaže `LinkedList`. Sposobna je namreč v konstantnem času izvajati prve 3 zahtevane operacije, hkrati pa je idealno združljiva z metodo `Collections.sort()`, ki uporablja urejanje z zlivanjem (merge sort), pisano na kožo strukturam z zaporednim dostopom [32,33]. Če jo primerjamo s strukturama `Vector` in `ArrayList` ugotovimo, da sta slednji veliko počasnejši pri brisanju elementov na tak način. Njuna prednost, ki je v konstantnem naključnem dostopu, pa se nikoli ne uporabi. Struktura `HashSet` prav tako zagotavlja konstantni čas prvih treh zahtevanih operacij, vendar je zaradi razpršilne funkcije rahlo počasnejša od `LinkedList`. Glavna njena slabost pa je, da ni kompatibilna z metodo `Collections.sort()`.

Da vse našteto drži, moramo `LinkedList` pravilno uporabljati. Za listanje po seznamu ne smemo uporabljati metode `.get( index )`, ki nima konstantnega časa, temveč `iterator()` oziroma zanko `foreach`. Za listanje z brisanjem moramo uporabiti iteratorjevo funkcijo `remove()`.Ta namreč edina zna odstraniti trenutni element brez prekinitve listanja z izjemo `ConcurrentModificationException`.

Ko opravimo z izločanjem in sortiranjem, se je pametno strukture `LinkedList` znebiti. Pri brskanju kolekcije izdelkov v nadaljevanju bomo zagotovo veliko uporabljali naključni dostop, ki je pri `LinkedList` v najslabšem primeru `O(n)`. Izdelke lahko prepišemo v `Vector` ali celo navadno tabelo, saj je njihovo število zdaj že znano. Strukturi `ArrayList` se zaradi večnitnostnega dostopa uporabniškega vmesnika raje izognemo, saj njene operacije niso sinhronizirane [34].

Vse te operacije opravimo takoj po izmerjeni meritvi in izkoristimo čas, ko uporabnik pregleduje rezultate. V tem času dodamo na vmesnik majhno obvestilo o nalaganju izdelkov in onemogočimo gumb za nadaljevanje, dokler nimamo prilagojenega seznama izdelkov z vsemi podatki.

5.2.4.1. Izločanje po spolu

Prvo izločanje, ki ga lahko izvajamo, je izločanje po spolu obiskovalca. V prvih korakih merjenja smo od njega pridobili informacijo o spolu, zato lahko izločimo vse izdelke, ki so

narejeni specifično za nasprotni spol. To izločanje lahko izvajamo že pri branju izdelkov, vendar s tem pridobimo zelo malo, izgubimo pa centraliziranost vseh izločevalnih pogojev.

5.2.4.2. Izločanje izdelkov izven sortimenta

Zaradi opravljene meritve poznamo tudi okvirno velikost obiskovalčeve noge. Takoj lahko izločimo vse modele, ki se v takšni velikosti sploh ne izdelujejo. Obiskovalčeva velikostna številka je torej izven sortimenta. Na ta način poskrbimo za pravilno ločevanje otroške in odrasle obutve. Tudi to izločanje pogojno lahko izvajamo že ob branju izdelkov.

5.2.4.3. Izločanje izdelkov glede na trenutno zalogo prodajalne

Omejevanje kolekcije glede na trenutno zalogo prodajalne je že bolj dvomljivo in odvisno od situacije. V nastavitve programa je pametno dodati opcijo vklopa ali izklopa izločanja po tem kriteriju, saj merilnik ne bo vedno imel dostopa do podatkov o zalogi. V primeru Alpine imamo opravka z zelo velikim številom različnih izdelkov, katerih pregled bi kljub delitvi v skupine bil otežen. Hkrati za stranko ni prijetno, da najde čevelj, ki ji je zelo všeč, nato pa izve, da ga ne more kupiti. Za merilnike v prodajalnah smo torej predvideli izločanje izdelkov, ki jih ni na zalogi, medtem ko za mobilne merilnike zaloge ne preverjamo.

Preverjanje zalog je lahko precej zapleten postopek. Nikakor ni sprejemljivo, da bi podatke morali prodajalci vnašati ročno. V primeru prodajaln Alpine imamo podatke o zalogi shranjene na računalniku, ki se uporablja za blagajno. Program za prodajo je zastarel in je izdelan v še programskem jeziku Clipper. Podatke shranjuje v datotekah formata dBase, ki jih lahko pripravimo za skupno rabo v omrežju z bralnimi pravicami. Merilniku nastavimo pot do datoteke z zalogo, ki jo prebere in izloči neobstoječe izdelke. Za branje dBase datotek lahko uporabimo knjižnico JavaDBF [35].

Poleg izločanja modelov, ki jih prodajalna sploh ne prodaja, pa smo lahko v tem procesu še bolj selektivni. Iz datoteke o zalogi je razvidna zaloga za vsako velikostno številko, v procesu meritve pa smo pridobili velikostno številko obiskovalca. Tako lahko izločimo tudi vse modele, pri katerih je ustrezna velikostna številka trenutno razprodana.

Da bi bila ta operacija čim hitrejša, jo moramo izvajati z linearnim branjem datoteke z zalogami. Druga možnost bi bila linearno branje izdelkov in naključni dostop do dBase datoteke zalog, kar se ne bi dobro končalo. Optimizirati moramo torej prvo možnost. Naključnega dostopa do našega seznama ne moremo pričakovati, saj ni naslovljiv s šifro izdelka, ki jo bomo našli v datoteki zalog. Lahko se vedno znova sprehodimo po seznamu od začetka do konca in iščemo pravi izdelek, vendar bi bilo to pri velikem številu zapisov zalog zelo počasno. Zapisov zalog pa je dejansko veliko, saj imamo za vsak model in vsako velikostno številko zapisano število parov na zalogi. Rešitev najdemo v slovarju šifer izdelkov, ki ga ustvarimo ob branju iz kolekcijskih datotek. Zanj uporabimo strukturo `HashMap` s ključem `String` za šifro in vrednostjo tipa `Izdelek`. Tako pridemo do linearnega dostopnega časa do vseh izdelkov preko šifre z minimalnim dodatnim časom pri branju izdelkov.

5.2.4.4. Izločanje glede na natančnost prileganja (fit)

Definicija natančnosti prileganja se začne že pred obiskovalčevo meritvijo. Če hočemo primerjati in ocenjevati prileganje določenega modela z obiskovalčevo nogo, moramo najprej poznati velikosti in parametre samega modela. Natančnost bo odvisna od podatkov, ki jih imamo o modelu. Če imamo samo podatek o velikostni številki, potem samega prileganja ne moremo izvajati, lahko pa vseeno svetujemo predvideno najprimernejšo velikostno številko.

Boljše primerjanje lahko dosežemo z merjenjem notranjosti modela za vsako velikostno številko. Upoštevamo lahko tudi materiale, iz katerih je model izdelan, saj ti poleg dejanskih mer bistveno vplivajo na udobnost. Zbrani podatki nam lahko služijo za precej dobro oceno ustreznosti modela, zato lahko govorimo o izdelkih z veliko boljšo podporo primerjanju.

Dobre podatke za primerjanje lahko pridobimo tudi iz kopit, ki so bila uporabljena za izdelavo posameznih modelov. Teoretično je oblika kopita tudi idealna oblika noge za posamezni izdelek, vendar vse zapletejo še izbrani materiali in način izdelave. Metoda ima prednosti predvsem v tem, da lahko podatke o kopitih uporabimo za več modelov, saj si veliko modelov deli isti tip kopit. Za pridobivanje podatkov na tak način potrebujemo dejanska kopita, ki niso na voljo pri izdelkih, ki jih samo preprodajamo (dokup). Izdelki s podatki pridobljenimi po tej metodi spadajo v skupino z dobro podporo primerjanju.

Prva stopnja izločanja glede na natančnost prileganja je torej stopnja zanesljivosti podatkov za podporo primerjanju posameznih modelov. Strogost tega filtra mora biti nastavljiva, saj je od okoliščin uporabe merilnika odvisno kako zanesljivo mora biti primerjanje. Če želimo dobre rezultate prileganja lahko obdržimo samo izdelke z ustrezno natančnimi podatki, če pa želimo čim večji razpon kolekcije, pa obdržimo tudi izdelke brez podatkov, ki jih svetujemo le približno glede na izmerjeno velikostno številko stopal obiskovalca.

Druga stopnja izločanja glede na natančnost prileganja temelji na podlagi dejanskega primerjanja med podatki o modelu in izmerjenimi podatki o stopalu obiskovalca. Neprimerne modele lahko preprosto izločamo, delno primerne pa ustrezno označimo in uredimo. V primeru Alpininega merilnika podatke prileganja dobimo z uporabo komponente matching.exe, ki jo je napisal dr. Matija Jezeršek. Grafični vmesnik mora najprej sestaviti seznam izdelkov, ki pridejo v poštev za primerjanje. Seznam naredi iz prebrane kolekcije, po možnosti že zmanjšane z vsemi zgoraj opisanimi metodami. Komponenta matching.exe vzame podatke o meritvi in seznam izdelkov ter sestavi datoteko z rezultati prileganja z vsemi podanimi izdelki. Grafični vmesnik nato prebere to datoteko in na njeni podlagi označi ali odstrani izdelke po kvaliteti prileganja.

Na merilniku Alpine smo se odločili za grafični prikaz prileganja v obliki preprostega grafa, ki pokaže, koliko je noga preširoka oziroma preozka. Za izdelke brez podatkov je ta graf prazen, hkrati pa so označeni z opozorilom, da je predlog velikostne številke zgolj približen.

5.2.4.5. Razvrščanje preostalih izdelkov

Preostale izdelke lahko razvrščamo po poljubnih ključih, ki narekujejo, kateri izdelki se bodo obiskovalcem prikazali prvi. Seveda se z oddaljenostjo od prvega mesta manjša tudi verjetnost, da bo obiskovalec izdelek sploh videl. Izdelke bi lahko razvrstili po vrsti kolekcije, prileganju, aktualnosti, zalogi, imenu, ceni, itn. Podatke za želene ključe moramo imeti v podatkovni bazi o izdelkih ali v rezultatih prileganja, drugače po njih ne moremo razvrščati. V primeru Alpininega merilnika smo se odločili najprej prikazati izdelke z dobro podporo primerjanju (dobrimi podatki o dimenzijah modela), te pa razvrstimo po ustreznosti za izmerjeni stopali. Preostale modele razvrstimo po sezoni in nato po abecedi.

Za izvajanje takšnega primerjanja uporabimo metodo `Collections.sort()`, ki ji kot parameter podamo še razred z vmesnikom `Comparator`, v katerem implementiramo poljubno zapleten filter.

5.2.5. Animacija pri pregledu kolekcije

Pregled in kolekcije lahko zastavimo na več načinov, vsem pa je skupno, da vseh izdelkov na enkrat ne moremo pokazati. Poskrbeti moramo za gladko menjavanje trenutne podmnožice in hkrati poskrbeti za jasno navigacijo, da uporabnika ne zmedemo.

Primer je zaporedni prikaz izdelkov na linearnem traku z oknom, ki ga uporabnik pomika levo in desno. Pri tem načinu k navigaciji uporabnika zelo pripomore animirano pomikanje okna izdelkov, saj to daje boljši občutek orientacije na traku.

Implementacija premikanja okna se še izboljša, če je premik nelinearen. To pomeni, da med pomikom na želeno lokacijo najprej pospešuje in nato zavira. Računanje položaja ločimo v ločeno nit, ki od vmesnika pričakuje indeks ciljnega izdelka, kateri naj bi bil v sredini okna. Iz trenutnega in ciljnega položaja ter internih stanj hitrosti in pospeška lahko nit povsem samostojno premika slike izdelkov. Za to uporablja osnovne fizikalne formule pospešenega gibanja. Upoštevamo tudi, da bolj ko je velika razlika med trenutnim in ciljnim izdelkom, hitreje moramo zavrteti trak, saj nočemo, da animacija uporabnika ovira.

S klikanjem na najbolj oddaljen izdelek v oknu trak zavrtimo precej hitro. Zaradi te hitrosti se lahko v kratkem času v animaciji izmenja veliko slik izdelkov. Če jih med tem še grafično obdelujemo, na primer s svetlenje (alpha) neizbranih izdelkov, se animacija lahko začne zatikati. Da bi to preprečili implementiramo prednalaganje vseh slik v normalni in osvetljeni različici v pomnilnik. Takšna animacija bo delovala veliko bolj gladko, kot če bi za vsak prikazan izdelek morali sliko sproti nalagati iz trdega diska in jo osvetliti. Operacijo prednalaganja izvedemo ob že omenjeni priložnosti, ko si obiskovalec ogleduje rezultate meritve.

