UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andraž Selan

Sistem za nadzor orodij

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE

RAČUNALNIŠTVO IN INFORMATIKA

MENTOR : izr. prof. dr. Uroš Lotrič

Ljubljana, 2013

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00468/2013
Datum: 15.04.2013

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

Kandidat: **ANDRAŽ SELAN**

Naslov: **SISTEM ZA NADZOR ORODIJ**
PRODUCTION TOOLS MANAGEMENT SYSTEM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Računalniška evidenca orodij, ki se uporabljajo v proizvodnem procesu, lahko pomembno vpliva na učinkovitost proizvodnega procesa. Za izdelavo aplikacije za nadzor orodij izberite ustrezno tehnologijo, izdelajte aplikacijo in ocenite njene učinke na proizvodni proces.

Mentor:

izr. prof. dr.Uroš Lotrič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Izjavljam, da sem avtor diplomskega dela Sistem za nadzor orodij. Vpisan sem pod številko **63080148**.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Uroša Lotriča;
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) in ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela;
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki Dela FRI.

Podpis avtorja:

V Ljubljani, 10. februarja 2013

Zahvala

Za pomoč pri spoznavanju proizvodnih procesov in svetovanju se zahvaljujem zaposlenim v Hella Saturnus Slovenija.

Za strokovno pomoč in korekten odnos med nastajanjem diplomske naloge se zahvaljujem mentorju izr. prof. dr. Urošu Lotriču.

Zahvaljujem se tudi staršem, ki so mi omogočili študij, in fakulteti za kvaliteten študijski program.

Kazalo

Uporabljene kratice in pojmi

- SNO – Sistem za nadzor orodij
- SUPB – sistem za upravljanje s podatkovnimi bazami
- HSS – Hella Saturnus Slovenija
- SQL – Standard Query Language
- TCP – Transmission Control Protocol
- UDP – User Datagram Protocol
- SNMP – Simple Network Managament Protocol
- MDB – Management Database
- MIB – Managament Information Base
- Web platform – Skupek tehnologij HTML 5, CSS 3 in JavaScript
- HTML – Hypertext Markup Language
- CSS – Cascading Style Sheets
- IIS – Internet Information Services
- MVC – Model-View-Controller
- AD – Active Directoy

Povzetek

Zaradi vse večjega obsega proizvodnje žarometov se je v Helli Saturnus Slovenija pojavila potreba po evidenci orodij, ki jih uporabljajo pri izdelovanju polizdelkov za sestavo žarometov. S tem namenom je bila razvita spletna aplikacija Sistem za nadzor orodij, ki omogoča lažje upravljanje z orodji. Evidenca, ki se jo vodi v aplikaciji, omogoča tudi časovno primernejše čiščenje orodij, s čimer povečamo kvaliteto polizdelkov. Poleg tega je v aplikaciji tudi nekaj dodatnih lastnosti, ki so bile načrtovane že v začetku ali pa so se zanje ideje porodile pri razvoju. Z uvedbo aplikacije so zmanjšali stroške, povečali produktivnost zaposlenih in kvaliteto polizdelkov. V diplomski nalogi je opisan proizvodni proces, problematika in celoten proces razvoja programske opreme.

Ključne besede:

spletna aplikacija, žarometi, Hella Saturnus Slovenija, evidenca

Abstratct

Due to the ever increasing production of headlights, the company of Hella Saturnus Slovenia has shown the need for an inventory of tools which are being used in the production of semi-manufactured goods needed when producing the headlights. With this purpose in mind a web application which enables an easier management of such tools was developed. It is called “A System for the Control of Tools.” The inventory which is being controlled in this application also facilitates much better and more time-appropriate cleaning of such tools. In addition, it also enhances the quality of semi-manufactured goods. Besides, the application includes some additional features which were already initially planned or were later created as a result of the developmental process. All in all, the introduction of this application has lowered the costs, improved the productivity of employees and the quality of semi-manufactured goods. My diploma thesis thus outlines the production process of headlights along with the problems related to it and the entire process of the programming equipment and its development.

Key words:

web application, headlights, Hella Saturnus Slovenia, inventory, information, semi-manufactured goods;

Poglavje 1

Uvod

V proizvodnji Hella Saturnus Slovenija imajo zaradi povečanega obsega proizvodnje veliko orodij, ki jih uporabljajo za proizvodnjo žarometov. Ker ne vodijo evidence o trenutnih nahajališčih, v proizvodnji izgubijo veliko časa za iskanje teh orodij. Drugi problem nastane, ker ne vodijo evidence o uporabi teh orodij. Po določenem številu delovnih ciklov je potrebno orodja očistiti. Dogaja se, da jih prepozno ali prekmalu očistijo. Zaradi prepoznega čiščenja se na njih nabere odvečna snov, zaradi česar trpi kvaliteta polizdelka, ki se ga uporablja na tem orodju. Ker pa postopek čiščenja ni enostaven in hiter, je nesmiselno orodja prekmalu čistiti.

Pojavila se je ideja o izdelavi aplikacije, ki bi na enostaven in učinkovit način reševala problem evidence orodij. Aplikacija vsebuje vse potrebne podatke za lažje iskanje orodij in časovno primernejše čiščenje. Pri razvoju se je porodilo še nekaj dodatnih funkcionalnosti, ki omogočajo lažje spremljanje orodij. Z uvedbo aplikacije v proizvodni proces želimo izboljšati časovno iskanje orodij in optimalnejšo čiščenje, s čimer zmanjšamo število zaposlenih, zboljšamo produktivnost zaposlenih in kvaliteto polizdelkov, ki se jih uporablja pri orodjih za vakuumsko oslojevanje in brizganje plastike.

V prvem poglavju je opisan industrijski proces izdelovanja polizdelkov, ki so sestavni deli žarometov. Na kratko je opisano podjetje Hella Saturnus Slovenija, v katerem se srečujejo s to problematiko. Predstavljena je tudi analiza problema. Posebnost aplikacije je povezava z napravami, ki se uporabljajo za vakuumsko oslojevanje in brizganje plastike, zato je 3. poglavje namenjeno predstavitvi možnosti komunikacije s krmilniki, ki upravljajo te naprave. V naslednjem poglavju so opisani klasični procesi do uvedbe programske opreme v proizvodni proces. To je analiza problema, načrtovanje, implementacija, testiranje, prenos v produkcijsko okolje in vzdrževanje. Zadnje poglavje je namenjeno zaključku, v katerem so na kratko predstavljena dejstva, ugotovljena v času pisanja naloge.

Poglavje 2

Industrijski proces in analiza problema

Podjetje Hella Saturnus Slovenija je visokotehnološko podjetje z dolgoletno tradicijo, ki se ukvarja s proizvodnjo žarometov, meglenk in drugih svetil. Pri optimiziranju proizvodnih procesov želi biti pred konkurenčnimi podjetji. Zaradi vedno večjega povpraševanja in s tem širjenja obsega proizvodnje se sooča z mnogimi težavami pri proizvodnih procesih.

2.0 **Hella Saturnus Slovenija**

Podjetje Hella Saturnus Slovenija je del mednarodnega koncerna Hella in eden največjih slovenskih izvoznikov. Osnovna dejavnost je razvoj in proizvodnja svetlobne opreme za avtomobile, kot so žarometi, žarometi za meglo, žarometi za dnevno vožnjo ter eno- in več-funkcijske svetilke.

Leta 1921 so ustanovili podjetje Saturnus, ki se je prvotno ukvarjalo z izdelavo kovinske embalaže za prehrambno in kemijsko industrijo. S proizvodnjo svetlobne opreme za cestna vozila so začeli v letu 1948. Izdelujejo žaromete za avtomobilska podjetja Ferrari, Lamborghini, Mercedes, Audi, Nissan, Toyota, Volkswagen, BMW, Opel, Renault, Peugeot, Fiat, Lancia itd. V letu 1992 se je del proizvodnje svetlobne opreme odcepil in nastalo je podjetje Saturnus Avtooprema, d.d. Leta 2004 je celoten delež podjetja odkupila družba Hella KG Hueck & Co. Zaradi uglednega imena Saturnus se podjetje imenuje Hella Saturnus Slovenija.

Slika 1: Poslovni in proizvodni prostori

Družba Hella KG Hueck & Co danes deluje v več kot 20 državah po celem svetu, izdelke pa izvaža v več kot 100 držav. V celotni korporaciji je zaposlenih več kot 25.000 ljudi, ki kljub gospodarski krizi povečujejo dobiček. Raste tudi število zaposlenih.

2.1 **Industrijski proces**

Žaromet (Slika 2.2) je sestavljen iz več delov, glavni, osnovni del je ohišje, ki je sestavljeno iz plastike (slika 2.3). Ogrodje obdaja leča, ki je prav tako iz plastike. Na ohišje je pritrjena še elektronika in žarnice.

Slika 2: Primer žarometa Golf A7

Slika 3: Ohišje žarometa brez leče

Proces izdelovanja leče začnemo z brizganjem plastike, pri tem dobimo osnovno ogrodje. Uporabljamo večje in manjše naprave za brizganje plastike (slika 2.4). To so univerzalne naprave, ki jih uporabljamo v več industrijskih panogah za različne vrste plastičnih izdelkov.

Slika 4: Primer naprave za brizganje plastike

Za izdelavo potrebujemo še po meri izdelana orodja, ki jih nasadimo v napravo za brizganje. Ta orodja so iz hitroreznega jekla in imajo na trgu izjemno visoko ceno (slika 2.5), saj gre za posebno jeklo, ki ga enostavno obdelujemo in ki se tali pri višjih temperaturah.

Slika 5: Primer majhnega orodja za brizganje plastike

Ker Hella izdeluje žaromete in meglenke za množico modelov avtomobilov, uporabljajo v njeni proizvodnji veliko različnih orodij za izdelavo plastičnih polizdelkov. Na sliki 2.6 vidimo veliko skladišče orodij. Iz enega orodja prideta dva polizdelka, za levi in desni žaromet. Naprava deluje tako, da najprej segreje granulat (plastika), ki je v obliki majhnih kroglic, na 100-200 °C. Pridobi ga iz centralnega silosa in ga skozi odprtino nabrizga v orodje. Ko se plastika ohladi, nastane ulitek. Takrat se orodje razpre, robot ga z vakuumskim prijemalom prestavi na odlagalno površino za nadaljnjo obdelavo.

Slika 6: Skladišče orodij za brizganje plastike

Proces se nadaljuje z vakuumskim oslojevanjem plastike. Postopek se imenuje vakuumsko oslojevanje. Pri tem s posebnim kemijskim procesom na plastiko nanesejo tanko plast aluminija [1] z namenom, da se poveča svetilnost žarometa v temi zaradi odbijajočih se sten znotraj žarometa.

Oddelčni tehnolog preko sistema SAP dobi zahtevek za oslojevanje, v katerem je navedeno, katere vrste polizdelkov mora napariti in koliko kosov za vsako vrsto polizdelkov. Ti so določeni s projektom, tako da tehnolog dobi številko projekta in številko kosov za vsakega od projektov. V nadaljnjem procesu pošlje transporterja po planetnik, ki je primeren za določen projekt. Planetnik je valjasto orodje, na katerega natikajo polizdelke, in je po funkcionalnosti zelo podoben orodjem za brizganje, zato ga v našem sistemu prav tako imenujemo orodje. Za vsako vrsto polizdelka imajo na razpolago svoj planetnik (slika 2.7), ki je narejen po meri in je primeren za več različnih vrst polizdelka.

Slika 7: Primer dveh planetnikov za različne projekte (polizdelke)

Transporter v omari ali v skladišču poišče planetnike, ki so označeni z identifikacijsko številko, jih naloži na voziček in jih pripelje v skladišče naprave. Od tam jih delavci na napravi prestavijo na voziček naprave (slika 2.8).

Slika 8: Primer vozička naprave

Na voziček naprave gre osem planetnikov, ki jih delavci napolnijo s polizdelki. Ko jih napolnijo, jih eden zapelje v boben naprave (slika 2.9) in zažene proces vakuumskega oslojevanja.

Slika 9: Boben naprave

Vakuumsko oslojevanje je kemijski vakuumski postopek, ki temelji na kondenzaciji materiala (aluminija) na trdni podlagi (plastika). Z izparevanjem in sublimacijo material prenesejo na podlago, na kateri ga obstreljujejo z ioni, da postane trden. Postopek traja v povprečju 30 minut. V tem času pripravijo drugi voziček in ga napolnijo s polizdelki, da ga lahko hitro zamenjajo z vozičkom v napravi.

Voziček s planetniki, ki so jih neposredno pred tem naparili, potegnejo iz bobna naprave. Rezultat je polizdelek s tanko plastjo aluminija (slika 2.10).