5.2.6. Večjezična podpora

5.2.6.1. Prevodi besedila

Za dosego večjezičnosti programa moramo najprej biti sposobni zamenjati vso besedilo uporabniškega vmesnika. Pri manjših projektih se tega pogosto lotimo tako, da vsa besedila zamenjamo s spremenljivkami, katerih vrednosti nastavimo glede na izbran jezik vmesnika. Vrednosti spremenljivk za posamezen jezik lahko hranimo na več načinov [36]:

- vgrajene v izvorno kodo programa,
- v ločenih prevajalnih datotekah:
  - tekstovne datoteke (ini) preprostih parov ključ:vrednost (spremenljivka:prevod),
  - Gettext binarne datoteke (standard GNU),
  - XML datoteke svojega standarda ali enega izmed uveljavljenih XML standardov za zapis prevodov (Tbx, Tmx, Qt, …),
- v podatkovnih bazah.

Ti načini shranjevanja prevodov imajo vsak svoje prednosti in slabosti. Prevodi vgrajeni v kodo programa so najhitrejši, porabijo najmanj prostora, hkrati pa so tudi najtežji za popravljanje in dodajanje jezikov. Popravlja jih lahko samo programer, ki je redko tudi prevajalec. Tekstovne datoteke so preproste za urejanje in dodajanje; urejajo jih lahko tudi sami prevajalci. Paziti moramo le na pravilno in konsistentno uporabo kodnih tabel, saj lahko hitro pokvarimo posebne znake. XML datoteke so podobne, le da lahko vsebujejo meta podatke, manj je nevarnosti s kodnimi tabelami, so bolje berljive iz programskega stališča vendar slabše s strani prevajalca. Binarne datoteke Gettext ponujajo dobre performančne

lastnosti in nimajo problemov s kodnimi tabelami znakov, vendar za njihovo urejanje prevajalec potrebuje dodatna orodja. Vnos prevodov v podatkovno bazo je najbolj primeren za zelo dinamičen uporabniški vmesnik, ki se veliko spreminja, in se pogosteje uporablja pri programiranju spletnih strani.

Za prevod vmesnika za merilnik stopal, ki bo več let uporabljan le v parih jezikih, zadošča že prva možno vgrajenega prevoda. Če obstaja možnost dodajanja jezikov, pa lahko uporabimo preproste tekstovne datoteke ali močnejše XML prevode. V primeru dobre Gettext podpore izbranega programskega jezika pa pride v poštev tudi ta možnost.

5.2.6.2. Podpora prevodom besedil v Javi

Java podpira izdelavo prevodov z razredi `ResourceBundle` [37]. Vnaprej imamo pripravljeni dve implementaciji tega abstraktnega razreda:

- `PropertyResourceBundle` [38] za delovanje uporablja tekstovne datoteke v obliki javanskih lastnosti (properties), ki se končajo s končnico ».properties«. Vsaka vrstica vsebuje ključ in vrednost, s tem da je vednost lahko samo niz (`String`). Kodna tabela zapisa datoteke je vedno zahodnoevropski standard ISO-8859-1 [39,40]. Za vnos posebnih znakov moramo zato uporabljati unicode ubežne znake (unicode escape) \u, ki jim sledi unicode koda želenega znaka. Če želimo datoteko urejati v drugi kodni tabeli, lahko za prevajanje uporabimo orodje native2ascii [41], ki spremeni posebne znake v ubežne kode.
- `ListResourceBundle` [42] uporabimo z izpeljavo novega razreda, ki z implementacijo metode `getContents()` poda tabelo parov niza in objekta. Za razliko od prejšnje metode je tukaj lahko rezultat prevajanja katerikoli tip, ne samo niz. Težav s kodnimi tabelami nimamo, vsaj ne več kot pri običajni javanski izvorni kodi.

Tretja možnost je implementacija svojega podrazreda `ResourceBundle`, ki nam omogoča še večjo fleksibilnost pri ustvarjanju prevodov. Namenjena je predvsem izdelavi vmesnikov za uporabo drugih načinov shranjevanja prevodov.

V vseh primerih moramo datoteke za vsak jezik pravilno poimenovati. Med konec imena in končnico vstavimo podčrtaj in oznako lokalnosti (npr. »_sl_SI« ali »_en_US« ali samo »_sl«). Ob kreiranju objekta virov z metodo `getBundle()` se bo ustvarila veriga virov od najbolj do najmanj natančnega. Najprej se bo poskušalo naložiti razred s polnim imenom lokalnosti, nato tekstovno datoteko lastnosti s polnim imenom lokalnosti ter v tem zaporedju naprej z vedno krajšim imenom lokalnosti, dokler ne pridemo do osnovnega imena brez lokalnosti. Če na nekem nivoju obstaja razred, se datoteka z lastnostmi ne bo vstavila v seznam virov. Ob klicu metode `getString()` za prevajanje niza se bo po tej verigi poiskal najbolj natančen prevod.

V prevode lahko vstavimo označbe mest, ki jih kasneje zamenjamo s parametri. V Javi se za to operacijo lahko uporabi metodo `replaceAll()` razreda `String` (poljubni parametri), metodo `format()` razreda `String` (parametri v formatu `fprint`) ali pa metode razreda `MessageFormat`. Slednje so za prevode najmočnejše, saj v kombinaciji z razredom `ChoiceFormat` omogočajo uporabo pogojev ob različnih parametrih, npr. za številnost ali sklanjanje. Primer uporabe je dostopen na [43].

5.2.6.3. Prevajanje slik

Pri močno grafično oblikovanih uporabniških vmesnikih se hitro naleti na prvo težavo omenjenih besedilnih prevodov: besedila iz prevajalne datoteke se ne da grafično oblikovati

po zahtevah oblikovalca uporabniškega vmesnika. Vzrok so lahko posebne tipografije, oblike in barve besedila in obrob ter uporaba različnih efektov nad besedilom. Pogosto se je potrebno zatekati k grafični predstavitvi takšnih besedil, ki so praviloma majhna oziroma kratka. Hkrati to pomeni, da moramo ob vsaki spremembi besedila popraviti sliko, oziroma več slik za več jezikov. Za daljše ali dinamične vsebine si tega ne moremo privoščiti, medtem ko je za statično postavitev oblikovanja sprejemljivo.

Preprost primer izvedbe slikovnih prevodov je dodajanje jezikovne oznake k imenu datoteke, ali ločevanje datotek v imenike z nazivom jezika. Merilnik v Alpini je zelo grafično oblikovan, zato se je izbira prave slike vdelala v nalagalnik slik iz datotek. Ta iz nastavitev prebere izbran jezik in sestavi pravo ime datoteke, ki jo mora odpreti.

5.2.6.4. Lokalne značilnosti

Večjezičnost programa poleg prevodov obsega tudi značilnosti zapisa števil, datumov, valut, itn. Te značilnosti moramo upoštevati pri izpisovanju lokalno specifičnih podatkov, drugače lahko uporabnika zmedemo. V Javi se večino težav z lokalnim izpisom preprosto reši z uporabo ustreznega objekta `Locale`, ki ga znajo uporabljati vsi razredi, ki oblikujejo lokalno specifične podatke za izpis. Razred ustvarimo s pomočjo ISO 639-1 kode jezika [44], ISO 3166 kode države [45] in variante. Uporabimo lahko tudi samo kodo jezika oziroma kodi jezika in države. Posamezne kode ločimo s podčrtaji in jih v konstruktor podamo kot niz. Ustvarjeni objekt nato podajamo v oblikovalce izpisa, ali pa ga prijavimo kot privzeto lokalnost z metodo `Locale.setDefault()` [46].

5.3. Načrtovanje programskega vmesnika merilnika

Poleg grafičnega vmesnika mora aplikacija merilnika skrbeti tudi za nekatere operacije v ozadju. Te operacije so dovolj povezane z grafičnim vmesnikom, da jih ni imelo smisla ločevati v ločene aplikacije, ki bi medsebojno sodelovanje komponent kvečjemu otežilo.

5.3.1. Pisanje aplikacijskega dnevnika

Aplikacija, ki jo uporabljajo naključni mimoidoči, mora biti zelo robustna. Morebitne manjše napake mora skušati odpraviti brez obveščanja obiskovalca, medtem ko je pri večjih napakah smiselno le prijazno obvestilo o nedelovanju. Podrobnosti napake oziroma vzroki za nedelovanje se obiskovalcev ne tičejo, vseeno pa jih moramo nekje beležiti. Primerno mesto za to je datoteka z aplikacijskim dnevnikom.

V aplikacijski dnevnik moramo zapisati čim bolj podrobne informacije o tem, kaj se z merilnikom dogaja, da bi ob pojavitvi napake administrator imel dobro izhodišče za odpravljanje težave. V trenutku napake ga ne bo pri merilniku, hkrati se lahko zgodi, da prodajalka ne bo znala ponoviti napake, problematične noge pa bodo do takrat že odšle. Zabeležimo lahko uspešne in neuspešne meritve ter stanja meritvenih elementov med merjenjem, zagone merilnika, rezultate posodabljanja kolekcije in pošiljanja meritev na centralno lokacijo skupaj s stanjem omrežja, različna izločanja izdelkov zaradi nepredvidenih podatkov, oddaljene dostope, itn.

Uporabimo lahko obstoječo rešitev za pisanje aplikacijskega dnevnika, vendar za primer merilnika Alpine nismo našli nobene dovolj preproste in pripravne. Zato smo napisali svojo preprosto rešitev v obliki razreda s statičnimi funkcijami. Ob zagonu programa ga inicializiramo, nato pa med delovanjem kličemo tri metode za zapis v dnevnik, glede na resnost napake (info, warning, error). S pomočjo preoblaganja (overloading) pa te metode sprejemajo še razrede izjem (exceptions), ki v dnevnik zapišejo napako in sled programa ob

nastanku napake. Zaradi statičnosti metod in ene izhodne datoteke morajo te metode biti sinhronizirane za večnitno uporabo [47].

Pisanje dnevnika moramo tudi na nek način omejiti, saj nočemo, da se velikost datoteke nenadzorovano veča. V razred za pisanje aplikacijskega dnevnika smo tako vgradili še funkcijo avtomatičnega obračanja dnevniških datotek (log rotate) vsak mesec, ki hrani za eno leto aplikacijskega dnevnika v datotekah z avtomatično dodano pripono številke meseca.

Če uporabniku razreda omogočamo izbiro mape za hranjenje aplikacijskega dnevnika, je pametno preveriti, če mapa dejansko obstaja. V nasprotnem primeru jo lahko kreiramo, ali pa vrnemo ustrezni rezultat oziroma izjemo.

5.3.2. Oddaljeno posodabljanje

Programska oprema se bo v prihodnosti lahko še precej spreminjala in dopolnjevala. Postopek razpečevanja sprememb moramo čim bolj poenostaviti, hkrati pa obdržati možnosti za specifične verzije za posamezne merilnike. Možnost imamo, da se merilnik posodablja ob zagonu, ali pa posodobitve prožimo sami. Za primer Alpininega merilnika smo se odločili izdelati administracijski program, s katerim se bo administrator povezal na merilnik in mu določil verzijo programa oziroma posodobitve. Glavni razlog za to je, da želimo obdržati nekaj več kontrole nad posodobitvami in tudi časom posodabljanja. Ko bo programski vmesnik prešel v zrelo verzijo, ki bo imela le še redke posodobitve, lahko dodamo še opcijo avtomatike.

Posodobitev se prične z izdelavo posodobitvenega paketa. Izvedemo ga lahko kot stisnjen arhiv vseh novih datotek, z relativno potjo glede na korenski imenik aplikacije (root). Če torej želimo posodobiti le glavni program, v datoteko zapakiramo samo novejšo .jar datoteko aplikacije. Če nadgradnja potrebuje še kakšne nove zunanje datoteke (slike, zunanje .jar datoteke), pa v arhiv vključimo še te. Končni posodobitveni datoteki dodamo še .md5 datoteko s kontrolno vsoto arhiva in oboje pošljemo na oddaljeno lokacijo, do katere imajo merilniki dostop. To je lahko http ali ftp strežnik, kjer se nahajajo tudi aktualni kolekcijski podatki za merilnike. Izračun kontrolne vsote, kreiranje .md5 datoteke in pošiljanje na ustrezno mesto na spletu lahko združimo v preprosto aplikacijo, v kateri izberemo izvorno posodobitveno datoteko in dodamo naziv verzije ter kratek opis nadgradnje. Aplikacija nam sama izračuna kontrolno vsoto in jo skupaj s pravilno poimenovano posodobitveno datoteko pošlje na splet.

Naslednji korak je povezava na merilnik preko administracijskega programa (podrobneje opisanega v poglavju 5.5.1), s katerim sprožimo posodobitev na želeno verzijo.