Slika 10: Naparjeni polizdelek

Zaposleni prestavijo polizdelke iz planetnikov na vozičke in jih pošljejo v nadaljnjo obdelavo. Občasno pride do fizičnih okvar na planetnikih, ki jih odpravijo vzdrževalci. Po določen času uporabe se na njih nabere debela plast aluminija, ki slabša kvaliteto polizdelkov, zato morajo tehnologi po daljšem času uporabe poslati planetnike na čiščenje, pri katerem z luženjem odstranijo odvečni aluminij iz planetnikov. Polizdelki nato v škatlah potujejo do montažnih linij, na katerih jih zaposleni ročno ali s pomočjo robotov združijo s preostalimi deli v končni izdelek.

2.2 **Analiza problema**

Problem ki ga moramo odpraviti, je časovno neracionalno iskanje in luženje orodij. Ker v Helli potrebujejo za skoraj vsak polizdelek drugo orodje, se je do danes v proizvodnji namnožilo veliko orodij. Najprej smo raziskali problem pri planetnikih. Delovni cikel v povprečju traja 30 min. To pomeni, da imajo zaposleni toliko časa, da napolnijo drug voziček. Zaradi časovno potratnega iskanja, to lahko včasih traja dalj. Zato potrebujejo dodatnega

zaposlenega, ki jim pomaga pri hitrejšem iskanju planetnikov. Problematika je podobna tudi pri orodjih za brizganje plastike.

V Halli prav tako ne vodijo evidence uporabe orodij. Ne vodijo števila ciklov uporabe na napravi za vsako orodje in števila ciklov od zadnjega luženja, zaradi tega ne vedo, kdaj morajo planetnik poslati na luženje. Če ga pošljejo prekmalu, je to časovno potratno, saj lahko v takih primerih zaposleni porabijo preveč časa za čiščenje. Če pa ga pošljejo prepozno, se zniža kakovost polizdelkov, kajti na orodjih se nabere odvečna snov in zaradi tega so polizdelki slabše naparjeni.

S sistemom za nadzor orodij smo poizkušali odpraviti omenjene težave pri procesu.

Poglavje 3

Komunikacija z industrijskim krmilnikom

Zelo pomemben del pri Sistemu za nadzor orodij je tudi komunikacija z napravami oziroma komunikacija z industrijskimi krmilniki na napravah. Komunikacijo z napravami potrebujemo zaradi beleženja delovnih ciklov posameznih orodij v podatkovno bazo, slednjo pa potrebujemo zaradi funkcionalnosti čiščenja orodij. Ta se morajo po določenem času očistiti, saj se na njih naberejo odvečne snovi. Pri vakuumskem oslojevanju se nabere aluminij, pri brizganju pa odvečen granulat, zato trpi kvaliteta procesa vakuumsko oslojevanja in brizganja polizdelka. Poleg tega mora biti sistem zmožen nadgradnje za prikazovanje produktivnosti in evidence izmeta. Zaradi tega moramo beležiti delovni cikel orodij in uspešnost procesa. V poglavju so opisani trije najpogostejši načini komunikacije - protokoli, ki se uporabljajo za komuniciranje z industrijskimi napravami. Opisana je tudi rešitev, ki smo jo uporabili pri Sistemu za nadzor orodij, in vzrok, zaradi katerega smo se odločili za izbrani način.

Sistem mora avtomatsko osveževati podatke o aktivnosti naprav. To pomeni, da ko naprava preide iz aktivnega v neaktivno stanje ali obratno, nemudoma zapiše to spremembo v podatkovno bazo, kar pomeni, da v podatkovni bazi v tabeli lokacij, kjer se nahaja naprava v polje IsActive zapiše vrednost za aktivno stanje (ang. true), če je aktivna, ali vrednost za nekativno stanje (ang. false), če ni. Tako mora sistem preverjati vsako napravo posebej, torej sistem potrebuje časovnik, ki za vsako časovno enoto sproži preverjanje. Poleg tega mora naprava orodjem, ki so bila na aktivni napravi, prišteti deloven cikel. Ko naprava preide iz stanja aktivnosti v stanje neaktivnosti, prišteje vsem orodjem, ki so bila na aktivni napravi, en delovni cikel. To zapiše v tabeli, kjer so shranjena orodja, v polje CurrentCycleCount in CycleCount. V naslednji fazi razvoja sistema Sistema za nadzor orodij bomo to komunikacijo uporabili še za štetje dobrih in slabih kosov, ki pridejo iz naprave. S tem bomo lahko beležili evidenco izmeta in produktivnost zaposlenih. V naslednjih poglavjih so predstavljeni trije najpogosteje uporabljeni načini komunikacije, ki se večinoma uporabljajo med aplikacijo, ki teče na strežniku, in krmilniki na vseh napravah v proizvodnji.

3.1 **Serijska povezava**

Prva in najpogosteje uporabljena rešitev je komunikacija preko serijskega **vmesnika RS-232** s programskim protokolom UART, ki je najstarejši in najpogostejši protokol za prenos podatkov preko vmesnika RS-232. Komunikacija računalnik-krmilnik se najpogosteje uporablja za nalaganje programov na krmilnik, odkrivanje napak in nastavljanje parametrov programa. Lahko pa se uporablja tudi za pregledovanje stanja in upravljanje krmilnika za aplikacije, ki tečejo na računalniku. S **protokolom UART** beremo digitalne vhode in zapisujemo vrednosti v digitalne izhode, s čimer lahko kontroliramo motorje in aktuatorje, ki so priklopljeni na digitalne vhode in izhode krmilnikov [2]. V našem primeru mora aplikacija pri določeni časovni enoti preveriti stanje naprave s pomočjo branja digitalnih vhodov, na katere so priklopljena tipala. Če spremenimo stanje naprave, mora program odreagirati pravilno.

V našem primeru te komunikacije ne uporabimo, saj imamo centralni strežnik, na katerem teče program za preverjanje stanja vseh naprav, ne le ene. To pomeni, da bi morali razpotegniti serijske kable od centralnega strežnika do vseh naprav, kar pa je nesmiselno.

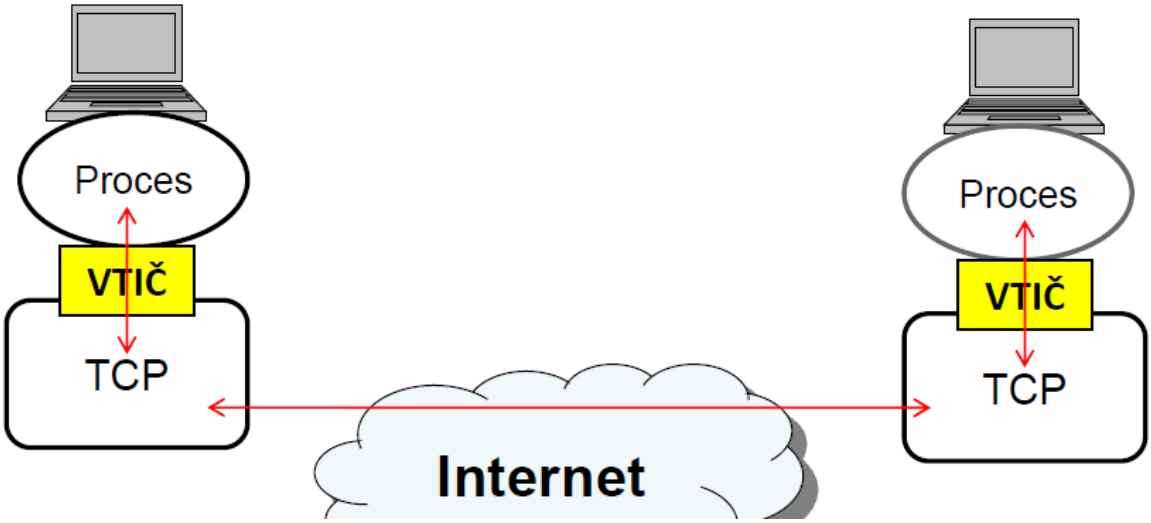
3.2 **Komunikacija profibus**

Profibus je uveljavljena komunikacija, ki se uporablja v več različnih oblikah. Prva oblika je Profibus-DP (Decentralized Periphery) in se uporablja za komunikacijo med merilnimi ter izvršilnimi sistemi s krmilniki. Druga oblika, ki je ena od možnosti za povezavo računalnika s krmilnikom, je Profibus-FMS (ang. Fieldbus Message Specification) in se uporablja za komunikacijo med krmilniki in komunikacijo med krmilniki in računalniki. Profibus-FMS uporablja **vmesnik RS-485**, ki omogoča dvosmerni prenos, vendar ne sočasno (ang Half Duplex) s hitrostjo do 12Mb/s. Za povezavo računalnika s krmilnikom potrebujemo na krmilniku vmesnik profibus, na računalniku pa IO-karto, preko katere aplikacija lahko bere in zapisuje vrednosti na vhodih in izhodih.

Ta vrsta komunikacije je primernejša od serijske povezave, kajti omogoča zaporedno vezavo kablov, kar pomeni veliko manj kablov kot pri serijski komunikaciji, zato si bomo pogledali, kako bi to lahko rešili s pomočjo TCP/IP-protokola, ki uporablja UTP-kable, ki so cenejši in so lahko dolgi do sto metrov.

3.3 TCP/IP

Druga rešitev je komunikacija na transportni plasti po protokolu TCP/IP. V našem primeru lahko uporabimo komunikacijo po protokolu TCP, saj omogoča zanesljiv prenos podatkov, brez napak in izgub paketkov. Lahko bi uporabili protokol UDP, ki pa se večinoma uporablja pri hitrem prenosu podatkov, pri katerem so napake in izgube paketkov dopustne. Primer uporabe UDP je prenos videa in slike, primer uporabe protokola TCP pa prenos spletnih strani, datotek in komunikacije med programi. Za komunikacijo med strežnikom in industrijskim krmilnikom, ki mora biti zanesljiva, se torej uporablja TCP-komunikacija. Proces (aplikacija), ki teče na nekem operacijskem sistemu, lahko komunicira z drugo aplikacijo preko vtiča TCP (ang. TCP socket). Ena TCP-povezava ima le en vtič na eni strani in drugi vtič na drugi strani. Primer povezave TCP prikazuje slika 5.1. Vtič je določen z naslovom IP in številko vhoda (ang. port). [3]



Slika 11: Komunikacija preko omrežja s TCP protokolom

Če želi proces na levi vzpostaviti povezavo s procesom na desni, mora odpreti vtič in poslati podatke na vtič, ki je na desni strani in je določen z naslovom IP in številko vrat. Drugi proces je lahko proces, ki teče na drugem znotraj ali zunaj omrežja ali pa na istem računalniku. Drugi proces mora imeti že prej odprt vtič, da lahko sprejme povezavo. Za vsak paket, ki ga vtič prejme, pošlje potrditev o prejetju paketa in potrditev o pravilnosti prejetega paketa. Med tem ko dva komunicirata, drugi proces ne more dostopati do iste povezave. To pomeni, da je lahko en vtič naenkrat odprt samo za eno povezavo TCP, in ne za več. Ko procesa končata s komuniciranjem, zapreta povezavo in ta se sprosti ter postane dostopna za drugi proces.

Pri industrijskih krmilnikih so se šele zadnja leta pojavili dodatni moduli, ki omogočajo komunikacijo po protokolu TCP/IP. Ti torej prenašajo protokol UART preko protokola TCP, namesto preko vmesnika RS232-protokol. Slika 5.2 prikazuje industrijski krmilnik Mitsubishi FX3U z modulom Ethernet. [4] Krmilniki Mitsubishi se uporabljajo tudi na napravah za vakuumsko oslojevanje v HSS.

Slika 12: Mitsubishi FX3U z Ethernet modulom

Na krmilniku imamo več digitalnih vhodov. Z določenim ukazom tako preko povezave TCP dobimo trenutno stanje vhodov. Kot odgovor dobimo zaporedje bitov. Primer: 0110 1001. Zaporedje bitov predstavlja vhode od x0 do x7. To pomeni, da so aktivni vhodi le x1, x2, x4 in x7. Seveda lahko preberemo tudi več vhodov ali pa celo spreminjamo izhode. Tako lahko na enega od vhodov vežemo signal iz naprave in ugotovimo, ali je naprava aktivna ali neaktivna.

S tako rešitvijo na preprost način vzpostavimo komunikacijo med računalnikom in industrijskim krmilnikom z modulom Ethernet. S programom se povežemo na krmilnik, s katerim lahko beremo vrednosti iz vhodov in zapisujemo vrednosti za izhode. Ta rešitev je torej primerna za preprosto komunikacijo. Moramo pa se zavedati, da je potrebno vedno znova preverjati, ali je prišlo do spremembe, kar prinese kar nekaj odvečnega prometa v omrežju. Za sam protokol TCP brez protokolov na višji ravni smo se odločili, ker gre za enostavno rešitev, pri kateri se ne pretaka veliko podatkov.

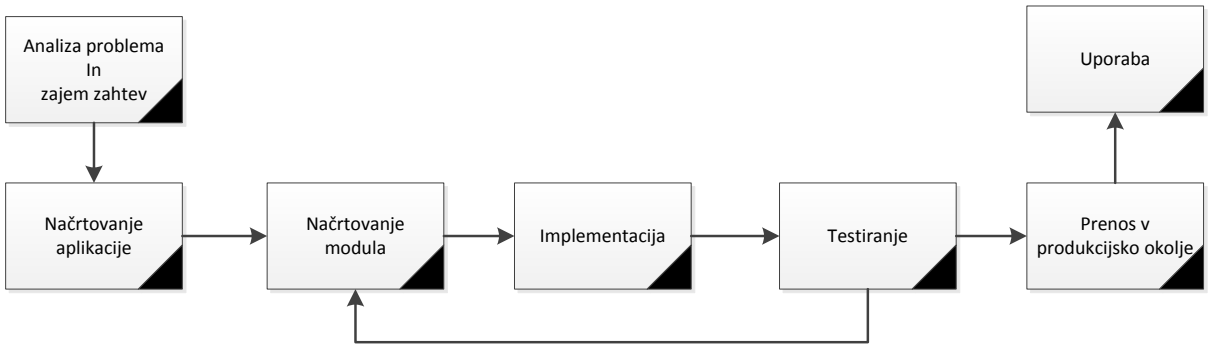
Poglavje 4

Sistem za nadzor orodij

V tem poglavju je predstavljen razvoj Sistema za nadzor orodij v procesu izdelovanja žarometov. Razložen je proces analize, načrtovanja modulov, podatkovne baze, implementacije, testiranja, produkcije in vzdrževanja. Natančno so razložene uporabljene tehnologije in vzrok uporabe. Opisan je tudi postopek testiranja aplikacije in prenos v produkcijsko okolje ter vzdrževanje. Na koncu so predstavljene možnosti izboljšav v aplikaciji.

4.1 Faze razvoja programske opreme pri SNO

Poznamo več različnih modelov razvoja programske opreme. Naš model razvoja aplikacije pri Sistemu za nadzor orodij je najbolj podoben postopnemu razvoj programske opreme. Pri postopnem modelu se funkcionalnost postopoma širi. V vsaki fazi smo se držali prototipnega razvoja, kar pomeni, da smo na začetku načrtovali in razvili prototip modula, ga testirali in nato ta postopek ponavljali, dokler nismo dobili stabilnega modula. Ker gre za relativno enostavno aplikacijo, je ta model primeren, saj je hiter. Pri kompleksnejših aplikacijah pa ni vedno primeren, saj šele na koncu ugotovimo pomanjkljivosti, vsaka sprememba pa je zelo kompleksna in draga. [5]



Slika 13: Diagram faz razvoja programske opreme pri SNO

V prvi fazi smo analizirali problem in zajeli zahteve, ki jih je naročnik podal v knjigi zahtev. Opravili smo tudi ekonomsko analizo in jo predstavili naročniku. V tej fazi smo dobili natančen odgovor, kaj potrebujemo in kaj moramo razviti. V naslednji fazi smo okvirno načrtali podatkovno bazo ter aplikacijo in jo razdelili po vsebinskih sklopih – modulih. V naslednji fazi smo začeli z načrtovanjem bistvenega modula. Nato smo ta modul implementirali, ga testirali in predstavili naročniku. Ob prvi iteraciji smo vedno našli še veliko programskih in vsebinskih napak. Poleg tega je skoraj vedno tudi naročnik našel pomanjkljivosti, tako da smo morali modul ponovno načrtati, ga implementirati in testirati, dokler nismo dobili stabilnega modula. Postopek smo ponovili za vse vsebinske module. Na koncu smo aplikacijo prenesli v produkcijsko okolje in jo začeli uporabljati.

4.2 Analiza problema in zajem zahtev

4.2.1 Analiza problema

Zaradi vse večjega števila orodij zaposleni vse več časa porabijo za njihovo iskanje. Zaradi tega se pogosto dogaja, da naprave niso izkoriščene, ker zaposleni iščejo prava orodja. S tem ustvarjajo veliko ekonomsko škodo, saj je pri tako velikem obsegu proizvodnje pomembna vsaka minuta. Z vodenjem ustrezne evidence bi se takim težavam izognili.

Prav tako v podjetju ne vodijo evidence uporabe orodij, števila ciklov uporabe na napravi za vsako orodje in števila ciklov od zadnjega luženja. Zaradi tega ne vedo, kdaj morajo orodje poslati na luženje. Če ga pošljejo prekmalu, je to časovno potratno, če ga pošljejo prepozno, trpi kvaliteta polizdelkov. Podroben opis industrijskega procesa in analiza problema se nahaja v 1. poglavju.

4.2.2 Zajem zahtev

Zahteve so bile podane v knjigi zahtev, v kateri so opisane vse zahteve o funkcionalnosti, ki jo mora aplikacija omogočati. Knjiga zahtev je nastala po Hellini standardni **Knjigi zahtev**, ki jo uporabljajo pri vseh novih projektih. To mora izvajalec projekta dosledno upoštevati .

Aplikacija mora operirati z bazo podatkov in omogočati izvoz in vpogled v podatke preko tematskih iskalnih oken. V tematska okna je možen direkten vpis ali vpis s pomočjo iskalnega okna. Tehnologi morajo imeti dostop in možnost urejanja, dodajanja in odvzemanja podatkov iz baze z uporabo več oddelčnih računalnikov naenkrat. Prikaz in delo z aplikacijo morata biti enostavna za rokovanje. Z uvedbo aplikacije nadzora orodij so želeli zagotoviti optimalnejše iskanje orodij, to pomeni časovno optimalnejšo porazdelitev planetnikov na voziček glede na čas nalaganja polizdelkov na planetnik, kemijsko čiščenje orodij glede na uporabo in nastavljen časovni normativ. Poleg tega so v Knjigi zahtev opisane tudi zahteve podatkovne baze in vrste podatkov, ki jih mora hraniti podatkovna baza. Njena struktura je opisana v naslednjih poglavjih.

V tehničnem opisu je zapisano, da mora biti aplikacija naložena na strežniku in mora delovati po principu **odjemalec-strežnik** (ang. client-server). Tehnična razpoložljivost aplikacije mora biti 99 %. Do aplikacije lahko dostopajo vsi z uporabo osebnega gesla. Dosegljiva mora biti na vseh oddelčnih računalnikih, torej na vseh računalnikih, ki so poleg naprav za brizganje ali oslojevanje plastike.

4.2.3 Ekonomska analiza

Z izračunom optimiziranja procesa in povečanja kvalitete smo vodilnim v podjetju predstavili stroške investicije in prihranke, ki nastanejo ob uporabi aplikacije. Izračun smo začeli pri

letnih stroških, ki nastanejo zaradi neoptimiziranega vakuumskega oslojevanja plastike. V naslednjih odstavkih je opisan proces izračuna, s katerim smo naročniku hitro prikazali, da lahko na enostaven način optimiziramo proizvodnjo.

Naprava ima za delovanje na voljo dva vozička. Zaposleni začnejo proces z vakuumskim oslojevanjem prvega vozička, drugega pa že pripravljajo za naslednji cikel. Glede na to, da traja delovni cikel v povprečju 30 minut, imajo prav toliko časa, da pripravijo drugi voziček. Najprej morajo poiskati planetnike, ki ustrezajo polizdelkom, ki jih morajo napariti. V povprečju potrebujejo za iskanje in dostavo voza k napravi 10 minut. Če bi zaposleni vedeli, kje se nahajajo planetniki, bi to lahko storili v bistveno krajšem času, okoli 3 minut. Zato potrebujejo na napravi več zaposlenih. Brez potratnega iskanja planetnikov bi prihranili 1,9 delavca na letni ravni na vseh petih napravah ob 4-izmenskem delavniku, kar je približno 35.150 € letnega prihranka.

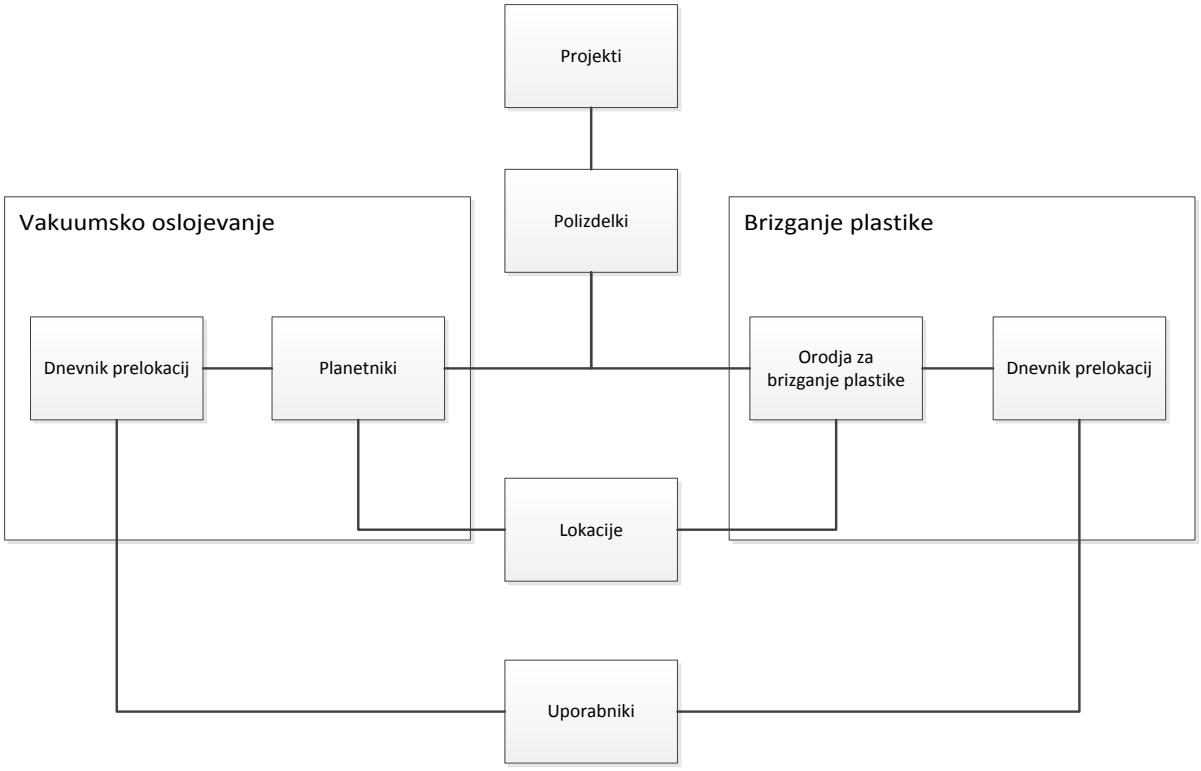
Začetna investicija v razvoj aplikacije je relativno majhna v primerjavi s povečanjem dobička in zmanjšanjem izmeta. Začetna investicija v razvoj znaša okoli 10.000 €, kar pomeni, da bo investicija povrnjena v približno štirih mesecih. Poleg tega so še mesečni stroški vzdrževanja aplikacije, ki znašajo 200 €, kar pa je zopet minimalno glede na večji dobiček, ki ga prinese podjetju aplikacija. Predvsem zaradi velikega obsega proizvodnje pomeni majhen odstotek povečanja velik dobiček.

Izpostaviti moramo še povečanje kvalitete polizdelkov, ki nastopi zaradi pravočasnega luženja orodja. Zaradi tega se zmanjša število reklamacij, s čimer raste ime podjetja. Tehnologije imajo zaradi boljše organiziranosti orodij boljši pregled in lahko na enostaven način upravljajo z orodji, s čimer se povečuje produktivnost zaposlenih.

4.3 **Načrtovanje aplikacije**

4.3.1 Vsebinska razdelitev

Načrtovanje aplikacije smo začeli pri vsebinski razdelitvi problema, ki smo ga razdelili na področja, ta pa predstavili v obliki diagrama.