Merilnik bi pričel s prenosom zahtevane datoteke, ki jo bi shranil v mapo »update«. Še pred tem bi moral nekam zapisati svojo namero skupaj z verzijo, na primer v tekstovno datoteko »todo«. Ta korak je potreben za primer večjih posodobitev na počasnejših internetnih povezavah, kjer se lahko zgodi, da se merilnik med prenosom ugasne. Nadgradnja se mora pri naslednjem zagonu torej nadaljevati, za kar služi datoteka »todo«. Po prenosu posodobitvene datoteke se naj prenese še .md5 datoteka. Z njo lahko preverimo pravilnost prenosa in morebitne prekinitve pri prenosu. Če kontrolna vsota ni pravilna, se postopek prenosa ponovi.

Ob pravilni kontrolni vsoti je vse pripravljeno za nadgradnjo. Operacije do sedaj se izvajajo v ozadju brez vplivanja na morebitno uporabo merilnika. Če je merilnik trenutno v uporabi, moramo s posodobitvijo počakati. Nit namenjena nadgradnji lahko čaka toliko časa, dokler merilnik ni vsaj par minut nedejaven na začetnem zaslonu. Ko se to zgodi, se lahko izvede nadgrajevanje.

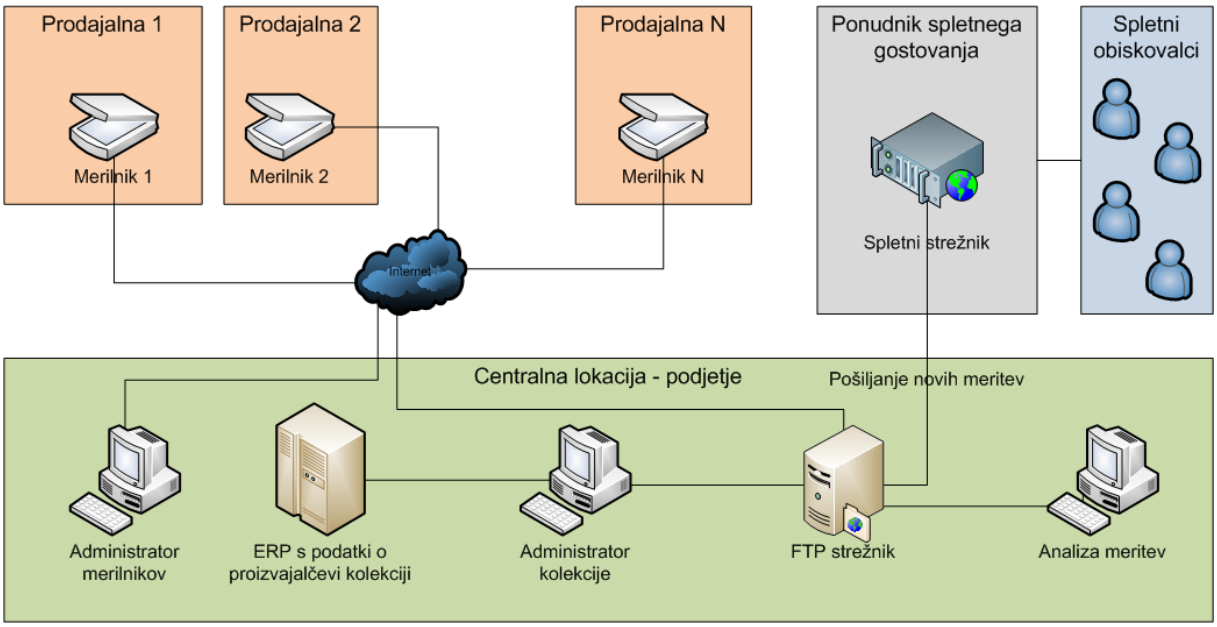
Nadgradnjo moramo izvajati v ločeni aplikaciji, saj se lahko pojavijo težave pri prepisovanju datoteke, ki se trenutno izvaja. Ta aplikacija počaka, da se uporabniški vmesnik zapre, nato pa odpakira vse arhivske datoteke iz »update« mape v korensko mapo aplikacije. Ob zaključku mora ponovno pognati uporabniški vmesnik merilnika. V času nadgradnje na zaslonu prikazujemo obvestilo o poteku nadgrajevanja, kljub temu, da je opisan postopek največkrat zelo kratek.

5.4. Načrtovanje prenosov podatkov v mreži merilnikov

5.4.1. Povezava komponent celotnega sistema

Delujoč merilnik je le začetna točka sistema. Za celovito delovanje potrebujemo še številne komponente, ki morajo delovati povezano med sabo (Slika 17). Za zagotavljanje kolekcijskih podatkov, fotografij, primerjalnih datotek in tudi za samodejno pošiljanje izmerjenih meritev potrebujemo povezavo merilnikov s centralnim datotečnim strežnikom. Ker so merilniki lokacijsko zelo raznoliko postavljeni, je tukaj potrebna čim bolj standardna rešitev za prenos podatkov. Primer takšnega protokola je FTP, ki s svojim pasivnim načinom [48] uspešno zagotavlja prenose preko morebitnih požarnih zidov, hkrati pa je preprost in uveljavljen način prenosa datotek, ki ga podpirajo številne programske knjižnice. Za zagotavljanje varnosti ob FTP prenosih lahko uporabimo manj standardno različico varnega SFTP protokola ali pa se zanašamo na VPN povezave.

Za centralni nadzor nad merilniki se potrebuje nadzorno aplikacijo, ki ima možnost dostopa do posameznega merilnika. Takšna aplikacija se navadno nahaja pri administratorju merilnikov v centralnem podjetju. Za pošiljanje ukazov in informacij med nadzorno aplikacijo in merilnikom smo sestavili preprost mrežni protokol podrobneje opisan v poglavju 5.5.1 Administracija merilnikov.



Slika 17 Pregled komponent mreže merilnikov in povezanih sistemov

Sledi nadzor nad kolekcijo na posameznih merilnikih, ki ga opravlja zaposleni v podjetju preko namenske aplikacije. Osnovo mu predstavljajo podatki iz proizvajalčevega ERP sistema, z aplikacijo pa lahko dopolni še morebitne manjkajoče informacije. Izdelane

kolekcijske datoteke aplikacija odloži na FTP strežniku, kjer počakajo samodejno osveževanje merilnika.

Izmerjene meritve, ki so jih merilniki samodejno poslali na FTP strežnik lahko zaposleni v podjetju uporabljajo v različnih analizah in poročilih.

Za ogled meritve na spletu potrebujemo še prenos meritev na spletni strežnik. Zajeta meritev je prevelika in preveč natančna informacija za spletnega obiskovalca, zato se na spletni strežnik lahko pošilja le primerne povzetke. Tukaj lahko izberemo implementacijo z navadnim POST zahtevkom ali pa uporabimo močnejše spletne servise, kot je WSDL [49]. Meritve želimo na spletno stran pošiljati iz centralne lokacije in ne iz merilnikov. S tem lažje držimo konsistentnost centralne lokacije in spletne strani. Pri posodabljanju te komponente pa nam ni potrebno posodabljati vseh merilnikov.

5.4.2. Osveževanje kolekcijskih datotek

Če hočemo na merilniku brskati po kolekciji izdelkov moramo to najprej imeti pri sebi. Zaradi dinamične postavitve merilnikov na lokacije brez ali z nezanesljivim internetnim dostopom, se ne moremo zanašati na prenos kolekcije v živo ob pregledovanju. Kolekcija mora biti na nek način shranjena v merilniku in se posodabljati, ko internetna povezava to dopušča. Možnih je več načinov shranjevanja:

- besedilna datoteka z ločilnimi znaki,
- XML datoteka ali
- vgrajena podatkovna baza.

Od naštetih je najmanj primerna uporaba podatkovne baze, saj preprost prikaz kolekcije na omejenem prostoru zaslona na dotik ne bi izkoristili nobene njene prednosti, kot so močna podpora spreminjanju podatkov, hitrost in prilagodljivost. Podatki o kolekciji za vsak posamezni merilnik bodo vedno točni na strežniku in prenos podatkov iz merilnika na strežnik ne bo nikoli potreben. Baza kolekcije prav tako ne more biti tako velika, da bi bil potreben inkrementalni prenos, saj bi bila v tem primeru preobsežna za brskanje na merilniku. Najpreprostejši način bi zato bil kar prenos kolekcijskih datotek s strežnika, kar bi ustrezalo prvima dvema načinoma shranjevanja kolekcije. Med njima za naše potrebe ni bistvene razlike. Novejši način z uporabo XML datotek prinaša bolj razširljivo strukturo, določa pravilni zapis posebnih znakov in je na splošno primernejši za prenos podatkov med različnimi subjekti, ki te podatke uporabljajo z različnimi programi. Če ne pričakujemo sprememb v strukturi podatkov te prednosti niso bistvene. Poleg tega je tekstovna datoteka manjša in za neizkušenega uporabnika preglednejša. V kolikor pa obstaja možnost spremembe in dopolnitve podatkov, pa moramo zastaviti zelo fleksibilno strukturo tekstovne datoteke, oziroma boljše, uporabimo XML datoteke.

Poleg podatkov o kolekciji so za njen prikaz potrebne tudi slike izdelkov. Z istim razlogom o nestalni internetni povezavi se mora tudi baza slik nahajati lokalno na merilniku.

Za zagotavljanje prenosov obeh skupin datotek uporabimo prej omenjeni FTP strežnik. Na njem se nahajajo kolekcijske datoteke za vsak merilnik posebej in slike vseh izdelkov. Merilniki se ob prisotni internetni povezavi periodično povezujejo na strežnik in posodabljajo svojo lokalno kopijo teh datotek. Najprej prenese datoteke kolekcije, ki so mu dodeljene, nato pa iz teh datotek sestavi seznam izdelkov, za katere nato prenese slike.

Dobra lastnost FTP strežnika, ki jo uporabimo tukaj, je hranjenje podatka o zadnji spremembi datoteke. S pomočjo tega podatka lahko izvajamo inkrementalno posodabljanje datotek, ki je

nujno predvsem pri prenosu slik izdelkov. Velikost vseh slik je prevelika za večkratne prenose na merilnike, če pa bi prenašali samo novo nastale datoteke, izgubimo možnost naknadnega popravljanja slik. Rešitev je torej primerjava datuma lokalnih datotek z datotekami na FTP strežniku in prenos samo novejših datotek.

5.4.3. Prenašanje meritev na centralno lokacijo

Rezultat vsake opravljene meritve na merilniku je sestavljen iz:

- surove 3D datoteke stopal,
- datoteke s podatki o izmerjeni nogi skupaj z informacijami, ki jih je obiskovalec vnesel med merjenjem,
- datoteka s seznamom izdelkov, ki jih je obiskovalec postavil v ožji izbor ob pregledovanju kolekcije,
- datoteka z rezultatom ankete, če jo je obiskovalec rešil.

Merilnik te datoteke shrani v arhivsko datoteko, ki jo shrani v izbran imenik na disku merilnika. Ime datoteke lahko sestavimo iz lokacije merilnika (zapisane v nastavitveni datoteki) in časovnega žiga trenutka izvajanja meritve. S tem je zagotovljeno unikatno ime vsake meritve, razen v primeru napačne nastavitve dveh merilnikov, ki bi imela isto ime lokacije in bi naredila meritev v isti sekundi. Ta problem lahko rešimo s prijavljanjem merilnikov v sistem s parom njihovega IPja in nastavljene lokacije, ali pa le zmanjšamo verjetnost napake z dodajanjem naključnih znakov v ime datoteke. S sprotno analizo števila meritev po lokacijah bi hitro ugotovili, da je nek merilnik nastavljen napačno.

V ozadju aplikacije merilnika ustvarimo servis z nizko prioriteto namenjen prenašanju teh meritev na centralno lokacijo. Ta ob zagonu in nato periodično med delovanjem pregleduje izbran imenik na disku ter pošilja nove datoteke na FTP strežnik. Ko je prenos uspešno zaključen označi arhivsko datoteko s preimenovanjem, na primer doda imenu datoteke besedo »_archive«, s tem datoteka ni več aktualna za prenos na strežnik. V primeru nenadne prekinitve delovanja merilnika ali internetne povezave se meritev ne izgubi, saj bo ob naslednji priložnosti še vedno bila aktualna.

5.4.4. Prenašanje meritev na spletno stran

Meritve iz vseh merilnikov so uspešno prenesene na centralno lokacijo v podjetju, kjer bodo uporabljene za analize, raziskave in razvoj. Takšne meritve so obsežne, saj vsebujejo surove 3D podatke oblike stopal. Datoteke niso primerne za direktno uporabo na spletni strani. Ko na spletno stran pride zahtevek za prikaz meritve, moramo za dejanski prikaz porabiti čim manj sredstev. To pomeni, da morajo meritve biti v naprej ustrezno pripravljene. Poleg tega bo redko centralna lokacija meritev dostopna spletnemu strežniku. V večini primerov bo spletni strežnik pri ponudniku spletnega gostovanja, medtem ko bo centralna lokacija meritev v samem podjetju proizvajalca oziroma prodajalca obutve.

Potrebujemo aplikacijo, ki periodično pošilja nove meritve na spletni strežnik. Takšno aplikacijo lahko namestimo na FTP strežnik, kjer se dejansko zbirajo meritve. Da ne bi omejevali izbire operacijskega sistema za FTP strežnik, tudi tukaj izberemo programski jezik Java. V primeru Linux FTP strežnika lahko periodično poganjanje aplikacije dosežemo z uporabo orodja **cron** [50].

Iz zahtev vemo, da na spletni strani potrebujemo sliko izmerjenega stopala in osnovne podatke iz meritve. Ti podatki se nahajajo v besedilni datoteki, ki jo vrne modul za merjenje

in je prisotna v arhivirani datoteki, preneseni iz merilnika. Aplikacija mora torej najprej odpreti arhivsko datoteko in iz nje izluščiti to besedilno datoteko.

Sledi pošiljanje te datoteke na spletni strežnik. Primeren način je uporaba HTML zahtevka POST s pripeto datoteko, lahko pa posežemo po bolj naprednih možnostih spletnih servisov WSDL [49]. Na spletnem strežniku je pametno omejiti dostop do strani za odlaganje meritev z vezavo na zunanji IP ali z uporabo avtentikacije.

Spletna stran mora prejeto datoteko ustrezno obdelati. Za hitrejši prikaz spletne strani je pametno, da slike stopal ustvarimo vnaprej. Iz prejetih podatkov spletna stran ustvari ustrezno sliko in jo shrani v podatkovno bazo skupaj s preostalimi podatki o meritvi. Tako je vse pripravljeno za hitro serviranje obiskovalcem spletne strani naravnost iz podatkovne baze na spletnem strežniku.

5.5. Načrtovanje programske opreme za administracijo merilnikov in kolekcije

5.5.1. Administracija merilnikov

Za upravljanje merilnika v prvi vrsti potrebujemo aplikacijo za oddaljen dostop do samega računalnika z merilnikom. S to aplikacijo lahko na merilniku opravimo vsa programska opravila, brez odhoda na lokacijo. Slabost uporabe teh aplikacij je ta, da v času popravljanja merilnika ta ni na voljo obiskovalcem. Za pregled stanja in osnovni nadzor zato predvidimo posebno aplikacijo, ki omogoča določena opravila v času, ko je merilnik na voljo za uporabo obiskovalcem.

Osnovni del aplikacije je seznam merilnikov, ki omogoča hitro povezovanje na izbrane merilnike. Dodamo kontrole za urejanje in dodajanje seznama ter shranjevanje seznama v datoteko ali podatkovno bazo, odvisno od predvidenega obsega merilnikov in administratorjev.

Komunikacija med izbranim merilnikom in administracijsko aplikacijo poteka preko TCP IP povezave, ki jo v Javi implementiramo s pomočjo razredov `Socket`. Za krajša sporočila bi bil primeren tudi UDP protokol, vendar bi morali izdelati preverjanje dostave, kar bi po nepotrebnem zapletlo aplikacijo. Zato se raje odločimo za zanesljiv (reliable) protokol TCP, ki že sam zagotavlja dostavo podatka.

Večina informacije, ki se bo prenašala med merilnikom in aplikacijo, bo v obliki vprašanje-odgovor v smeri od administracije proti merilniku. Zato ne potrebujemo naprednega asinhronega (nonblocking) strežnika. Dovolj bo zaporedna komunikacija, pri kateri imata obe strani odprt svoj bralni in pisalni kanal.

Sestavimo protokol s seznamom ukazov, ki jih administracijska aplikacija pošlje merilniku, ta pa odgovori s predvidenim odgovorom. Ukaze lahko oblikujemo kot XML elemente z imenom in seznamom parov imen parametrov in njihovih vrednosti v sledeči obliki:

```
<IME_UKAZA param_1='vrednost_1' param_2='…'/>
```

Odgovor nima posebej določenega formata, največkrat bo šlo le za vrednosti v predvidenem vrstnem redu, ki jih lahko ločujemo in zaključimo z znakom za novo vrsto.

Enkripcija protokola ni nujno potrebna, vendar jo zaradi preproste implementacije v Javi vseeno lahko dodamo. Gre namreč le za ustvarjanje ustreznih certifikatov ter dodajanje parih vrstic pri ustvarjanju `Socket` objektov [51].

5.5.1.1. Primer komunikacijskega protokola

Protokol za administracijo merilnikov obsega naslednje ukaze:

- **Pridobivanje različic programov**
  `<GETINFO/>`
  Vrne različico aplikacije merilnika in različici programov za izvajanje meritve in primerjanje ločene z znakom za novo vrsto. Vrednosti se zapišejo na pripravljena mesta v glavi podatkov o merilniku.
- **Pridobivanje aplikacijskega dnevnika za določen mesec**
  `<GETLOG month='st_meseca'/>`
  Vrne celotno vsebino datoteke aplikacijskega dnevnika za izbran mesec. Datoteko izpišemo v veliko tekstovno področje, kjer se lahko ogleda rezultat. Dodatno lahko preneseno log datoteko shranimo na lokalni računalnik.
- **Pridobivanje in shranjevanje nastavitvene datoteke**
  `<GETCONFIG/> in <SAVECONFIG/>`
  Vrne vsebino nastavitvene datoteke merilnika. Datoteko prikažemo v veliko tekstovno področje, kjer jo lahko popravimo, nato pa jo pošljemo nazaj merilniku.
- **Ponovni zagon merilnika**
  `<RESTART/>`
  Ukaz, s katerim lahko ponovno poženemo vmesnik merilnika. Primeren na primer za uveljavljanje sprememb v nastavitveni datoteki.
- **Posodobitev programa za posodabljanje**
  `<UPDUPDATER/>`
  S tem ukazom se merilnik poveže na FTP in prenese najnovejšo različico programa za posodabljanje. Ukaz je predviden zaradi možnih novejših verzij programa za posodabljanje, ki ga ne moremo zamenjati po običajni poti posodabljanja merilnika.
- **Posodobitev merilnika z novejšo verzijo**
  `<UPDATE version='ime_verzije'/>`
  Po prejemu tega ukaza merilnik prične proceduro posodabljanja, opisano v poglavju 5.3.2.
- **Posodabljanje kolekcije**
  `<UPDDB/>`
  S tem ukazom prekinemo spanje niti, ki je odgovorna za posodabljanje kolekcije. Tako povzročimo takojšnjo posodobitev kolekcije, če je potrebna.
- **Pridobivanje trenutnega stanja merilnika**
  `<GETPAGE/>`
  Ukaz vrne številko strani, na katerem se nahaja čarovnik uporabniškega vmesnika. Tako lahko hitro vidimo, če ima merilnik trenutno obiskovalca.
- **Prekinitev povezave z merilnikom**
  `<QUIT/>`
  Po prejemu tega ukaza merilnik zapre vzpostavljeno povezavo z administracijskim programom.

5.5.2. Administracija kolekcije

Za prikaz izdelka v zastavljenem sistemu merilnikov so potrebni naslednji podatki:

- naziv izdelka ali oznaka modela,

- vrsta obutve za osnovno delitev v grupe,
- oznaka spola – odrasli ali otroški, moški ženski ali uniseks,
- oznaka sortimenta – seznam velikostnih številk modela
- obstajati mora slika izdelka (na prednastavljenem mestu)
- različne oznake za obutev s posebnostmi (npr. modeli v več širinah)

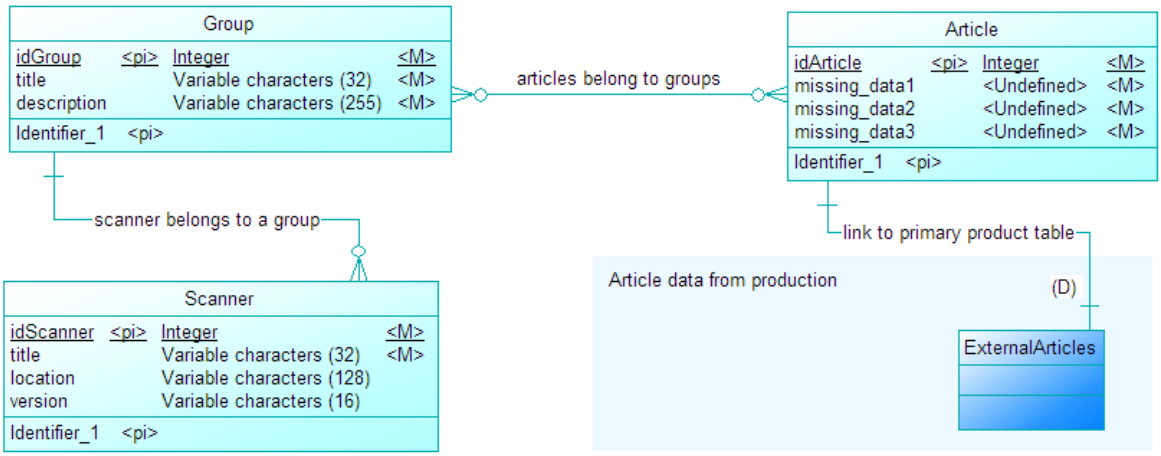
Proizvajalec v svojem podatkovnem sistemu verjetno nima vseh potrebnih podatkov ali pa so ti podatki v neustrezni obliki. Sistem podjetja Alpina ima drugačno delitev grup, ki implicitno zajemajo tudi podatke o spolu. Nekatere podatke lahko iz proizvajalčevega sistema preslikamo avtomatično, druge pa moramo za prenos vnesti ročno. Pri dodajanju podatkov imamo dve možnosti: lahko prilagodimo prodajalčev sistem opisa izdelkov in ga uporabljamo direktno, lahko pa naredimo dopolnilne tabele z manjkajočimi podatki in jih povežemo z obstoječimi podatki. Ker je poseg v sistem prodajalca največkrat nezaželen, bo pogostejša druga rešitev.

Potrebna je implementacija aplikacije z vmesnikom v obliki tabele vseh izdelkov. Manjkajoče podatke črpamo iz dopolnilnih tabel, in jih dopolnjujemo s podatki, ki v proizvajalčevem sistemu že obstajajo. Urejanje tabele izdelkov obsega dodajanje in brisanje izdelkov ter urejanje manjkajočih podatkov.

Pomembna je tudi faza preverjanja pravilnosti vnosov, predvsem obstoj vseh informacij in slik. Ne želimo namreč, da bi na merilnike poslali izdelke, ki bi lahko povzročili nepravilno delovanje.

Merilnike moramo združevati v skupine, ki imajo enako podmnožico izdelkov iz celotne kolekcije. Za obisk vojaškega sejma namreč potrebujemo na merilniku drugačno kolekcijo kot na zimskih sejmih ali v splošni prodajalni. Aplikacija mora torej omogočati izdelavo skupin in povezovanje merilnikov v skupine. Te pa moramo povezati z izdelki v kolekciji. Za hranjenje te informacije potrebujemo preprosto podatkovno strukturo, ki jo dodamo v proizvajalčev sistem za upravljanje podatkovnih baz, ali pa postavimo preprost namenski SUPB (Slika 18).

Zadnji korak aplikacije je pošiljanje kolekcijskih datotek na FTP strežnik za vsak merilnik, ki je član izbrane skupine. S takšno rešitvijo imamo izvedeno urejanje kolekcije na enem mestu za vse merilnike.



Slika 18 Primer dodatnih entitet za podporo administracije kolekcijskih podatkov

5.6. Načrtovanje programske opreme za obdelavo zbranih meritev

5.6.1. Knjižnica za napredno obdelavo meritev

Ko imamo delujoč sistem zbiranja meritev, se lahko lotimo obdelav podatkov. Rezultat delovanja mreže merilnikov je skupina stisnjenih datotek, ki jih merilniki pošiljajo na centralno lokacijo. Takšna skupina datotek je brez orodij precej neuporabna in neobvladljiva. Pričakujemo lahko zelo raznolike zahteve za obdelavo, ki bi potrebovale raznolika orodja. Zato raje najprej izdelamo knjižnico preprostih razredov in vmesnikov, ki vsebujejo značilne operacije, skupne vsem potencialnim orodjem. Z njihovo uporabo bi za vsako novo obdelavo morali napisati le majhen del programske kode.

Uporaba večine razredov je zamišljena z dedovanjem vsake aplikacije iz razreda, ki nam pokrije čim več operacij, ki jih potrebujemo.

5.6.1.1. Uporabniški vmesnik in aplikacijski dnevnik

Prva skupna točka vseh orodij za avtomatsko obdelavo je preprost uporabniški vmesnik, ki praviloma obsega le sprotne informacije o obdelavi. Iste oziroma podobne informacije se zapišejo tudi v aplikacijski dnevnik.

Prva možnost je kar izpis na standardni izhod. Gre za dobro izbiro pri obdelavah, ki se bodo opravljale brez uporabnikovega posega ali na operacijskih sistemih brez grafične podpore. Omejitve takšnega pristopa so v podajanju parametrov, saj po pričetku obdelave težko vplivamo na njen potek.