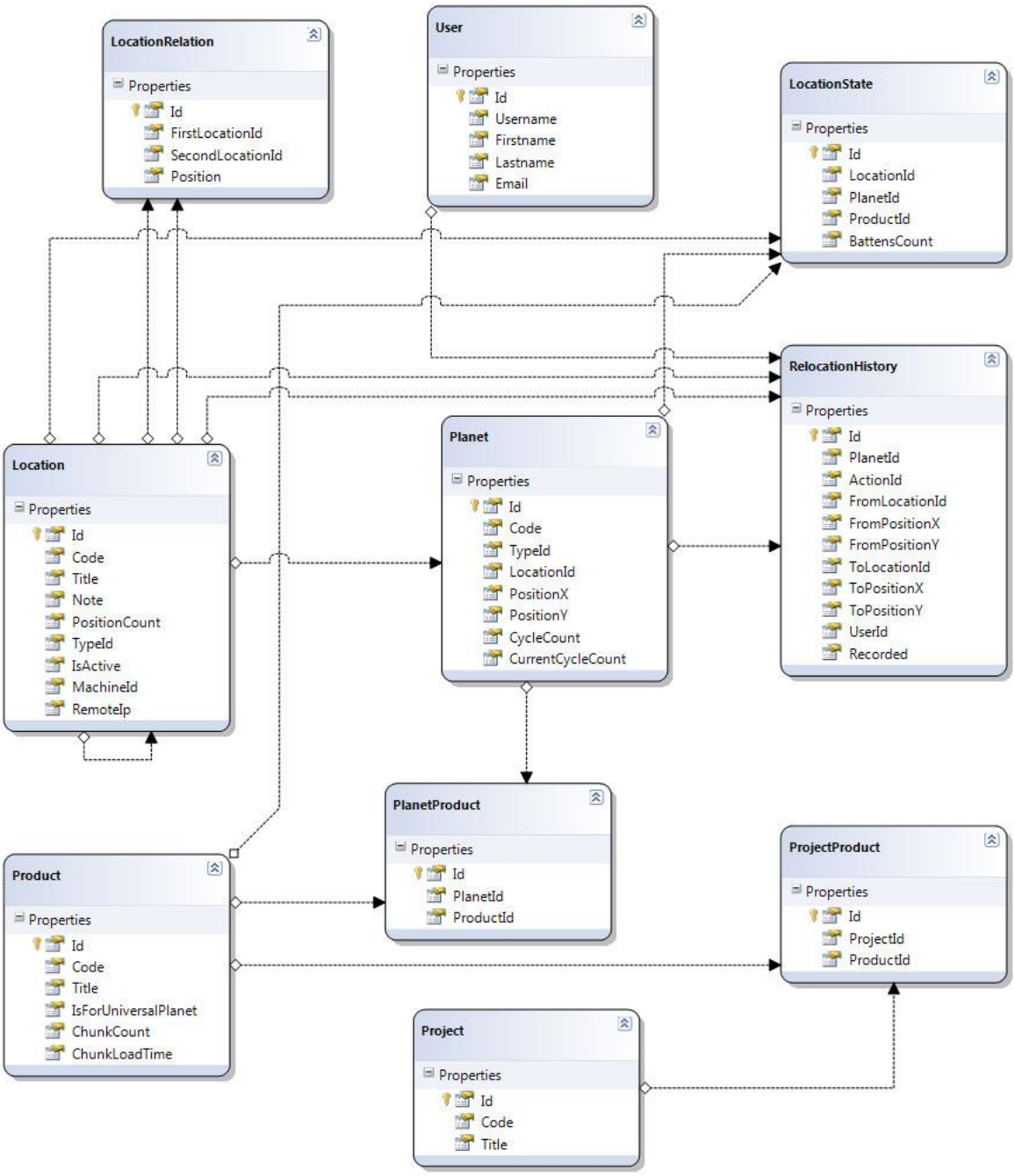


Slika 14: Diagram vsebinske razdelitve aplikacije

Diagram smo razdelili na dva dela. Prvi, levi, je podpora **vakuumskemu oslojevanju**, drugi, desni del, pa je podpora **brizganju plastike**. Pod vakuumsko oslojevanje spadajo planetniki, lokacije in dnevnik premestitev planetnikov. Pod lokacije spadajo naprave za vakuumsko oslojevanje in brizganje, omare, delavnice in čistilnice. Pri podpori brizganju plastike je situacija podobna, vendar se nekoliko razlikuje. Pri brizganju uporabljajo orodja za brizganje, in ne planetnikov, in drugačen dnevnik premestitev kot pri vakuumskem oslojevanju. Prav tako so vezani na enake polizdelke, ki jih imenujemo kar produkti. To pomeni, da se polizdelek najprej nabrizga v orodju za brizganje, nato pa se napari na planetniku, ki je vezan na isti polizdelek. Ti produkti so vezani na projekte, ki lahko vsebujejo več produktov, kar omogoča lažje iskanje planetnikov in orodij za brizganje. Dnevnika premestitev pri vakuumskem oslojevanju in premestitev pri brizganju plastike se vežeta na uporabnike, saj je uporabnik odgovoren za premestitev, ki se zabeleži v dnevnik. **V prvi fazi razvoja aplikacije smo razvili samo podporo vakuumskemu oslojevanju**, pri načrtovanju in implementaciji pa smo upoštevali možnost razširitve v prihodnosti tudi za podporo brizganju plastike.

4.3.2 Načrtovanje strukture podatkovne baze pri Sistemu za nadzor orodij

Pri načrtovanju strukture podatkovne baze smo dodobra spoznali proces brizganja plastike. V prvi fazi razvoja sistema smo zajeli samo evidenco planetnikov, ne pa tudi orodij za brizganje, tako da smo kot osrednji entitetni tip vzeli Planet (ang. Planetnik). Ustvarili smo še ostale entitetne tipe, ki so pomembni za delovni proces v sistemu, in jih prenesli v podatkovno bazo.



Slika 15: Struktura podatkovne baze pri SNO

Osrednja tabela, okoli katere se zbrane vse ostale tabele, je Planetnik. Namen tabele je hranjenje podatkov o vseh planetnikov in osnovne informacije o njih. Zabeležena je tudi trenutna lokacija, število delovnih ciklov, ki jih je opravil planetnik na napravi, in ali gre za univerzalni planetnik (ang. typeId). V strukturi vidimo tudi tabelo Planetnik-Polizdelek (ang. PlanetProduct), ki je vezna tabla med planetnikom (ang. Planet) in polizdelkom (ang. Product). Predstavlja seznam polizdelkov, ki so vezani na planetnike, torej je lahko polizdelek vezan na več planetnikov in obratno. Tabelo uporabljamo za lažje iskanje planetnikov glede na polizdelek. Tabela Planetnik-Polizdelek je vezana na tabelo Polizdelek, v kateri se hranijo vsi polizdelki. Poleg osnovnih podatkov o produktu nosi še informacije o največjem številu polizdelkov, ki jih lahko naložimo na planetnik, in čas nalaganja največjega števila polizdelkov. Tabela Polizdelek ima prav tako vezno tabelo s tabelo Projekt (ang. Project), ki predstavlja seznam projektov, ki so vezani na polizdelek, kar pomeni, da je lahko projekt vezan na več polizdelkov in obratno. Tabela se uporablja za lažje iskanje planetnikov glede na projekt preko polizdelka. V tabeli Projekt so prikazani samo osnovni podatki projektov. Vse premike lokacij planetnikov imenujemo premestitev in jih hranimo v Zgodovini premestitev (ang. RelocationHistory). Tabela služi hitremu odkrivanju napak pri premestitvah. Tu so shranjeni podatki o premikih. To so podatki o predhodni lokaciji ter poziciji in trenutni lokaciji in poziciji, odgovornem uporabniku ter času premestitve. Tabela je vezana na tabelo Lokacija (ang. Location), ki predstavlja seznam vseh lokacij, od omar in naprav do delavnice (za namen luženja). Poleg osnovnih informacij nosi še informacije o največjem številu mest, ki jih premore lokacija (na primer omara ima na voljo le določeno število mest). Namen je preprečiti dodajanje planetnikov v omaro, ki je že polnozasedena. Preko Tipa (ang. TypeId) ločimo omare od naprav in delavnice, za katere nosi tabela tudi informacijo o številki naprave in številke IP-krmilnika, ki je na napravi (RemoteIp). Tabela Lokacija se povezuje s tabelo Povezava med lokacijami (ang. LocationRelation), v kateri je zapisan vrstni red za vsako lokacijo glede na oddaljenost od ostalih lokacij. Vidimo tudi tabelo Stanje lokacije (ang. Location state), tu gre predvsem za stanje na aktivnih napravah in za podatke o tem, kateri planetniki so v napravi in koliko letev imajo, če gre za univerzalne planetnike. Na koncu ostane še tabela Uporabnik (ang. User), v kateri so zapisani osnovni podatki o uporabniku in podatki o avtentikaciji za programsko opremo.

Seznam vseh tabel in njihove lastnosti:

- **Planet** (planetnik):
    - Id: umetna identifikacijska številka;
    - Code: identifikacijska številka za fizično označevanje;

  - TypeId: tip;
  - LocationId: povezava, za trenutno lokacijo;
  - PositionX: številka mesta X koordinare (v omari ali v napravi);
  - PositionY: številka mesta Y koordinare (v omari ali v napravi);
  - CycleCount: število vseh ciklov, ki jih je planetnik opravil;
  - CurrentCycleCount: število ciklov od zadnjega luženja.

- **PlanetProduct** (vezna tabela med planetnikom polizdelkov):
  - Id: umetna identifikacijska številka;
  - PlanetId: povezava, planetnik;
  - ProductId: povezava, polizdelek.

- **Product** (polizdelek):
  - Id: umetna identifikacijska številka;
  - Code: povezava, identifikacijska številka prejeta iz SAP- sistema;
  - Title: naziv;
  - IsForUniversalPlanet: če je namenjena uporabi na univerzalnih planetnikih;
  - ChunkCount: največje število polizdelkov, ki jih lahko naložimo na en planetnik;
  - ChunkLoadTime: čas nalaganja največjega števila polizdelkov na en planetnik. .

- **ProjectProduct** (vezna tabela med projektom in polizdelkom):
  - Id: umetna identifikacijska številka;
  - ProjectId: povezava, projekt;
  - ProductId: povezava, polizdelek.

- **Project** (projekt):
    - Id: umetna identifikacijska številka;
    - Code: povezava, identifikacijska številka prejeta iz SAP- sistema;
    - Title: naziv.
- **RelocationHistory** (zgodovina premestitev):
    - Id: umetna identifikacijska številka;
    - PlanetId: povezava, planetnik;
    - ActionId: tip premestitve (na napravo, omaro, luženje);
    - FromLocationId: povezava, predhodne lokacije;
    - FromPositionX: številka pozicije koordiante X, iz katere se prestavi;
    - FromPositionY: številka pozicije koordiante Y, iz katere se prestavi;
    - ToLocationId: povezava, nova lokacija;
    - ToPositionX: številka pozicije koordiante X, na katerega se prestavi;
    - ToPositionY: številka pozicije koordiante Y, na katerega se prestavi.
- **Location** (lokacija):
    - Id: umetna identifikacijska številka;
    - Code: identifikacijska številka za fizično označevanje;
    - Title: naziv;
    - Note: Dodatno označevanje;
    - PositionCount: največje število mest, ki jih premore lokacija;
    - TypeId: tip lokacije (naprava, omara, delavnica, itd.);
    - IsActive: aktivnost naprave;

  - MachineId: dodatna umetna šifra, za označevanje naprav;
  - RemoteIp: IP številka krmilnika, ki je na napravi;

- **LocationRelation** (povezave med lokacijam):
  - Id: umetna identifikacijska številka;
  - FirstLocationId: povezava, prva lokacija;
  - SecondLocationId: povezava, druga lokacija;
  - Position: vrstni red oddaljenosti prve od druge lokacije.

- **LocationState** (stanje na lokaciji):
  - Id: umetna identifikacijska številka;
  - LocationId: povezava, lokacija;
  - PlanetId: povezava, planetnik;
  - ProductId: povezava, polizdelek, ki je na planetniku;
  - BattensCount: število letev, če gre za univerzalni planetnik.

- **User** (uporabnik):
  - Id: umetna identifikacijska številka;
  - Username: uporabniško ime, sinhronizirano iz ActiveDirectory podjetja;
  - FirstName: ime, sinhronizirano iz ActiveDirectory podjetja;
  - LastName: priimek, sinhronizirano iz ActiveDirectory podjetja;
  - Email: e-poštni naslov, sinhronizirano iz ActiveDirectory podjetja.

4.3.3 Vrste podatkovnih modelov

Poznamo tri najbolj razširjene vrste podatkovnih modelov:

- relacijski podatkovni model

- objektno usmerjeni model
- mrežni podatkovni model
- hierarhični model

Najenostavnejši za uporabo in najbolj razširjen je **relacijski podatkovni model**, ki temelji na relacijah. Relacijska podatkovna baza se uporabniku kaže kot množica tabel. Vsebino tabele dobimo s pomočjo poizvedbenih jezikov, ki so osnovani na relacijski algebri. Edini poizvedbeni jezik pri relacijskih podatkovnih bazah, ki je doživel standardizacijo, je SQL. Z njim lahko delamo poizvedbe, poleg tega pa lahko tudi gradimo strukturo podatkovne baze. To pomeni, da lahko z jezikom SQL ustvarjamo tabele. [6]

Mrežni podatkovni model temelji na podatkovnih zapisih, ki jih imenujemo mreža. Njen osnovni gradnik je zapis. V podatkovni bazi je množica zapisov, ki predstavljajo entitetne tipe. Zapis je sestavljen iz podatkovnih elementov, ki predstavljajo vrednost posamezne lastnosti entitete. [6]

Objektno usmerjeni podatkovni model temelji na objektih in metodah. Podatkovno bazo si lahko predstavljamo kot množico objektov, ki lahko gnezdijo med seboj. Objekt predstavlja entiteni tip, metoda pa povezavo med njimi. Izkaže se kot izredno hiter pri poizvedovanju, zlasti pri velikih podatkovnih bazah. [7]

Hierarhični podatkovni model je poseben model mrežnega modela. Uporabnikov pogled na hierarhično podatkovno bazo je pogled na gozd, ki ga sestavljajo drevesa. Podatkovno strukturo torej imenujemo gozd, njen osnovni gradnik je drevo, ki je sestavljeno izmed seboj povezanih zapisov. Zapis je torej podoben kot pri mrežnem modelu in predstavlja en entitetni tip. [7]

Pri Sistemu za nadzor orodij smo se odločili za uporabo najbolj razširjenega relacijskega podatkovnega modela, saj struktura ni zapletena in v njej ni veliko podatkov. Odločili smo se za Microsoft SQL Server 2012, ki je plod večletnega razvoja in testiranja. T-SQL, ki se uporablja za poizvedovanje, je dobro dokumentiran in uporabniku prijazen.

4.3.4 Uporabljene tehnologije

Za grajenje aplikacije smo se odločili za tehnologije Spletne platforme (ang. Web Platform) za interakcijo z uporabnikom, na strežniku se bo za procesiranje zahtev uporabljal ASP.NET MVC 4 v navezi z C# in Microsoft SQL Server 2012, za hranjenje podatkov. Prikaz

uporabljene tehnologije je na sliki 5.3. Za razvojno okolje smo uporabili Visual Studio 2010, ki omogoča napredne funkcionalnosti za razvoj programske opreme. Omogoča hitrejše razvijanje s pomočjo predlagane kode, avtomatsko testiranje modulov itd. Za upravljanje z verzijami in za deljenje projekta med uporabniki smo uporabili Microsoftov Team Foundation Server, ki ga podpira Visual Studio.

Slika 16: Diagram uporabljenih tehnologij

Spletna platforma je nov izraz, ki se uporablja za skupek tehnologij HTML 5 (ang. Hyper Text Markup Language), CSS 3 (ang. Cascading Style Sheets) in JavaScript za interakcijo z uporabnikom. HTML se uporablja za grajenje strukture in vsebine spletne aplikacije. Je vedno bolj uveljavljen standard za izmenjavo informacij preko svetovnega spleta. HTML je odprtokoden in ga lahko uporablja vsak za komercialne ali nekomercialne namene brez omejitve. Prav zaradi tega in neodvisnosti od operacijskega sistema se je močno razširil. Največja prednost je fleksibilnost, s tem mislimo predvsem delovanje na vseh možnih operacijskih sistemih, prilagajanje na različne velikosti ekranov; deluje na strojno zmogljivejših in manj zmogljivih sistemih, lahko tudi brez internetne povezave itd. Zadnja leta se pojavlja verzija 5, ki naj bi bila bolje standardizirana od prejšnjih verzij, zaradi tega je prihajalo do velikih razlik v brskalnikih. Za oblikovanje se uporablja CSS-verzija 3, ki je prav tako boljše standardizirana kot prejšnje. Za spreminjanje vsebine in strukture strani pa se uporablja JavaScript. Torej smo za interakcijo na strani odjemalca uporabili sodobne in fleksibilne tehnologije, ki tečejo v brskalniku. [8]

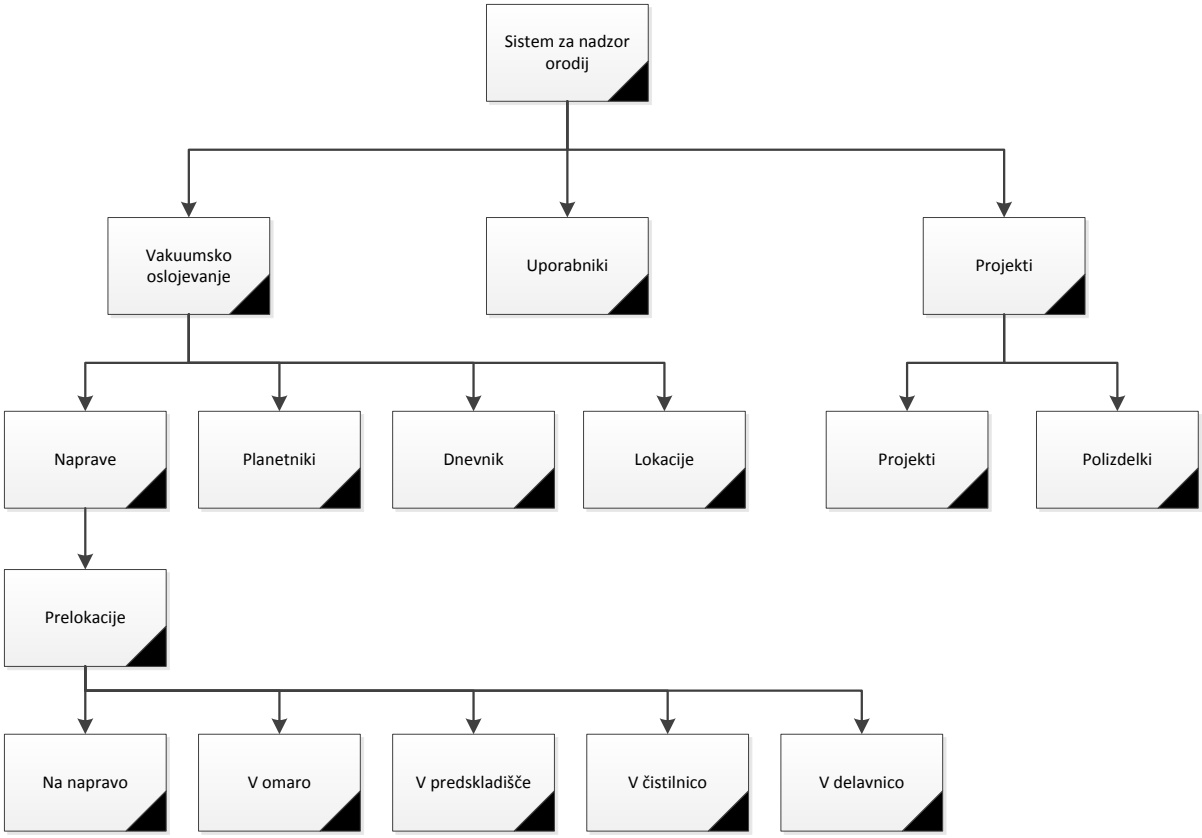
Za uporabo **Spletne platforme** potrebujemo spletni strežnik in tehnologijo za grajenje strukture HTML. Za to smo uporabili Microsoftovo ASP.NET MVC 4 tehnologijo, ki temelji na principu MVC (ang. Model-View-Controller). Je nov princip razvoja programske opreme, s katerim ločujemo kodo uporabniškega vmesnika (ang. view), zahteve uporabnika (ang. controller) in interakcijo s podatkovno bazo (ang. model) zaradi večje fleksibilnosti pri kombiniranju teh treh komponent in večje preglednosti kode. Kjer ne uporabljajo tega principa, na primer pri klasičnem ASP.NET (ang. Active Server Pages), se koda vmesnika, procesiranje zahtev in interakcija s podatkovno bazo pomešajo med seboj. Prednost principa MVC je tudi uporaba istih modelov (ang. model) pri različnih zahtevah uporabnika (ang.

controller) in pri različnih pogledih, le z drugimi parametri, s čimer preprečimo podvajanje kode, prihranimo čas, ki bi ga porabili za dodatno testiranje, in dosežemo lažje nadgrajevanje modulov. Torej za grajenje pogledov uporabimo HTML in CSS, za procesiranje zahtev uporabimo objektni programski jezik C# in za interakcijo s podatkovno bazo prav tako C# (ang. model). C# je Microsoftov napredni programski jezik, ki ima odlično podporo za delo s podatkovnimi bazami in za delovanje na različnih operacijskih sistemih se uporablja NET-programsko ogrodje. Lahko ga uporabimo tudi pri razvoju namiznih, mobilnih in spletnih aplikacijah. Torej na strani strežnika uporabljamo ASP.NET MVC 4 v navezi s C#, za kar pa potrebujemo spletni strežnik IIS (ang. Internet Infomation Services), ki lahko teče skoraj na vseh Microsoftovih operacijskih sistemih. [9]

Za podatkovno bazo smo si izbrali Microsoft SQL Server 2012. Do podatkovne baze dostopamo z LINQ knjižnico, ki je del NET- programskega ogrodja, s katerim poenostavljeno gradimo enostavne in zapletene SQL-zahteve, delamo poizvedbe, vstavljamo, posodabljamo in brišemo podatke. Microsoft SQL Server je napreden sistem za upravljanje s podatkovnimi bazami, predvsem v smislu hitrega in učinkovitega delovanja. S tem smo zaključili fazo načrtovanja aplikacije.

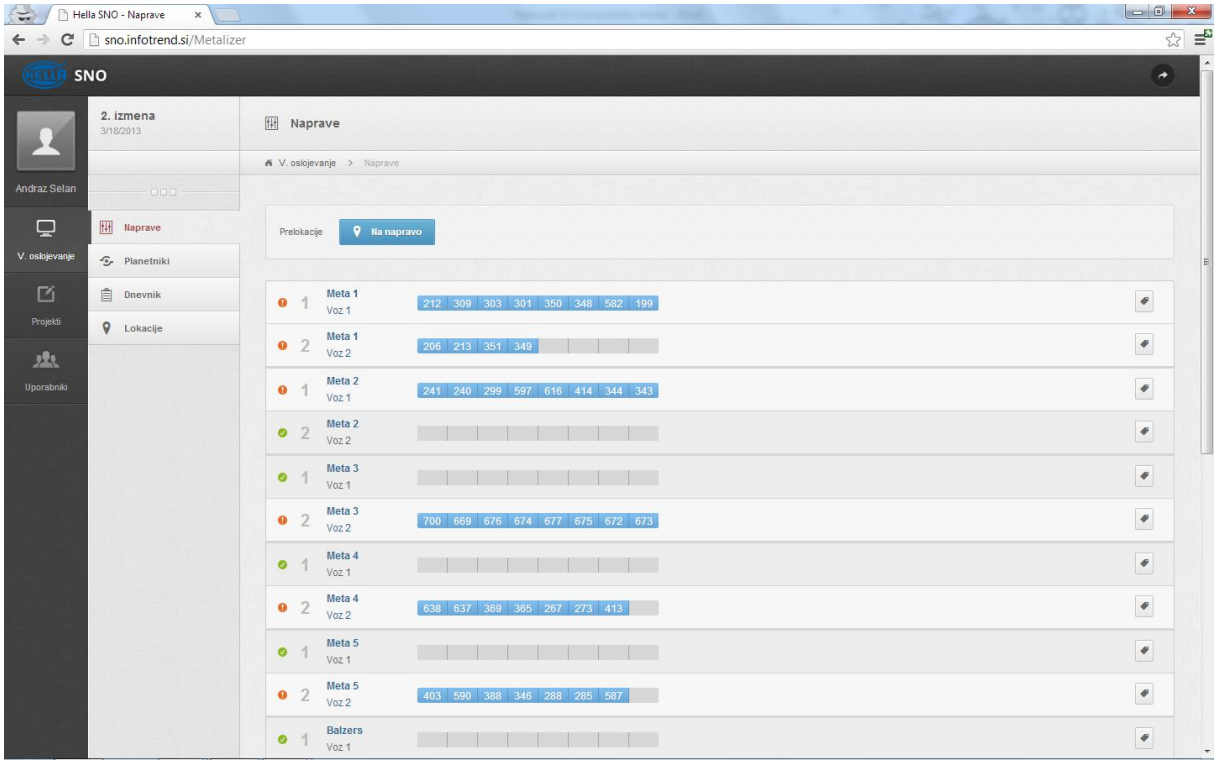
4.4 **Implementacija aplikacije**

Izdelavo spletne aplikacije smo začeli pri izgradnji strukture podatkovne baze na podatkovnem strežniku. Nato smo začeli graditi uporabniški vmesnik po principu MVC. Uporabniški vmesnik smo razdelili na več vsebinskih razdelkov (slika 5.4) **Vakuumsko oslojevanje**, **Projekti** in **Uporabniki**.