Druga možnost je ustvarjanje okna, ki vsebuje tekstovno polje, v katerega se vpisujejo praktično enake informacije, kot na standardni izhod oziroma aplikacijski dnevnik. Prednost tega pristopa je lepši izgled za končnega uporabnika, lažje kopiranje in pregled izpisanega besedila ter možnost vplivanja na obdelavo z dodajanjem kontrolnih gumbov.

Prvi razred torej pokrije obe možnosti prikaza stanja obdelave ter pisanje aplikacijskega dnevnika. Vsebuje tudi možnosti za dodajanje gumbov, na katere lahko prijavimo svoje metode ob dogodkih (event handlers). Glavni del razreda pa so metode, ki nam omogočajo hiter in preprost izpis na izbran vmesnik (standardni izhod, aplikacijski dnevnik, tekstovno okno).

5.6.1.2. Listanje in filtriranje datotek za avtomatsko obdelavo

Naslednji skupni del vseh obdelav bo zaporedni prelet podmnožice zbranih datotek. Razred najprej poskrbi za shranjevanje nastavitve mape za obdelavo, ki jo lahko pogosto menjamo, lepo pa je, da ostane v spominu zadnja uporabljena.

Ko imamo izbrano mapo velikega števila meritvenih datotek se hitro pojavijo zahteve za filtriranje pred obdelavo. Razred s svojo metodo filtriranja omogoča izločanje datotek po imenu datoteke (ki recimo že vsebuje lokacijo meritve in časovni žig). Lahko pa to osnovno metodo tudi prepišemo (override) in sprogramiramo svoje filtre datotek. Metoda se namreč kliče za vsako datoteko v izbrani mapi.

Razred nato naredi nit, ki se sprehodi po pridobljenem seznamu datotek in za vsako datoteko kliče metodo za obdelavo, ki jo moramo implementirati s kodo posamezne obdelave. Razred poskrbi tudi za štetje obdelanih datotek in dinamično računanje časa do konca obdelave, ki ga oceni od časa porabljenega v naši metodi za obdelavo.

5.6.1.3. Odpiranje arhivov datotek

Pogosta operacija pri avtomatskem listanju po zbranih meritvah je odpiranje arhiva in dostop do dejanskih datotek meritve. Ta razred je izpeljava prejšnjega, ki kot prvi korak obdelave odpre arhivsko datoteko in nam pripravi dostop do datotek v arhivu (stream).

5.6.2. Primeri uporabe knjižnice za napredno obdelavo meritev

5.6.2.1. Prenašanje meritev na spletno stran

Omenjeno aplikacijo smo že opisali v poglavju 5.4.4. Zato le na kratko poglejmo, kako nam lahko te knjižnice pomagajo pri njeni izdelavi.

Aplikacija teče na FTP strežniku, torej ne potrebuje grafičnega vmesnika. Uporabno pa je pisanje dnevnika, ki ga knjižnica zagotavlja. Prav tako potrebujemo prelet vseh datotek v mapi (prebrani iz nastavitvene datoteke) in njihovo napredno sortiranje. Vse to nam omogoča knjižnica, dopolnimo le izločanje, ki ga izdelamo z branjem in primerjanjem datoteke že poslanih meritev.

Sledi obdelava vsake meritve. Pri tej aplikaciji potrebujemo iz arhiva odpreti tekstovno datoteko z osnovnimi podatki o meritvi ter jo poslati na spletni strežnik. Knjižnica nam že pripravi vir te datoteke (stream) v arhivu, zato moramo sprogramirati samo še pošiljanje na spletni strežnik.

5.6.2.2. Pretvorba osnovnih podatkov o meritvah v podatkovno bazo

Drugi primer je aplikacija, ki iz zbranih meritev naredi podatkovno bazo, ki jo bomo prej ali slej potrebovali za lažjo obdelavo osnovnih podatkov.

Razrede knjižnice zložimo v podobno zaporedje, le da tokrat omogočimo še uporabniški vmesnik, saj bo podatkovno bazo ustvaril poljubni uporabnik na zahtevo. Vse do vira datoteke z osnovnimi podatki o meritvi nam opravi naša knjižnica. V tej aplikaciji torej sprogramiramo samo še zapis prebranih podatkov iz odprtega vira arhivske datoteke v podatkovno bazo.

Uporabimo knjižnice za dostop do Microsoft Access datotek .mdb, ki so dobro izhodišče za uporabo baze podatkov v Microsoft Office izdelkih. Access datoteko vnaprej pripravimo s tabelami, poizvedbami in poročili. Ustvarimo tabelo vseh meritev, vsaka pa ima še dve povezavi v tabelo vseh stopal (za levo in desno stopalo). Aplikacija bo pri dostopu do te datoteke nato osveževala podatke v teh dveh tabela.

5.6.2.3. Iskanje največjega stopala

Kot zadnji primer poglejmo še primer hitro spisane aplikacije v obliki poizvedbe nad arhivom baz meritev. Spet zložimo razrede knjižnic enako, da nam aplikacija ponudi že vir meritve v arhivski datoteki. Dodamo le kodo, ki primerja dolžino stopala vsake ponujene arhivske datoteke s trenutno največjim in na koncu izpiše rezultat.

5.6.3. Končna preprosta poročila o meritvah

Z aplikacijo za izdelavo Access podatkovne baze opisano v prejšnjem poglavju (5.6.2.2) pridemo do prilagodljive oblike podatkov o vseh meritvah. Že v Access-u lahko iz njih pripravimo številne povzetke in poročila, lahko pa jih uporabimo tudi v Excel-u za pripravo naprednih vrtilnih tabel. Takšna poročila pa lahko pregleduje tudi uporabnik brez specifičnega znanja o samem sistemu ali meritvah, torej so že primerni za predstavitev vodstvu. Iz njih hitro vidimo trende obiskov merilnikov, statistične povzetke raznolikosti velikosti stopal, starosti merjenih obiskovalcev, časa merjenja po posameznih starostnih skupinah, itd.

Uporabni so tudi za kontrolo sistema, saj pokažejo, če se v sistemu pojavljajo nepravilnosti, kot je npr. izpad podatkov za posamezne lokacije.

Z izvozom podatkov do orodij Microsoft Office smo zelo poenostavili nadaljnje obdelave, zavedati pa se moramo, da so podatki le osnovni. Za napredno obdelavo v smislu načrtovanja kopit iz povprečij izmerjenih stopal potrebujemo 3D podatke, ki se tudi nahajajo v arhivu meritvenih datotek, se pa ne morejo prenesti v Access .mdb datoteko. Za njihovo obdelavo še naprej uporabljamo knjižnico za napredno obdelavo meritev, ki ji dodamo še operacije za lažjo uporabo teh 3D podatkov.

5.6.4. Potreba po varovanju osebnih podatkov

Z zbiranjem podatkov v tem sistemu nismo v neskladju z zakonom o varstvu osebnih podatkov, saj obiskovalec ne odda nobenega podatka, po katerem se bi ga lahko identificiralo. Najbližje temu pride 3D meritev stopala, ki pa ni dovolj natančna in predvsem ni stalna za vsakega posameznika posebej, da bi se jo lahko obravnavalo kot biometričen podatek [52]. Pri hranjenju teh podatkov nam torej ni potrebno skrbeti za ustrezno varovanje osebnih baz podatkov ali izvajati anonimizacije [53].

5.7. Končna struktura paketov na primeru implementacije

Za organizacijo datotek izvorne kode v Javi uporabljamo pakete (packages). Vse do sedaj naštete funkcionalnosti lahko združimo v posamezne zaključene celote, ki predstavljajo posamezne pakete:

- **analyse** – aplikacija za presnemavanje shranjenih arhivov meritev iz centralnega FTP strežnika v lokalni imenik in njihovo pretvorbo v Accessovo mdb bazo za analize in izdelavo poročil (poglavje 5.6.2.2).
- **batch** – knjižnica za napredno obdelavo meritev, ki vsebuje razrede za hitro gradnjo specifičnih obdelav nad centralno lokacijo vseh meritev (poglavje 5.6.1).
- **clipperReader** – paket vsebuje objekte za delo z dBASE .dbu, .dbf in .dbx datotekami. Lahko je le vmesnik do zunanjih knjižnic, ali pa izdelamo razčlenjevalnik (parser) kar v celoti (poglavje 5.2.4.3).
- **dbHandler** – vsebuje objekte za predstavitev baze izdelkov in njihovo združevanje v skupine (grupe). Vsebuje tudi metode za polnjenje teh objektov iz datotek in primerjanje izdelkov med sabo.
- **dbUpdater** – vsebuje razrede za posodabljanje podatkov o zalogah v objektih za predstavitev kolekcije.
- **filesystem** – vsebuje razred FS, v katerem so implementirane preproste statične metode za delo z datotekami na disku (copyFile, createDir).
- **ftpOperations** – vsebuje razrede za delo s FTP povezavami. Za dejansko implementacijo protokola FTP ta paket uporablja knjižnico Jakarta Commons Net [54].
- **logger** – razred Logger za pisanje aplikacijskih dnevnikov (poglavje 5.3.1).
- **net** – vsebuje definicijo protokola za mrežno komunikacijo med administracijsko aplikacijo in merilnikom (poglavje 5.5.1.1).
- **releaseCreator** – aplikacija za izvoz nove verzije na FTP skupaj z izračunano kontrolno vsoto v datoteki .md5 (poglavje 5.3.2).
- **scannerAdmin** – aplikacija za urejanje kolekcije in njen izvoz na posamezne merilnike (poglavje 5.5.2).
- **scannerGUI** – glavna aplikacija grafičnega vmesnika merilnika (poglavje 5.2).

- **scannerStatus** – aplikacija za nadzor in pregled merilnikov iz centralne lokacije (poglavje 5.5.1).
- **scannerUpdater** – aplikacija za nadgradnjo datotek vmesnika merilnika (poglavje 5.3.2).
- **useful** – različni razredi za splošne operacije, kot so na primer pretvorbe sortimentov, standardna poimenovanja, pridobivanje verzij, pravila poslovanja, itn.
- **websubmit** – aplikacija za pošiljanje meritev iz centralne lokacije na spletni strežnik (poglavji 5.4.4 in 5.6.2.1).
- **zipper** – paket z razredom za delo z arhivskimi datotekami.

5.8. Načrtovanje spletne strani za ogled meritve

5.8.1. Izbira programskega jezika in ogrodja za izdelavo spletne strani

Za generiranje dinamične spletne vsebine prikaza meritev imamo na voljo širok nabor programskih jezikov. Zahteve narekujejo dokaj preprosto spletno stran, ki ne zahteva obdelave velikega števila hkratnih obiskovalcev, zato lahko v izbor poleg jezikov s prevajanjem vključimo še interprecijske oziroma skriptne jezike. Prav tako zaradi majhnih zahtev pametno izločiti jezike, ki bi v projekt pripeljali dodatne stroške. Ti so povezani s cenami razvojnih orodij, gostovanj spletne strani ter dodatnih izobraževanj razvojne ekipe.

Po teh kriterijih smo izločili drago Microsoftovo tehnologijo .NET in Adobov ColdFusion ter različne skriptne jezike, ki jih ne poznamo dovolj dobro (Perl, Ruby on Rails, …). Glavna preostala kandidata sta bila Java/JSP in PHP. Ker Java na podanih zahtevah ne prinaša bistvenih prednosti in ima hkrati nekaj dražje gostovanje, smo se odločili za PHP. Najpomembnejši razlog od vseh je boljše obstoječe znanje in izkušnje ter posledično hitrejša izdelava aplikacije.

Večina naštetih tehnologij za razvoj spletnih aplikacij ima možnost uporabe dodatnih ogrodij za razvoj (framework). Glavna naloga ogrodij je pohitriti razvoj z že izdelanimi pogostimi operacijami pri spletnem razvoju, kot so dostop do podatkovne baze, izdelava predlog s HTML kodo, upravljanje s sejami, piškoti, itd. Pogosto skušajo tudi standardizirati način izdelave aplikacije in omogočajo ponovno uporabo izdelane kode v novih aplikacijah [55,56].

Izbira PHP ogrodij je še posebej pestra. Na voljo je vse, od majhnih in preprostih dodatkov, ki vsebujejo samo podmnožico funkcionalnosti ogrodja, do obsežnih in zapletenih ogrodij, ki od razvijalca zahtevajo več uvajanja, kot sam programski jezik. Prav zaradi te zahtevnosti tudi pri izbiri ogrodja največ štejejo pretekle programerske izkušnje razvojne ekipe.

Kljub temu, da so osnovni pristopi med ogrodji podobni, pa se med sabo lahko ogrodja vseeno močno razlikujejo. Nekatera nudijo le osnovno pomoč pri gradnji aplikacije, medtem ko so druga lahko že skoraj pripravljena aplikacija, ki jo samo še spreminjamo. Pri slednjih gre bolj za sistem za urejanje vsebine (CMS). Bolj ko ogrodje določa končno zgradbo, hitrejša je lahko izdelava, vendar je težje skreniti iz zastavljenih okvirjev. Naletimo lahko na hude omejitve, zaradi katerih porabimo veliko časa s prilagajanjem na ogrodje oziroma prilagajanjem ogrodja samega.

Za spletno aplikacijo v primeru merilnika Alpine smo se odločili za močno in fleksibilno ogrodje Zend Framework [57] podjetja Zend Technologies Ltd. Gre za zelo prilagodljivo rešitev, ki se je pri ekipi v preteklosti izkazala za dobro osnovo zelo raznolikih spletnih

aplikacij. Celotno ogrodje je precej obsežno, vendar se lahko uporabi le poljubne dele oziroma komponente. Poleg izkušenj ekipe so prednosti Zend Frameworka še ustrezna licenca (temelji na BSD licenci), dobra in obsežna dokumentacija in močna skupnost uporabnikov. Za ogrodjem stoji priznano podjetje, ki izvaja podporo, tečaje, svetovanja in certificiranje strokovnjakov, kar daje izbiri določeno zanesljivost in zaupanje za prihodnost.

Zend Framework vsebuje vse sodobne pristope in pripomočke za gradnjo spletnih aplikacij. Naštejmo nekaj glavnih, ki jih bi lahko uporabili pri gradnji spletne strani za ogled meritve [58]:

- MVC – Model-View-Controller je sodobni pristop gradnje spletne strani, ki ločuje podatkovni model s poslovnimi pravili od uporabniškega vmesnika. Oba dela tako lahko razvijamo in testiramo ločeno z različnim tipom strokovnjakov. V spletni razvoj ta pristop prinese fleksibilnost pri vzdrževanju in spreminjanju vsebine v življenjskem ciklu aplikacije. MVC vsebuje tudi sistem za generiranje HTML kode s predlogami in postavitvami (template, layout). S pravili lahko določimo preslikavo spletnih naslovov v kontrolerje in akcije, kar polepša berljivost in izboljša najdljivost na spletnih iskalnikih [59,60].
- ORM – Object-relational mapping je način pretvorbe relacijskih podatkovnih baz v objektno orientirane programske jezike. Zend Framework podpira preslikavo tako tabel kot vseh možnih relacij med njimi. Hkrati pa omogoča tudi neposredni dostop do podatkovne baze za primere, kjer bi ORM bil prepočasen. Vse skupaj lahko z minimalno spremembami v kodi izvajamo na različnih sistemih za upravljanje s podatkovnimi bazami (SUPB) [61].
- Izdelava obrazcev – omogoča hitro gradnjo obrazcev, ki jim nastavimo pravila za preverjanje pravilnosti in način sporočanja napačnih vnosov.
- Podpora za internacionalizacijo (i18n) in lokalizacijo (l10n) vseh izpisov valut, datumov in mer, skupaj s podporo prevodov vsebine [62].
- Sistem za pošiljanje e-pošte, ki generira HTML sporočila.
- Sistem za avtentikacijo in avtorizacijo obiskovalcev ter upravljanje sej, ki hranijo podatke med sicer ločenimi zahtevami za strani na strežniku.

5.8.2. Načrtovanje preprostega in varnega sistema uporabnikov

V zahtevah smo določili, da uporabnika za ogled meritve ne bi obremenjevali z registracijo. Vsak obiskovalec spletne strani ima torej možnost vnosa kode opravljene meritve, ki mu omogoča vpogled v rezultate meritve.

Tak pristop ima dve posledici. Prva se nanaša na možnost, da obiskovalec izgubi papir s kodo meritve. Problem rešimo z opcijo vnosa e-naslova zraven kode meritve, ki jo poimenujemo prijava meritve na e-naslov. Če obiskovalec vnese svoj e-naslov, dobi avtomatično elektronsko pošto s podatkom o kodi meritve. Tako ima za prihodnost kodo shranjeno še nekje drugje, kot samo na listu papirja. Poleg tega dodamo na spletno stran možnost vnosa e-naslova, na katerega se pošlje seznam vseh kod meritev, ki so prijavljene na ta e-naslov.

Druga posledica je nevarnost neomejenega poskušanja anonimnega vnašanja kod meritev, ki bi ga lahko izvajal nekdo, ki želi odkriti meritve, ki niso njegove. Kode meritev si sicer ne sledijo zaporedoma, vendar bi z dovolj velikim številom poskusov vseeno lahko uganil »preostale« meritve. Problem lahko rešimo z omejevanjem števila poskusov vnosa kode na recimo 5 poskusov na uro iz posameznega naslova IP, hkrati pa lahko dodamo tudi varovalko števila vnosov iz vseh naslovov IP, ki bi preprečevala močnejše napade iz več različnih virov.

Pomaga pa že dobro beleženje in nato analiza poskusov vnašanja meritev, ki jo izdelamo že zaradi zanimivosti iz poslovnega vidika.

Na podoben problem naletimo tudi, če želimo omogočiti ogled meritev preko naslova URL (GET request). Glavni namen tega je možnost obiskovalca, ki je odprl svojo meritev, da kopira in prilepi naslov trenutne strani prijatelju. Za to nalogo ob pošiljanju meritve na strežnik generiramo posebno 32 mestno kodo (hash – md5), ki unikatno določa vsako meritev. Ta koda je potem del naslova URL za ogled meritve. Zaradi ogromnega števila različnih možnih kod (2^{128}) ni mogoče, da bi nekdo ugibal te kombinacije [63].

5.8.3. Identifikacija nalog spletne strani iz zahtev

Z upoštevanjem zahtev spletne strani za ogled meritve dobimo skupino nalog, ki jih mora aplikacija pokriti (Slika 19 in Slika 20).

Ločevati moramo med dvema tipoma obiskovalcev. Obiskovalec brez meritve si bo ogledal kolekcijo na običajni način, brez posebnosti. Imel bo tudi možnost odpreti povezavo z meritvijo, ki mu jo bo pokazal prijatelj.

Obiskovalec z meritvijo ima možnost vnosa kode meritve. Po opisanem sistemu uporabnikov v poglavju 5.8.2 potrebujemo možnost vnosa kode meritve in možnost vnosa e-naslova za pridobivanje izgubljenih meritev. Ob vnosu kode meritve je meritev mogoče tudi prijaviti na e-naslov.

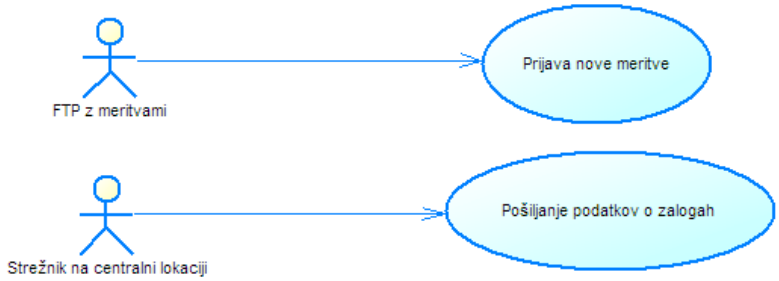
Po vneseni kodi meritve si bo obiskovalec lahko ogledal isto kolekcijo, vendar z dodatnimi podatki o najustreznejši velikosti zanj in o stanju zalog te številke za posamezen model.

Slika 19 Primeri uporabe spletne strani z vidika obiskovalcev

Podatke o ustreznosti (prileganju) je potrebno izvajati na strežniku, saj le tako zagotovimo aktualne podatke. Primerjalne datoteke se lahko menjajo, modeli se dodajajo in obsegi velikostnih številk se spreminjajo. Zato je nemogoče vnaprej izračunati prileganje in nanj pozabiti. Kljub temu prileganja ne moremo računati ob vsakem prikazu modela, zato izberemo vmesno pot. V podatkovno bazo shranimo izračune za vsako primerjalno datoteko in jih uporabljamo pri prikazu kolekcije. Omenjene spremembe v primerjalnih datotekah lahko obravnavamo na sledeče načine:

- v primeru dodajanja modelov z isto primerjalno datoteko, nam podatkov ni potrebno dodajat,
- v primeru dodajanja novih primerjalnih datotek, bo manjkajoči podatek očiten, zato takrat ponovimo primerjanje in posodobimo podatke,
- v primeru spreminjanja obstoječih primerjalnih datotek oziroma njihovih sortimentov pa je najpreprosteje izbrisati vse obstoječe rezultate te primerjalne datoteke. Na ta način dosežemo ponovni izračun po prejšnji točki.

Meritve na spletno stran dostavlja servis na FTP strežniku z meritvami, ki pošlje vse novo izmerjene meritve. Na spletnem strežniku se podatki vnesejo v podatkovno bazo za hiter prikaz. Eden izmed podatkov je tudi slika stopal, izrisana po točkah v meritveni datoteki. Ta se v prihodnosti ne bo spremenila, razen če bi spremenili podatke o meritvi. V tem primeru bi morali tudi na novo poslati podatke. Smotrno je torej izrisati sliko v naprej in jo hraniti v podatkovni bazi skupaj z ostalimi podatki. Prikaz slike stopal bo tako veliko hitrejši, kot če bi se risanje izvajalo ob vsakem prikazu.



Slika 20 Primeri uporabe spletne strani z vidika servisov mreže merilnikov

Poleg meritev se mora spletnemu strežniku dostavljati tudi podatke o zalogah po posameznih prodajnih mestih. Ti se nahajajo na centralni lokaciji v poslovnem sistemu proizvajalca. Dostopanje do tega sistema ob vsakem prikazu strani ni sprejemljivo, zato potrebujemo podatkovno bazo s kopijo podatkov in servis za njihovo posodabljanje.

V sistemu Alpine se zaloge posodabljajo enkrat na dan, zato smo tudi pošiljanje podatkov na spletni strežnik izvedli v enakem intervalu. Poslati je potrebno seznam vseh prodajnih mest, v katerih se izbrana linija izdelkov prodaja, nato pa še seznam količin modelov in velikostnih številk za posamezno prodajno mesto.

Vpeljava podatkov o zalogah ima tudi ta stranski učinek, da potrebujemo seznam vseh prodajnih mest. To izkoristimo kot dodatno podstran, kjer si lahko obiskovalec ogleda celotno prodajno mrežo proizvajalca. Če je ta obsežna, jo še nadrobneje razdelimo na države in regije.

5.8.4. Načrtovanje strukture spletne strani

Identificirane primere uporabe moramo zaobjeti v strukturi modelov, pogledov in kontrolerjev (MVC) in jih povezati z url naslovi, ki bodo predstavljali posamezne podstrani. Uporabimo lahko osnovno logiko izbire kontrolerjev in akcij ogrodja Zend Framework, ki mu dodamo še začetni člen vseh url naslovov, ki predstavlja jezik strani:

```
http://domena.com/jezik/kontroler/akcija/[param1/vrednost1/[.../]]
```

Če url naslov nima začetnega člena jezika ga poskušamo najti v piškotu obiskovalca (cookie), ki ga bo imel v primeru, da je stran že kdaj obiskal. V nasprotnem primeru ga določimo iz želje poslane ob zahtevku strani (`Accept-Language`) [64].

Naslednji člen url naslova predstavlja izbiro kontrolerja, sledi pa izbira akcije v tem kontrolerju. Naslovu nato dodamo poljubno število parametrov in njihovih vrednosti. Oglejmo si kontrolerje in akcije, ki jih potrebujemo za implementacijo vseh identificiranih primerov uporabe.

5.8.4.1. Kontroler za ogled meritev (`MeasurementController`)

Vse operacije povezane z meritvami združimo v kontrolerju za meritve. Uporabnik z vnosom kode meritve v obrazec sproži akcijo registracije, ki odpre obrazec z možnostjo vnosa e-naslova. Ta obrazec se lahko preskoči v primeru, da je e-naslov za to meritev že vnesen. Url naslov potreben za to akcijo je lahko preprosto `/measurement/register/`. Ob uspešnem odpiranju meritve se ta zapiše v sejo med uporabnikom in strežnikom za uporabo na vseh ostalih delih strani. V tej akciji izdelamo tudi zaščito za ugibanje kod skupaj z dnevnikom odpiranja meritev.

Sledi preusmeritev na akcijo ogleda meritve. Url naslov tukaj mora biti tak, da ga lahko uporabnik skopira drugi osebi, zato uporabimo hash kodo shranjeno z meritvijo. Npr: `/measurement/view/id/:hash_koda/`. Pri ogledu meritve preko takšnega naslova lahko prilagodimo pogled za obiskovalce, ki nimajo v seji shranjene te meritve. Zanje odstranimo kodo meritve in obrazec za vnos e-naslova. Lahko odstranimo tudi starost, spol in datum merjenja. Za dopolnitev te akcije potrebujemo še dodatno akcijo, ki iz podatkovne baze prebere sliko meritve in jo ustrezno posreduje, torej brez html kode in z določenim ustreznim tipom vsebine v glavi odgovora (`Content-Type`) [64]. Uporabimo lahko url oblike `/measurement/image/id/:hash_koda.jpg`.