Slika 17: Struktura razdelkov uporabniškega vmesnika

Temu primerno smo zgradili glavni meni, ki vsebuje tudi podmeni. Pri grajenju vsebinskih razdelkov smo se držali vsebinske razdelitve, ki smo jo ustvarili v fazi načrtovanja. Pri gradnji HTML-pogledov smo izkoriščali tehnike prilagajanja ekranom (ang. responsive web page), ki jo omogoča 3. verzija CSS, ki spletno aplikacijo na enostaven način prilagodi različno velikim zaslonom. Pri manjših zaslonih skrije ali pomanjša manj pomembno vsebino, da ne zmanjka prostora za pomembnejšo vsebino. Na primer pomanjša ali skrije meni, glavo in nogo. To omogoča dobro izkušnjo na zaslonih osebnih in tabličnih računalnikov ter na mobilnih telefonih.



Slika 18: Osnovna postavitev spletne aplikacije

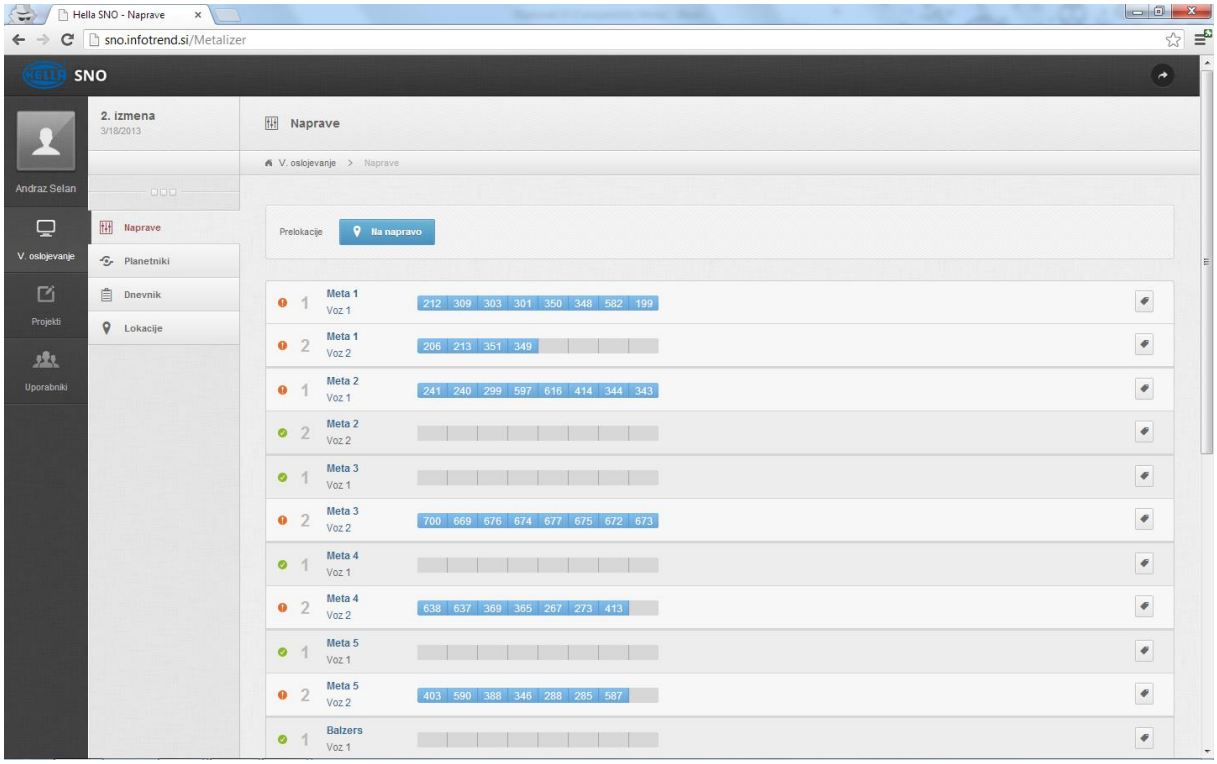
Slika 5.5 prikazuje osnovno postavitev spletne aplikacije. V levem zgornjem kotu se nahaja logotip in prikaz aktivnega uporabnika, v katerem se skrivajo povezave do upravljanja z računom in osebnimi podatki. Pod uporabnikom se nahaja meni, ki je sestavljen iz vakuumskega oslojevanja, projektov in uporabnikov. Poleg imamo podmenije vsakega aktivnega menija. Od podmenijev desno je vsebina trenutne podstrani. Na vrhu se nahaja naslov podstrani in pod njim navigacijska struktura vse do korena zaradi lažjega pomikanja nazaj, v desnem zgornjem kotu pa je tipka za odjavo iz aplikacije. Uporabo aplikacije je razložena v nadaljevanju. Ko se prijavimo v aplikacijo, se prikaže razdelek Vakuumsko oslojevanje.

4.4.1 Razdelek Vakuumsko oslojevanje

Razdelek Vakuumsko oslojevanje je bistvenega pomena v aplikaciji. Tu je združeno večina funkcionalnosti in logike. Uporabniki imajo možnost **premestitev in upravljanja z planetniki**. Sestavljen je iz razdelkov Naprave, Planetniki, Dnevnik in Lokacije.

V **razdelku Naprave** je pregled nad vsemi napravami za vakuumsko oslojevanje (slika 5.6). Vrstica v seznamu prikazuje voziček naprave. Za to sta za eno napravo dve vrstici, ker ima

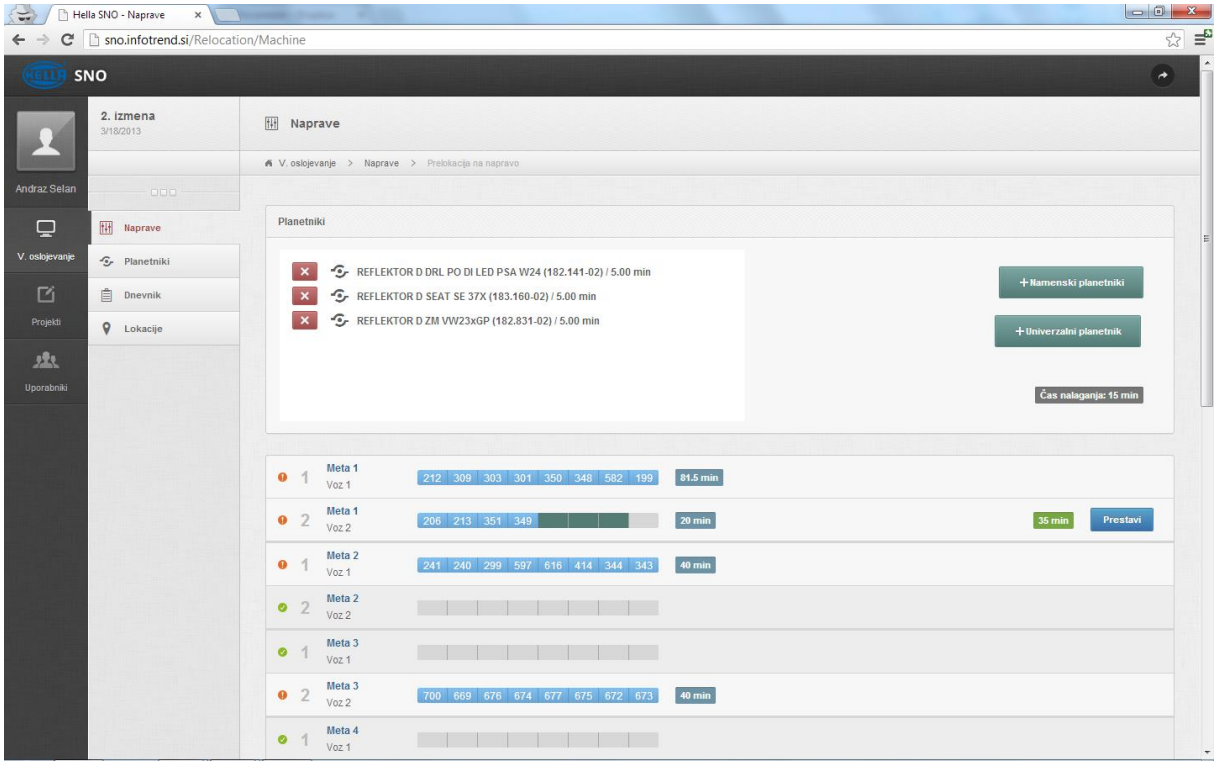
naprava dva vozička. Poleg številke vozička je podatek o njegovi aktivnosti, ob zeleni ikoni je voziček v napravi in naprava je aktivna, ob rdeči ikoni je zunaj naprave. Poleg tega je na voljo informacija o planetnikih, ki so naloženi na vozičku. Za premestitev planetnika na voziček je potrebno pritisniti tipko Na napravo v polju Premestitve, v katerem je tudi možnost premestitve iz naprav v omaro, skladišče naprave in v skladišče, kar pa je prikazano le ob zbranem vsaj enem planetniku, ki se nahaja na vozičku. Za premestitev planetnikov iz naprave je potrebno planetnik najprej izbrati. Izbere se ga s pritiskom na številko v modrem okvirju. Ob tem se na vrhu pokaže možnost premestitve v omaro, skladišče naprave ali glavno skladišče. S tem je omogočeno hitro menjavanje planetnikov iz vozička. Pri premestitvi planetnikov v omaro za uravnoteženo razporejanje po pozicijah v omari skrbi poseben algoritem, ki samodejno dodeli pozicijo v omari. S tem je doseženo fizično uravnoteženo razporejanje po omarah, s čimer je preprečena preobremenitev motorja, ki vrti verigo s planetniki.



Slika 19: Seznam vozičkov vsake naprave

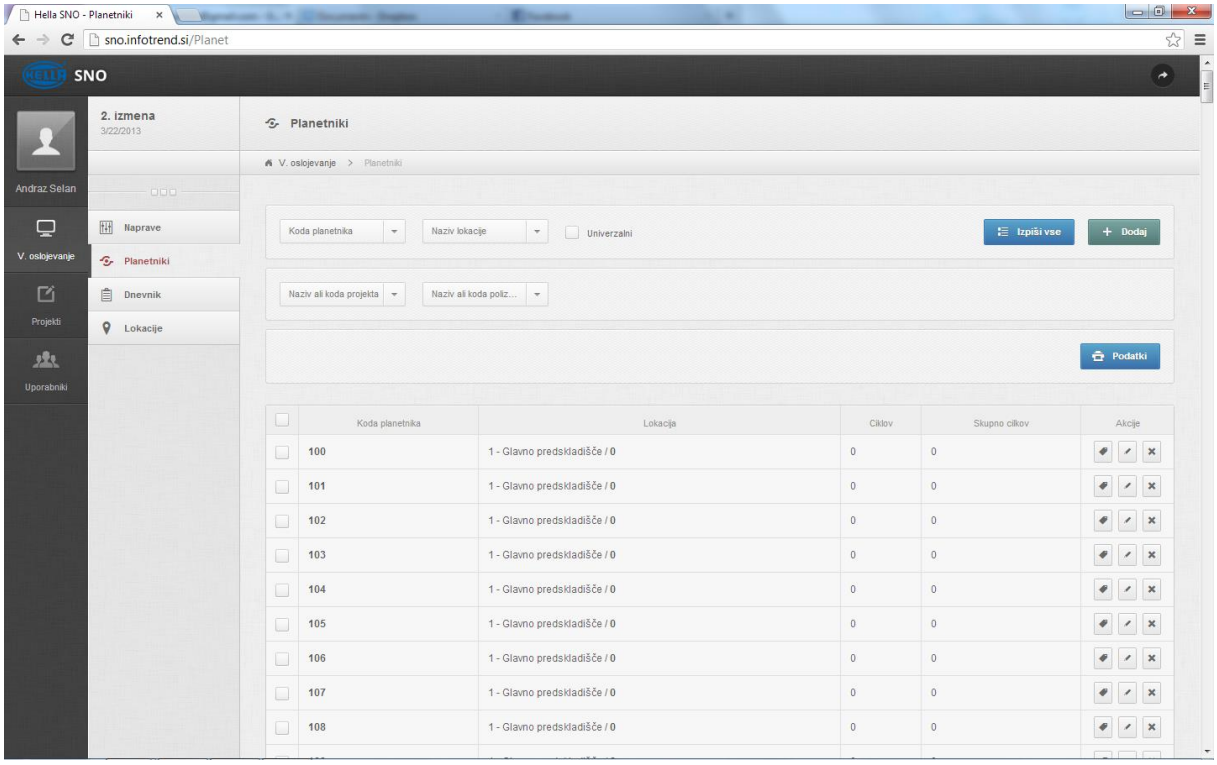
Pri premestitvi planetnikov na voziček se prikaže podstran, v kateri je potrebno najprej izbrati polizdelke, ki jih moramo napariti (slika 5.7). Tu se na enostaven način naredi premestitev na napravo, pri čemer je v pomoč prikaz zasedenosti vozičkov za vsako napravo posebej in je prikazan skupen čas nalaganja, ki omogoča lažjo odločitev uporabnika, na katero

napravo bodo odložili planetnike. Tako se zmanjšuje skupni povprečni čas nalaganja planetnikov na voziček. Po izbiri polizdelkov, ki jih je potrebno napariti, aplikacija samodejno predlaga planetnike. Vrstni red predlaganja je sledeč: planetnik iz skladišča naprave, iz bližnje omare, daljne omare, ostalih skladišč naprav in glavnega skladišča. Planetnikov, ki so na ostalih napravah, na luženju ali v delavnici, ne ponudi. Skupni čas nalaganja je prikazan z namenom, da ne presežemo priporočljivega časa nalaganja 30 minut. Na napravo imamo možnost premestiti tudi univerzalni planetnik. Za to imamo poseben gumb, ki prikaže okno za izbiro produktov in število letev.