Za primer pozabljenih meritev izdelamo akcijo, ki ob vnosu e-naslova pošlje pošto s seznamom vseh na ta e-naslov prijavljenih meritev. Uporabimo url `/measurement/lost/`.

Potrebujemo še akcijo za dodajanje meritev iz centralne lokacije sistema. Ta akcija preko POST zahtevka prejme besedilno datoteko s podatki meritve, ki jo mora prebrati in ustrezne vse podatke vnesti v bazo. Uporabimo url oblike `/measurement/add/` in ga zaščitimo z vezavo na seznam dovoljenih naslovov IP. Ustvarila bo en zapis v tabeli meritev in dva v tabeli stopal, po enega za vsako stopalo. Pred tem mora izrisati vnaprej pripravljeno sliko stopal iz koordinat v datoteki in določiti unikatno hash kodo za prijavljeno meritev. Akcija se zaključi z izpisom dogovorjenega niza za pravilni zaključek ali kodo napake.

5.8.4.2. Kontroler za ogled kolekcije (`CollectionController`)

V kolekcijski kontroler lahko združimo vse akcije, ki omogočajo brskanje po kolekcija. V njegovi inicializaciji poskrbimo za izdelavo izbirnika, ki ga sestavimo iz seznama kolekcij in spolov izdelkov v podatkovni bazi. Za brskanje po tem izbirniku potrebujemo akcijo pregleda, ki zna prikazati seznam izdelkov v posamezni izbiri. Za delovanje dvonivojskega izbirnika zadošča url oblike `/collection/overview/[section/:param1/[type/:param2/]]`.

Ko obiskovalec izbere izdelek, se aktivira druga akcija kontrolerja za njegov ogled. Ta prikaže vse njegove podatke in lastnosti. Za vsak izdelek imamo lahko več barvnih različic in več slik. Za pokritje teh lastnosti lahko uporabimo tehnologijo AJAX za asinhrono posodabljanje spletne strani [65], vendar vseeno v ozadju potrebujemo rezervni sistem za uporabnike brez AJAX podpore. Zahtevane operacije pokrije url `/collection/model/title/:param3/[color/:param4/[image/:param5/]]`.

Poleg tega mora akcija poskrbeti še za manjšo spremembo izbirnika, ki ima ob določenem izdelku še dve novi vrednosti: lastnosti in zaloga.

Če ima obiskovalec vneseno meritev, je potrebno pred prikazom izdelka preveriti, če obstaja predlog velikostne številke, da ga lahko izpišemo poleg lastnosti izdelka. V nasprotnem primeru je potrebno prej pognati proces primerjanja.

Zadnja akcija je prikaz zalog izbranega izdelka. Zanjo iz url naslova potrebujemo samo ime izdelka, zato zadošča oblika `/collection/stock/title/:param3/`. Če imamo obiskovalčevo meritev, lahko prikažemo le prodajalne, ki imajo izbran model na zalogi v ustrezni velikostni številki. Drugače prikažemo tiste, ki imajo na zalogi npr. več kot 5 kosov izbranega izdelka.

5.8.4.3. Kontroler za ogled prodajnih mest (`StoresController`)

Del strani s seznamom prodajnih mest je preprost dvonivojski izbirnik, ki prikaže vse prodajalne, ki ustrezajo trenutni izbrani lokaciji. V seznamu poskrbimo za posebno označitev prodajaln, v katerih se nahaja merilnik. Zanje lahko odpremo posebno mesto v izbirniku, čeprav ne gre dejansko za lokacijo. Url, ki zadošča vsem potrebam te akcije je oblike `/stores/list/[country/:param1/[region/:param2/[#store36]]]`. Zadnji člen je povezava do točno določenega prodajnega mesta na podstrani s pomočjo sidro povezave (anchor). Takšne povezave uporabljamo na strani prikaza zalog, kjer s klikom na prodajno mesto pridemo do podrobnejših podatkov o prodajalni. Dodamo še skripto na strani odjemalca, ki to mesto bolje označi z animacijo.

5.8.5. Primer podatkovnega modela

Primer konceptualnega podatkovnega modela (Slika 21) za zajete naloge se prične z združevanjem modelov v kolekcije. Te kolekcije lahko uporabimo tudi za začetni izbirnik na podstrani za ogled modelov. Vsak model ima ustrezne podatke za prikaz, med drugim tudi seznam lastnosti modela (parametrov). Posebnost delovanja sistema Alpine je ločevanje širinske različice na dve različni kodi, ki sta v centralnem sistemu vodeni popolnoma neodvisno, dodaja pa se jima dvomestna koda barvne različice.

Sledi entiteta barvnih različic vsakega modela, ki so z Alpinskim sistemom povezani z dvomestno barvno oznako (`modelTitle`). Ostali podatki so potrebni za delovanje izbirnika barvnih različic na strani določenega modela, ki vsebuje ime barve, njeno url pot in pot do datoteke s slikovno predstavitvijo barve. Za vsako barvno različico, velikost in širino potrebujemo podatke o količini izdelka na zalogi, in sicer za vsako prodajalno posebej.

Prodajalne vsebujejo splošne podatke za prikaz na spletni strani in so združene v pokrajine, te pa v države s pomočjo entitete lokacij. Ta entiteta je izvedena v obliki dvo ali več nivojskega urejenega seznama.

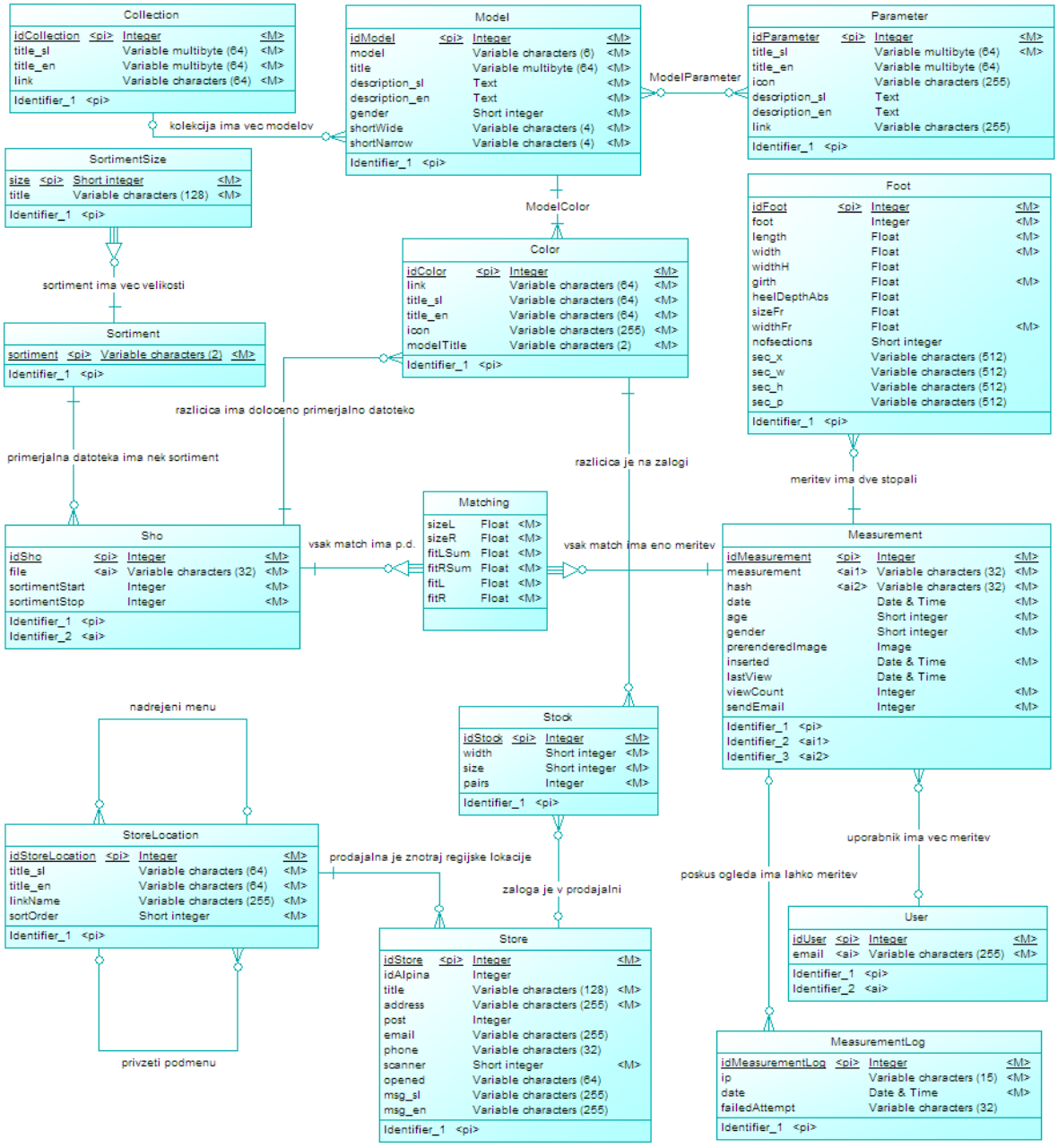
Vsaka barvna različica je povezana tudi s primerjalno datoteko (SHO), ki vsebuje podatke o obliki idealnega stopala za primerjanje z izmerjenim stopalom. Dejanski podatki se nahajajo v tekstovni datoteki, ki jo uporabi aplikacija za izračun prileganja. Primerjalna datoteka določa tudi sortiment izdelkov in v njem začetno in končno velikostno številko.

Primerjalne datoteke so povezane s posameznimi meritvami preko povezovalne tabele, ki vsebuje izračunane podatke o prileganju izmerjenih stopal s skupino izdelkov. Podatki obsegajo velikostno številko ter dve različni oceni ujemanja za levo in desno nogo. Entiteta meritev vsebuje vse podatke, ki jih vrne merilnik. Doda se še unikatno hash kodo za zaščito,

vnaprej izrisano sliko stopal ter statistične podatke o številu in datumu zadnjega ogleda. Vsaka meritev ima tudi dva zapisa stopal (levo in desno), kjer se nahajajo različne razdalje in mere pridobljene v procesu merjenja.

Za preprost uporabniški sistem se zbirajo samo e-naslovi in njihove poveze do posameznih meritev.

Pri odpiranju in prikazovanju meritev se piše dnevnik, ki vsebuje naslov IP in datum ogleda meritve, beležijo pa se tudi poskusi odpiranja neznanih meritev, ki jih lahko uporabimo za analizo, preprečevanje zlorab ali detekcijo napak.



Slika 21 Primer konceptualnega modela za podporo spletne strani za ogled meritev

6. Zaključek

6.1. Implementacija v praksi

V podjetju Alpina smo opisan načrt implementirali v praktični obliki in ne odstopa bistveno od opisanega načrta. Med izdelavo smo se sicer srečali z značilnim industrijskim pristopom, kjer sta čas in strošek izdelave najpomembnejša dejavnika. Zato so rešitve čim preprostejše in največkrat odstopajo od akademskih idealov. Z oranjem ledine se je implementacija veliko spreminjala in dograjevala, kar pusti sledi na sami strukturi projekta. V opisanem načrtu je zajeta že večina teh spoznanj in sprememb, ki so nastale v dosedanji življenjski dobi sistema.

Najbolj stabilna komponenta je bila uporabniški vmesnik merilnika, ki ni doživel skoraj nobenih popravkov. Več popravkov je bilo namenjenih programskemu vmesniku merilnika, predvsem zapisu kolekcijskih podatkov. Alpina ima zelo prilagojen in fleksibilen ERP sistem, ki skupaj z pogostimi spremembami kolekcije prinese veliko kombinacij izdelkov. Pojavila se je na primer zahteva, da so se barvna različice istega modela izdelovale v različnih sortimentih, z omejitvami v možnih širinah (večje številke se ni izdelovalo v ozkih različicah). Takšne zahteve se v poslovanju hitro pojavijo, pri izdelavi sistema pa le redko pomislimo nanje, ali pa jih zaradi rokov zavestno zanemarimo.

6.1.1. Nekaj zbranih rezultatov delovanja mreže merilnikov

Podjetje ima 7 aktivnih merilnikov stopal na prodajnih mestih po Sloveniji. Dodatno ima enega na domači lokaciji podjetja v Žireh, enega v prodajni mreži v Ameriki in enega na Švedskem. Poleg tega se uporablja še merilnik za obisk sejmov in večdnevnih akcij in pa prenosni merilnik za merjenje ob priložnostih (obisk vojske, diabetikov, sponzoriranih športnikov, itn).

Merilniki v prodajni mreži so stalno povezani s sistemom posodabljanja kolekcije in meritev in vsebujejo vse opisane komponente za analizo ter aktualno spletno stran.