Slika 20: Premestitve

Naslednji razdelek je **razdelek Planetniki**. Tukaj ima uporabnik možnost upravljanja s planetniki. Na voljo je seznam planetnikov, iz katerega lahko tudi doda nov planetnik, ureja obstoječe in jih briše. Slika 5.8 prikazuje seznam planetnikov.



Slika 21: Seznam planetnikov

V seznamu je prikazana številka, trenutna lokacija in pozicija, število vseh ciklov in število ciklov od zadnjega luženja. S pomočjo filtra lahko uporabnik hitro pride do informacij, kje se nahaja posamezen planetnik in kateri planetniki se nahajajo na posamezni lokaciji.

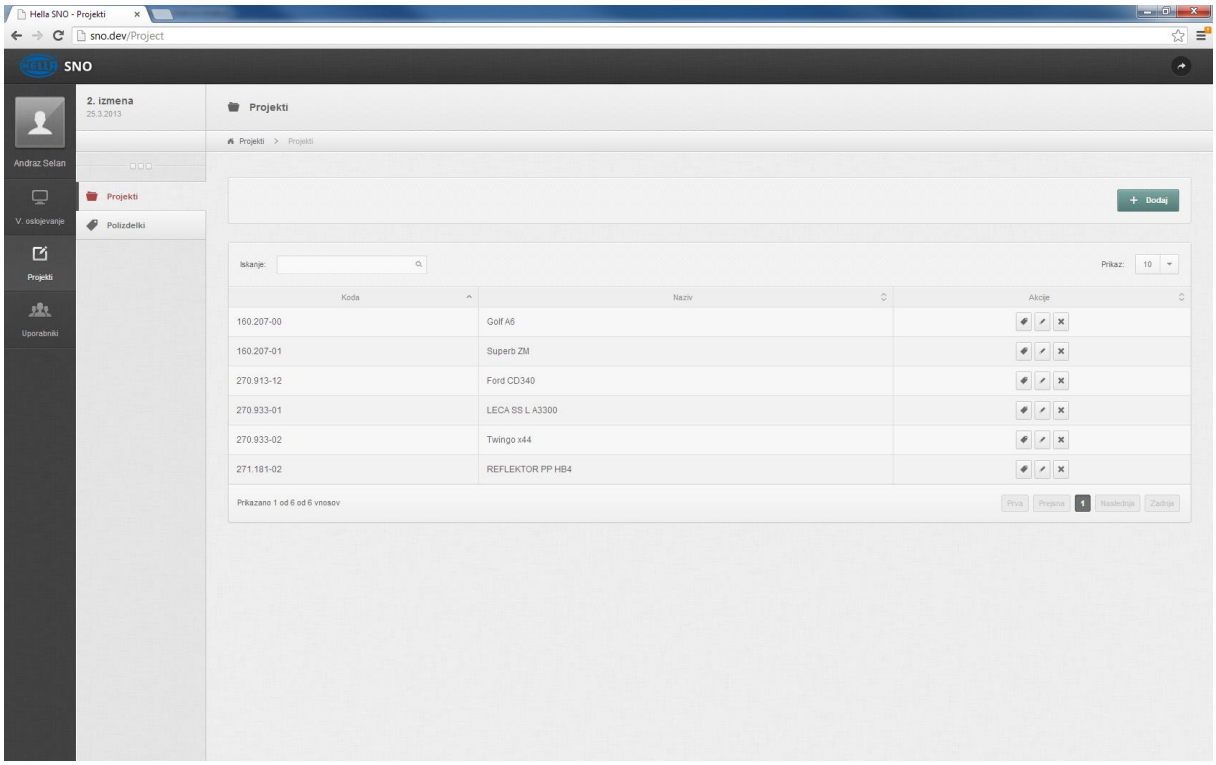
Nato sledi **razdelek Dnevnik**, v katerem je možnost pregledovanja seznama premestitev zaposlenih, kar omogoča lažjo detekcijo napak, ki nastanejo zaradi človeškega faktorja. V seznamu je podatek o prestavljenem planetniku, o vrsti premestitve, številki nove in predhodne lokacije. Pove nam tudi, kdo je opravil premestitev in kdaj se je izvršila.

V **razdelku Lokacije** je možnost upravljanja z lokacijami. Na voljo je možnost dodajanja nove lokacije ter urejanje in brisanje obstoječih. V seznamu so podatki o kodi, nazivu, opisu, številu mest, ki jih ima lokacija, in tipu naprave. Pri tipu naprave je možnost izbiranja med napravo, skladiščem, vozičkom in omaro. Evidenca teh lokacij se kot dinamičen šifrant uporablja v celotni aplikaciji za informacije o nahajališču planetnikov. To omogoča hitro dodajanje novih lokacij v primeru novih omar ali celo v primeru nove naprave.

4.4.2 Razdelek Projekti

V **razdelku Projekti** se upravlja s šifranti projektov in polizdelkov. Sestavljen je iz razdelkov Projekti in Polizdelki. Privzeto se prikaže razdelek Projekti

V **razdelku Projekti** je možnost upravljanja s šifrantom projektov, ki se uporablja predvsem za hitrejše iskanje planetnikov preko šifranta polizdelkov. Tu se lahko dodaja nove projekte ter ureja in briše obstoječe. Podatke o projektih se pridobi iz sistema SAP.



Slika 22: Seznam projektov

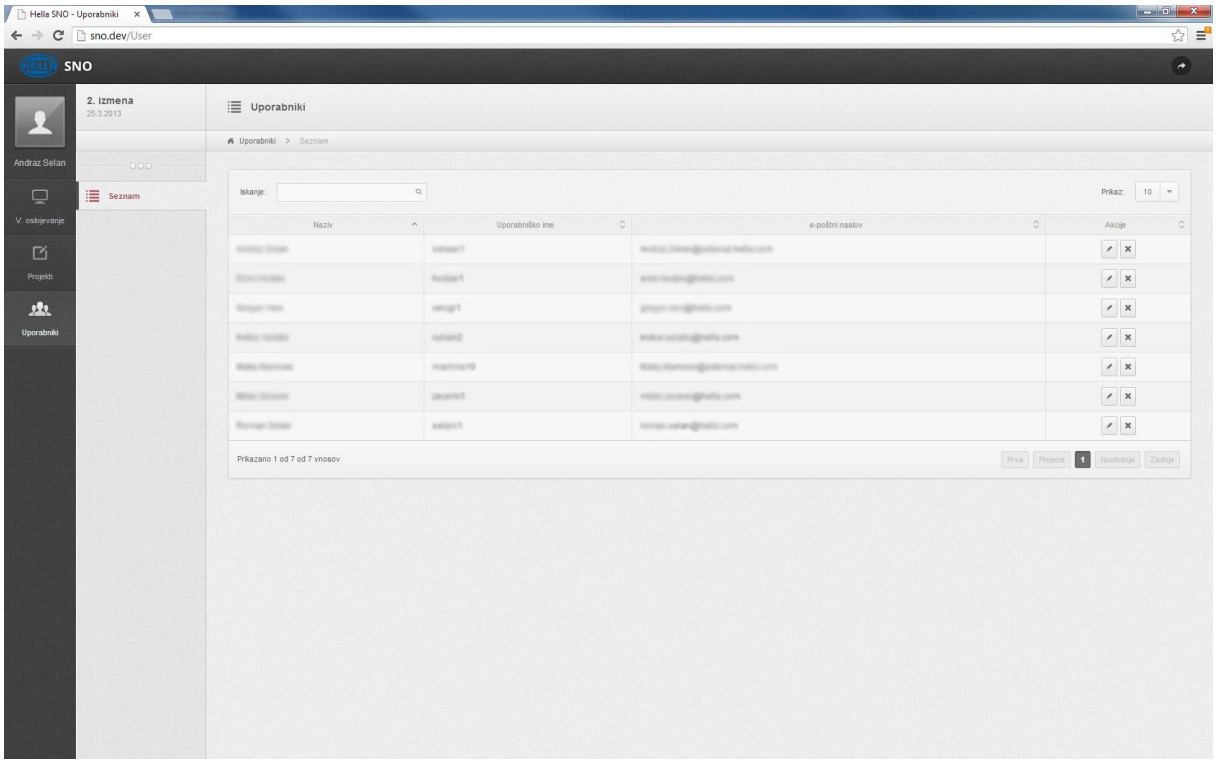
Slika 5.9 prikazuje seznam projektov, ki vsebuje tudi tipke za urejanje in brisanje posameznega projekta. Seznam prikazuje kodo in naziv projektov**.** V podstrani za njihovo urejanje je na voljo obrazec, v katerem se lahko ureja vse te podatke. V seznamu se nahaja še tipka za prikaz seznama polizdelkov, na katere je vezan določen polizdelek.

V **razdelku Polizdelki** je možnost upravljanja s šifrantom polizdelkov, ki se uporablja predvsem za hitrejše iskanje planetnikov, tudi preko šifranta projektov. Tu se lahko dodaja nove polizdelke ter ureja in briše obstoječe. Podatke o polizdelkih se prav tako dobi iz sistema SAP. Seznam polizdelkov vsebuje podatke o kodi in nazivu polizdelka, največje število kosov, ki se jih da naložiti na en planetnik in povprečen čas nalaganja največjega števila kosov na en planetnik. Preko podatka o povprečnem času nalaganja ima uporabnik informacijo o skupnem času nalaganja planetnikov na voziček. V seznamu se nahaja še tipka

za prikaz seznama projektov, na katere je vezan določen polizdelek. Na podstrani, na kateri se ureja polizdelke, je možnost urejanja vseh teh podatkov.

4.4.3 Razdelek Uporabniki

V **razdelku Uporabniki** se upravlja z uporabniki aplikacije. Tu je možnost urejanja in brisanja uporabnikov.



Slika 23: Seznam uporabnikov

Slika 5.10 prikazuje seznam uporabnikov, ki vsebuje podatke o imenu in priimku, uporabniškem geslu in e-poštnem naslovu. Na podstrani za urejanje uporabnika je možnost urejanja uporabniških pravic.

4.4.4 Avtentikacija in avtorizacija

Uporabnikom aplikacije so dodeljene različne pravice uporabe aplikacije. Možne vrste pravic aplikacije so:

- delavec

- tehnolog
- upravljalec
- superupravljavec

Uporabnik s pravicami »delavec« ima najmanj pravic pri uporabi, »superupravljavec« pa največ. Privzete pravice, ki jih uporabnik dobi pri registraciji uporabniškega računa so pravice »delavec«. Te vrste pravic so namenjene vsem zaposlenim, ki bi si radi ogledali seznam planetnikov, vendar vidijo samo razdelek Vakuumsko oslojevanje z razdelkom Planetniki in ničesar drugega. Pravice »tehnolog« imajo le oddelčni tehnologi, ki so odgovorni za naprevanje na posameznih napravah. Ti imajo v razdelku Vakuumsko oslojevanje tudi možnost premestitev planetnikov. To so tudi uporabniki, ki največ uporabljajo aplikacijo. Nato ima le nekaj uporabnikov pravice za »upravljalca«, ki je namenjen upravljanju planetnikov, projektov in polizdelkov. Pravice do »superupravljalca« imamo le razvijalci. Te omogočajo dodatne nastavitve aplikacije, urejanje uporabnikov in lokacij.

Aplikacija deluje le v omrežju Hella Saturnus Slovenija in do nje imajo dostop zaposleni, ki so vneseni v Aktivni imenik (ang. Active Directory) podjetja. To je podatkovna baza uporabniških računov za uporabo avtentikacije v različne sisteme, primarno pa za prijavo v Windows operacijske sisteme. Aplikacijo Sistem za nadzor orodij smo povezali z bazo Aktivni imenik, tako da ne potrebujemo dodatne baze uporabniških računov. Prednost je hranjenje uporabniških računov na enem mestu, kar zaposlenim v IT-oddelku prihrani čas pri kreiranju novega ali urejanju obstoječih uporabniških računov. Torej če podjetje dobi novega zaposlenega, zaposleni v IT-oddelku ustvari nov uporabniški račun z njegovimi podatki in ta lahko nemudoma uporablja Microsoftove operacijske sisteme, Sistem za nadzor orodij ter ostale aplikacije, ki so povezane z Aktivnim imenikom. Zaposleni v Sistemu za nadzor orodij za prijavo vnese le uporabniško ime in geslo. Dodatne pravice mu lahko dodeli uporabnik, ki ima pravice za »superupravljalca«.

4.5 **Testiranje aplikacije**

Testiranje aplikacije smo se lotili sistematično. Postopek testiranja smo razdelili na več delov.

Pri testiranju modulov smo testirali posamezne module (ang. Unit testing). Izvedli smo ga razvijalci. V veliko pomoč nam je bilo razvojno okolje Visual Studio 2010, ki omogoča avtomatizacijo testiranja, ?????????? ki omogoči natančnejše in hitrejše testiranje, s čimer lažje odkrijemo napake.

Pri integracijskem testiranju se testira povezave med posameznimi moduli in delovanje kot celota. Izvedejo ga razvijalci, ki so sodelovali pri nastajanju posameznih modulov.

Pri sistemskem testiranju aplikacijo testirajo razvijalci, sodelujeta pa še naročnik in uporabnik. Skupaj smo preverili delovanje posameznih modulov in povezave med njimi ter delovanje celote. Veliko napak smo odkrili tudi v tej fazi, kajti naročnik in uporabnik vidita delovanje aplikacije iz drugega zornega kota.

Prevzemno testiranje je podobno sistemskemu, le da tu aplikacijo testirajo uporabniki sami v okolju, ki je podoben produkcijskemu.

Regresijsko testiranje smo uporabljali vedno, ko smo spremenili ali dodali kakšen del kode. Tako smo preverili, ali so spremembe v kodi privedle do napak. Tako testiranje nam omogoča obvladovanje sprememb programske opreme. [10]

Pri razvoju programske opreme smo uporabili tri različna okolja. Prvo, v katerem smo razvijali posamezne module, je **razvojno okolje**. Pri nas smo za to uporabljali lokalne računalnike, na katerih so programerji razvijali programsko opremo. Tu smo izvajali testiranje modulov in integracijsko testiranje. Nato smo aplikacijo prenesli v **testno okolje,** v katerem so se izvajali integracijski in sistemski testi. Aplikacijo smo prenesli še v **produkcijsko okolje**, v katerem so uporabniki na testni podatkovni bazi izvajali prevzemno testiranje.

4.6 **Prenos v produkcijsko okolje**

Pri prenosu v produkcijsko okolje smo morali najprej evidentirati vse elemente, ki nastopajo v sistemu. Najprej smo se lotili označevanja in popisa planetnikov. Ko smo imeli njihovo evidenco, smo jih preko aplikacije vnesli v produkcijsko podatkovno bazo. Nato smo označili omare, odlagalna mesta v omarah in oddaljenost omar med seboj ter jih popisali. Prav tako smo jih preko aplikacije vnesli v produkcijsko podatkovno bazo. Sledil je popis naprav in vnos v podatkovno bazo. Podatke o projektih in polizdelkih smo dobili iz sistema SAP, podatki o uporabnikih pa so že v Aktivnem imeniku, tako da nam jih ni bilo potrebno posebej prenašati v našo podatkovno bazo. Uporabniki so najprej testirali uporabo aplikacije na napravi, ki za svoje delovanje potrebuje le planetnike, ki so specifični in se jih uporablja le na tej napravi. Tako nismo motili delovnega procesa na ostalih napravah. Ko smo aplikacijo pognali v produkcijskem okolju, smo naleteli na veliko težav, ki jih nismo predvideli ne v načrtovanju ne v funkcijski specifikaciji, zato smo morali aplikacijo popraviti in jo prilagoditi težavam. Šele ko smo bili prepričani o kvaliteti delovanja aplikacije, smo jo zagnali tudi na

ostalih napravah. Našli smo sicer napake, ki pa smo jih hitro odpravili, tako da nismo zmotili delovnega procesa. Ob koncu procesa prenosa smo predali še tehnično dokumentacijo.

Z ekonomskega vidika situacija vendarle ni tako enostavna, kot smo predvidevali. Pri razvoju so nastali dodatni stroški. Veliko časa so porabili tudi zaposleni, kar pomeni, da so trpele njihove glavne naloge. Za učinkovito in kvalitetno uporabo sistema so potrebni disciplinirani in zanesljivi zaposleni, kar je povezano z dolgotrajnim uvajanjem. V praksi se pokažejo prihranki, ki jih proizvodnja pridobi z uporabo aplikacije. Naši teoretični izračuni so pokazali, da aplikacija pripomore k 20 % boljšemu izkoristku pri proizvodnem procesu vakuumskega oslojevanja plastike, v praksi pa se kaže, da je ta številka le nekoliko manjša, okoli 18 %, kar je še vedno visok prihranek.

4.7 **Vzdrževanje**

Po končanem prenosu sledi faza vzdrževanja, v kateri je potrebno zagotoviti zanesljivo delovanje aplikacije, zato je potreben nadzor in opazovanje delovanja in zmogljivosti aplikacije. Predvsem v začetnem stanju vzdrževanja je potrebno zagotoviti odzivno in učinkovito tehnično podporo uporabnikom. Pomembno je, da je programska koda dobro strukturirana, berljiva in dobro dokumentirana. Ker se aplikacija veliko uporablja, je potrebno nenehno izboljševanje, reševanje in realizacija novih funkcionalnosti. Dokler aplikacija deluje oziroma dokler je ne nadomesti nova, je potrebno skrbeti za nemoteno delovanje in prilagajanje spremembam v proizvodnji. Vzdrževanje se pravzaprav nikoli ne konča.

Zaradi dobre zasnove in zgradbe aplikacije z vzdrževanjem ni veliko dela. Občasno mora skrbnik dodati kakšne nove vnose, kot so pravice novih uporabnikov in dodajanje ali urejanje lokacij. Prav tako ni veliko dela pri vzdrževanju podatkov v podatkovni bazi, saj za to skrbijo sistemski procesi aplikacije, ki samodejno posodabljajo podatke. Največ je dela s podporo uporabnikom, ki imajo težave z uporabo računalnikov ali pa niso prebrali navodil.

4.8 **Možnost izboljšav**

Možnosti za izboljšavo je vedno veliko, predvsem zaradi uporabnikov, ki najdejo veliko »pomanjkljivosti«, ki bi se jih dalo odpraviti, pa tudi veliko dodatnih funkcionalnosti, ki bi se jih dodalo v aplikacijo. Vseh funkcionalnosti se ne da dodati zaradi prezapletenega ali predragega razvoja ali pa funkcionalnost enostavno ni relevantna oziroma bi se jo redko uporabljalo. Izpostavili smo tri funkcionalnosti, ki bi bile vredne dodatnih stroškov razvoja in bi se dejansko uporabljale. Prva je povezava Sistema za nadzor orodij z sistemom SAP.

Druga je razvoj aplikacije za spremljanje produktivnosti zaposlenih, ki delajo na napravah, in učinkovitost naprav. Tretja funkcionalnost je razširitev aplikacije z možnostjo upravljanja z orodij za brizganje plastike.

V trenutni fazi administratorji ročno vnašajo podatke o projektih in njihovih polizdelkih, ki jih pridobijo iz sistema SAP. S pomočjo API-sistema SAP, bi bilo možno avtomatizirati prenos podatkov med tema sistemoma, s čimer bi zagotovili ažurnost podatkovne baze Sistema za nadzor orodij in bi tako prihranili delo tehnologom.

Vodje proizvodnje si želijo čim višjo produktivnost zaposlenih in čim manjši izmet polizdelkov, s čimer bi se zmanjšali stroški in bi se dvignil ugled podjetja, zato razmišljamo o razvoju dodatnih aplikacij, ki bi tekle na oddelčnih računalnikih in omogočale hiter vnos podatkov o slabih in dobrih kosih. S pomočjo teh podatkov bi lahko vodje optimizirali proizvodnje procese.

Proces brizganja plastike je zelo podoben procesu vakuumskega oslojevanja plastike. To pomeni, da bi lahko z majhnimi stroški dodatnega razvoja aplikacije Sistem za nadzor orodi*j* dodali še funkcionalnost za brizganje plastike. Tu imajo namreč prav takšne težave - iskanje in pravočasno čiščenje orodij.

Poglavje 5

Zaključek

Ko sem spoznaval industrijski proces v Helli, sem se seznanil s področjem industrijske avtomatike. Podjetje nameni veliko sredstev za razvoj industrijskih procesov, zato me je delo v tehnološko dovršeni proizvodnji navdušilo. Pri razvoju aplikacije mi je koristilo znanje, ki sem ga pridobil med študijem, in izkušnje, ki sem jih pridobil pri praktičnem izobraževanju. To znanje sem povezal z znanjem strojnih, elektrotehniških in kemijskih inženirjev. Dobil sem dodatne izkušnje in nova znanja pri razvoju programske opreme in z drugih področij.

V diplomski nalogi sem obravnaval področje optimizacije proizvodnih procesov. Cilj je bil razvoj aplikacije, s katero lahko na enostaven način upravljamo z orodji in olajšamo delo zaposlenim, s čimer podjetje prihrani veliko sredstev. Pri razvoju smo se s Hellinimi zaposlenimi srečali z mnogimi težavami, ki pa smo jih uspešno odpravili. Največ jih je bilo z zaposlenimi, ki morajo biti dosledni pri uporabi aplikacije, kar pa je težko doseči pri nizko izobraženem kadru v proizvodnji. Uspešno smo razvili in uvedli aplikacijo, s pomočjo katere smo optimizirali proizvodnjo.

Z uvedbo aplikacije je podjetje zmanjšalo čas iskanja orodij, zaradi česar v povprečju prihrani plačo za 1,9 zaposlenega, kar je na letni ravni dobrih 35.000 €. Zaradi večje organiziranosti orodij se je povečala produktivnost zaposlenih. Povečala se je tudi kvaliteta izdelkov, zaradi česar je manj reklamacij, s čimer raste ime podjetja. Začetna investicija se je podjetju torej hitro povrnila.

Kazalo slik

Literatura

[1] B. Janežič, Ščitenje sestavnih delov žarometov pri vakuumskem oslojevanju, Ljubljana: Univerza v Ljubljani, Fakulteta za strojništvo, 2011.

[2] „Universal asynchronous receiver/transmitter - Wikipedia, the free encyclopedia,“ Wikipedija, [Elektronski]. Available: http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter. [Poskus dostopa 20 marec 2013].

[3] J. Kurose in K. Ross, Computer Networking, 5th edition, California: Pearson, 2010.

[4] „Mitsubishi automation,“ [Elektronski]. Available: http://www.mitsubishi-automation.com/products/compactplc_content.html. [Poskus dostopa 20 marec 2013].

[5] W. W. Royce, Managing the development of large software systems, California: Computer Society Press Los Alamitos, 1987.

[6] T. Mohorič, Podatkovne baze, Ljubljana: BI-TIM d.o.o., 2002.

[7] J. Ullman in J. Widom, First course in Database Systems, Upper Saddle Rive: Pearson Education, Inc., 2008.

[8] W. platform, „Open Web Platform - Wikipedia,“ Wikipedija, [Elektronski]. Available: http://en.wikipedia.org/wiki/Open_Web_Platform. [Poskus dostopa 20 marec 2013].

[9] B. S. J. B. J. Palermo, ASP.NET MVC in Action, Connecticut: Manning Publications Co. Greenwich, 2009.

[10] S. Kodarin, Avtomatizacija testiranja programske opreme, Ljubljana, 2011.