V dvoletni razvojni dobi projekta se je zbralo 14.000 meritev, ki razkrijejo različne podrobnosti, kot so [5]:

- na merilnikih se pogosteje merijo ženske kot moški,
- največ meritev je opravljenih ob 11h in ob 17h,
- največ meritev je opravljenih v soboto in najmanj v nedeljo,
- daleč največ izmerjenih oseb je starih od 4 do 14 let.

Del spletne strani za ogled meritev ni bil tako uspešen, saj za uporabnika očitno ne ponuja velike dodane vrednosti. Smo pa uspeli na spletni strani vseeno narediti precej prometa z začetno nagradno uganko in nato z nagradnim natečajem za najboljšo poletno fotografijo. Dejanski namen celotne strani pa je v predstavitvi izdelkov iz kolekcije, kar pa opravlja več kot dobro.

6.1.2. Težave med delovanjem sistema

Od sedmih merilnikov so imeli trije kmalu po začetku obratovanja napako izpraznjene baterije na matični plošči, kar je pripeljalo do napačnih datumov v takrat izmerjenih meritvah. Za preprečitev težave smo v standardno inštalacijo vključili program Automachron, ki stalno uravnava čas računalnika (poglavje 5.1.2.2). Že prizadete meritve pa smo rešili z izdelavo

programa, ki je iz datuma prenosa meritve na FTP strežnik popravil ustrezne datoteke na eno uro natančno. Imeli smo tudi primer, ko datotek na ftpju ni več bilo, ker so bile umaknjene na arhivsko lokacijo. Takrat pa smo priredili program, da je podatke o času dobil iz dnevniške datoteke FTP strežnika.

6.2. Možnosti nadgradnje

Vsekakor je najbolj vroča naslednja faza projekta izdelava spletne trgovine s podporo opravljenih meritev. S tem korakom bi spodbudili merjenje in spletno prodajo in bi dejansko pomenil zaključeno celoto novega načina prodaje obutve.

Druga možnost je poudarek oglaševanja na prilagojenosti obutve stranki. K prilagojeni obliki stopalu bi lahko dodali še prilagojeno izbiro barv in materialov in tako dobili popolni paket kupcu prilagojenega izdelka. Informacijska podpora takšnemu pristopu bi bila spletna stran ali aplikacija na merilniku, kjer bi obiskovalec zbiral barve in materiale na posameznih delih čevlja. Primer takšnega pristopa, vendar brez podpore merilnikov, je NIKEiD [66].

Tretja možnost je posplošitev sistema, ki bi omogočal postavitev k poljubnemu proizvajalcu obutve. Na ta način se bi lahko sistem prodajal na trgu kot informacijska rešitev z večjo dodano vrednostjo, seveda na račun izgube trenutne unikatnosti ponudbe.

6.3. Sklep

Uspešna praktična implementacija v podjetju Alpina dokazuje uporabnost podanega načrta in primerov. Sistem deluje brez večjih izpadov na način, kot je bilo predvideno.

Vračilo investicije v razvoj se težko oceni, saj se njegove storitve ne zaračunava eksplicitno. Njegova vrednost je v zbiranju dragocenih podatkov in predvsem v možnosti uporabe v oglasnih akcijah za dviganje razpoznavnosti in unikatnosti podjetja, na sejmih obutve pa merilnik pripomore k posebnosti zakupljenega prostora.

Pravi učinek na poslovni rezultat bi lahko ocenili šele z izdelavo spletne trgovine in večjim številom merilnikov po prodajnih mestih, saj bi tako korenito dopolnili trenutno spletno prodajo obutve.

Obstaja pa tudi možnost ponudbe celotnega sistema na trgu drugim proizvajalcem obutve. Tako bi prodajali informacijsko rešitev, ki bi najbolje določila vrednost razvitega sistema. Poleg tega bi na ta način mogoče standardizirali nov način prodaje v veliko večjem obsegu.

7. Literatura in viri

[1] (2010) Shoe size - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Shoe_size

[2] ISO, "Shoe sizes - Mondopoint system of sizing and marking," Ženeva patent 9407, 1991.

[3] (2010) Metatarsus - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Metatarsus

[4] D. Stimpert. (2010) Children's Shoes - Tips & Advice for Choosing the Right Shoes for Children. Dostopno na: http://shoes.about.com/od/children_baby_shoes/a/childrens_shoes.htm

[5] M. Jezeršek, "Analiza meritev stopal," Alpina, d.o.o. interno gradivo, 2010.

[6] Alpina, d.o.o. (2008) Koncept večih volumnov - Binom. Dostopno na: http://www.binom-shoes.com/sl/binom/multivolume_concept/

[7] (2010) Canfit™ YETI™ 3D Scanner. Dostopno na: http://www.vorum.com/english/footware/measurement-carving-yeti-3d-scanner.php

[8] Vorum Research Corporation, "System Overview," in *YETI Foot Scanner User's Manual*. Canada, 2000, pogl. 2, str. 3-6.

[9] B. Novak, "Uporaba optičnih merilnikov v Alpini," in *Analiza modela kupcu prilagojene obutve*. Koper: Univerza na Primorskem, 2009, pogl. 2.4.4, str. 25-26.

[10] (2009) INFOOT USB (Standard Type) | Product information | I-Ware Lab. Dostopno na: http://www.iwl.jp/main/infoot_std.html

[11] (2009) FotoScan - powerful and affordable 3D scanners. Dostopno na: http://www.precision3d.co.uk/index.htm

[12] (2009) Dunk.net 3D Foot Scanner. Dostopno na: http://www.nectardesign.com/industrial-products/3d-scanner-industrial-design

[13] (2009) LaZerFit® Smart Soles. Dostopno na: http://www.lazerfit.com/details/index.shtml

[14] (2010) Canon Digital Imaging Developer Programme. Dostopno na: http://www.didp.canon-europa.com/

[15] M. Jezeršek, "Laserski sistem za tridimenzionalno merjenje hitro spreminjajoče se oblike teles," Univerza v Ljubljani, Fakulteta za strojništvo, doktorska disertacija, 2004.

[16] M. Jezeršek, "Three-dimensional scanning of feet," patent WO 2008/057056 A1, maj 15, 2008.

[17] (2008) Stardock ObjectDock. Dostopno na: http://www.stardock.com/products/objectdock/

[18] (2008) Touch-It Virtual Keyboard. Dostopno na:

http://www.chessware.ch/virtual-keyboard/

[19] (2008) Free Remote Access from LogMeIn. Dostopno na: https://secure.logmein.com/EU/products/free/

[20] (2010) Network Time Protocol. Dostopno na: http://en.wikipedia.org/wiki/Network_Time_Protocol

[21] (2010) Automachron 5.001. Dostopno na: http://www.softpedia.com/get/Desktop-Enhancements/Clocks-Time-Management/Automachron.shtml

[22] (2010) AVI Screen Saver. Dostopno na: http://www.pcworld.com/downloads/file/fid,7612-order,1/reviews.html

[23] P. K. Lawlis. (1997,) Guidelines for Choosing A Computer Language: Support For The Visionary Organization 2nd Edition. Dostopno na: http://archive.adaic.com/docs/reports/lawlis/content.htm

[24] (2010) Eclipse.org home. Dostopno na: http://www.eclipse.org/

[25] 37signals. (2005) Getting Real: Interface First. Dostopno na: http://gettingreal.37signals.com/ch09_Interface_First.php

[26] J. Miklavc. (2008) Oblikovanje grafičnega vmesnika merilnika stopal. Interno gradivo.

[27] (2010) Java Native Interface - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Java_Native_Interface

[28] (2009) RenderingHints (Java Platform SE 6). Dostopno na: http://java.sun.com/javase/6/docs/api/java/awt/RenderingHints.html

[29] (2010) Bilinear interpolation - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Bilinear_interpolation

[30] (2010) Bicubic interpolation - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Bicubic_interpolation

[31] (2010) Nearest-neighbor interpolation - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Nearest-neighbor_interpolation

[32] (2010) Merge sort - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Merge_sort

[33] (2010) Collections (Java Platform SE 6). Dostopno na: http://java.sun.com/javase/6/docs/api/java/util/Collections.html

[34] (2010) ArrayList (Java Platform SE 6). Dostopno na: http://java.sun.com/javase/6/docs/api/java/util/ArrayList.html

[35] (2010) Home of JavaDBF. Dostopno na: http://javadbf.sarovar.org/

[36] (2010) Zend Framework: Documentation: Adapters for Zend_Translate. Dostopno na: http://framework.zend.com/manual/en/zend.translate.adapter.html

[37] (2010) ResourceBundle (Java Platform SE 6). Dostopno na: http://java.sun.com/javase/6/docs/api/java/util/ResourceBundle.html

[38] (2010) PropertyResourceBundle (Java Platform SE 6). Dostopno na: http://java.sun.com/javase/6/docs/api/java/util/PropertyResourceBundle.html

[39] (2010) Properties (Java 2 Platform SE v1.4.2):. Dostopno na: http://java.sun.com/j2se/1.4.2/docs/api/java/util/Properties.html#encoding

[40] (2010) ISO/IEC 8859-1 - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/ISO/IEC_8859-1

[41] (2010) Native-to-ASCII Converter. Dostopno na: http://java.sun.com/javase/6/docs/technotes/tools/windows/native2ascii.html

[42] (2010) ListResourceBundle (Java Platform SE 6). Dostopno na: http://java.sun.com/javase/6/docs/api/java/util/ListResourceBundle.html

[43] (2010) MessageFormat (Java Platform SE 6). Dostopno na: http://java.sun.com/javase/6/docs/api/java/text/MessageFormat.html

[44] (2010) Codes for the Representation of Names of Languages. Dostopno na: http://www.loc.gov/standards/iso639-2/php/English_list.php

[45] (2010) ISO - Maintenance Agency for ISO 3166 country codes - English country names and code elements. Dostopno na: http://www.iso.org/iso/country_codes/iso_3166_code_lists/english_country_names_and_code_elements.htm

[46] (2010) Locale (Java Platform SE 6). Dostopno na: http://java.sun.com/javase/6/docs/api/java/util/Locale.html

[47] (2009) Intrinsic Locks and Synchronization. Dostopno na: http://java.sun.com/docs/books/tutorial/essential/concurrency/locksync.html

[48] (2009) Active FTP vs. Passive FTP, a Definitive Explanation. Dostopno na: http://slacksite.com/other/ftp.html

[49] (2009) Web service - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Web_service

[50] (2010) Using cron to run programs on a schedule. Dostopno na: http://www.scrounge.org/linux/cron.html

[51] IBM. (2009) JavaTM Secure Socket Extension (JSSE) IBMJSSE2 Provider Reference Guide. Dostopno na: http://www.ibm.com/developerworks/java/jdk/security/60/secguides/jsse2Docs/JSSE2RefGuide.html

[52] (2009) Informacijski pooblaščenec RS: Biometrija. Dostopno na: http://www.ip-rs.si/varstvo-osebnih-podatkov/informacijske-tehnologije-in-osebni-podatki/biometrija/

[53] (2009) Informacijski pooblaščenec RS: Zavarovanje zbirk osebnih podatkov. Dostopno na: http://www.ip-rs.si/varstvo-osebnih-podatkov/obveznosti-upravljavcev/zavarovanje-zbirk-

osebnih-podatkov/

[54] (2008) Commons Net - Jakarta Commons Net. Dostopno na: http://commons.apache.org/net/

[55] (2010) Web application framework - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Web_application_framework

[56] (2010) Comparison of web application frameworks - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

[57] (2008) Zend Framework. Dostopno na: http://framework.zend.com/

[58] (2010) Zend Framework Components. Dostopno na: http://framework.zend.com/about/components

[59] (2010) Model–view–controller - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Model_View_Controller

[60] (2010) Zend Framework: Documentation: The Standard Router. Dostopno na: http://framework.zend.com/manual/1.10/en/zend.controller.router.html

[61] (2010) Object-relational mapping - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Object-relational_mapping

[62] (2010) Internationalization and localization - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Internationalization_and_localization

[63] (2010) MD5 - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/MD5

[64] (1999) RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1 -- 14.4 Accept-Language. Dostopno na: http://tools.ietf.org/html/rfc2616#section-14.4

[65] (2010) Ajax (programming) - Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Ajax_%28programming%29

[66] (2010) NIKEiD. Dostopno na: http://nikeid.nike.com/nikeid/index.jsp

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

diplomskega dela

Spodaj podpisani/-a ANDRAŽ KOPAČ,

z vpisno številko 63040073,

sem avtor/-ica diplomskega dela z naslovom:

NAČRTOVANJE MREŽE MERILNIKOV STOPAL

IN POVEZANIH SISTEMOV V MALOPRODAJI OBUTVE

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek)

  prof. dr. Saša Divjak

  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 11. 3. 2010 Podpis avtorja/-ice: _____