UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

MAGISTRSKO DELO

VPELJAVA SCADA SISTEMA NA PROIZVODNO LINIJO S POMOČJO ORODJA IGNITION

ANEJ MARUŠIČ

2019
MAGISTRSKO DELO
ANEJ MARUŠIČ

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Magistrsko delo

Vpeljava SCADA sistema na proizvodno linijo s pomočjo orodja Ignition

(Implementation of SCADA System on the Production Line With Ignition Software)

Ime in priimek: Anej Marušič
Študijski program: Računalništvo in informatika, 2. stopnja
Mentor: doc. dr. Peter Rogelj
Delovni somentor: univ. dipl. inž. Damijan Mihelj

Koper, avgust 2019

Ključna dokumentacijska informacija

Ime in PRIIMEK: Anej MARUŠIČ

Naslov magistrskega dela: Vpeljava SCADA sistema na proizvodno linijo s pomočjo orodja Ignition

Kraj: Koper

Leto: 2019

Število listov: 82 Število slik: 15 Število tabel: 17

Število referenc: 23

Mentor: doc. dr. Peter Rogelj

Delovni somentor: univ. dipl. inž. Damijan Mihelj

UDK: 004.05(043.2)

Ključne besede: SCADA, Ignition, sledljivost proizvodne linije, proizvodni informacijski sistemi, programsko inženirstvo

Izvleček:

V sodobni industriji težimo k vedno večji avtomatizaciji proizvodnje ter k čim večji sledljivosti izdelkov tekom proizvodnje. Tako omogočimo, da slabi izdelki ne pridejo do kupca, hkrati pa tudi v primeru reklamacij lažje ugotovimo razlog za reklamirano napako. Tako sledljivost izdelkov in vodenje proizvodne linije opravlja t. i. SCADA sistem. V podjetju, s katerim smo sodelovali pri izdelavi magistrske naloge, je na veliko linijah že implementiran t. i. hišni SCADA sistem. Postavljen je bil torej samostojno, s strani podjetja, brez pomoči kakršnihkoli namenskih orodij za implementacijo takšnih sistemov. V tej nalogi predstavljamo poskus postavitve novega SCADA sistema s pomočjo orodja *Ignition*. Namen projekta je razviti prototipni sistem, ki bo zahteval manj dela s strani programerjev, zaposlenih v podjetju ter omogočal lažjo integracijo na novih proizvodnih linijah, ki jih bo podjetje v prihodnosti nabavilo, hkrati pa bo še vedno vseboval vse funkcionalnosti, ki so sedaj že implementirane v hišnem SCADA sistemu. V magistrski nalogi je podrobno opisan razvoj ter delovanje tega sistema s pomočjo faz programskega inženirstva. SCADA sistem vsebuje strežniški del, na katerem se izvaja vsa glavna logika tega sistema. Ta strežnik komunicira z drugimi deli sistema; z odjemalci, podatkovno bazo ter napravami na proizvodni liniji. Tako strežnik kot odjemalci so implementirani z orodjem *Ignition*, ki preko protokola OPC UA komunicira s PLC-ji naprav. Vsi podatki iz proizvodne linije se shranjujejo v lokalno bazo Microsoft SQL Server.

Key words documentation

Name and SURNAME: Anej MARUŠIČ

Title of the thesis: Implementation of SCADA system on the production line with Ignition software

Place: Koper

Year: 2019

Number of pages: 82 Number of figures: 15 Number of tables: 17

Number of references: 23

Mentor: Assist. Prof. Peter Rogelj, PhD

Co-mentor: Damijan Mihelj, B.Sc

UDK: 004.05(043.2)

Keywords: SCADA, Ignition, traceability of the production line, production information system, software engineering

Abstract:

In the modern industry we aim to increase the automation of production and maximize the traceability of products during production. This way we manage to achieve that bad products do not reach the customers and in the case of complaints, it is easier to find the reason for the mistake that was made. The traceability of the products and the operation of the production line is performed by SCADA system. In the company with which we collaborated for creation of this Master's Thesis there are already a lot of production lines in which the so called home SCADA system is already implemented. The SCADA system was therefore set up independently by the company without the help of any purpose built tools for the implementation of such systems. In this thesis we present a plan for installation of a new SCADA system using the *Ignition* tool. The purpose of this project is to develop a prototype system that will require less work by the company programmers and allow easier integration into new production lines that the company will acquire in the future, while still containing all the functionalities that are already implemented in the house SCADA system. The Master's Thesis describes in detail the development and operation of this system through the phases of software engineering. The SCADA system encompasses a server where all the main logic of this system is implemented. This server communicates with other parts of the system - clients, database and devices on the production line. Both the server and the clients are implemented with the *Ignition* tool,

which communicates with the PLC devices via the OPC UA protocol. All data from the production line is stored in the Microsoft SQL Server database.

KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK

SEZNAM KRATIC

API	Application Programming Interface	Aplikacijski programski vmesnik
DBMS	Database Management System	Sistem za upravljanje s podatkovno bazo
DCOM	Distributed Component Object Model	Objektni model porazdeljenih komponent
DMC	Data Matrix Code	Dvodimenzionalna črtna koda
ERP	Enterprise Resource Planning	Sistem za načrtovanje virov podjetja
HTTP	HyperText Transfer Protocol	Protokol za izmenjavo hiperteksta
IP	Internet Protocol	Internetni protokol
JDBC	Java Database Connectivity	Javanska knjižnica za povezavo s podatkovno bazo
JDK	Java Development Kit	Javanska razvijalska oprema
MES	Manufacturing Execution System	Sistem za upravljanje proizvodnje
OLE	Object Linking and Embedding	Microsoftov mehanizem za povezovanje in vgrajevanje objektov
OPC	OLE for Process Control	OLE za nadzor nad procesi
OPC DA	OPC Data Access	OPC za dostop do podatkov
OPC UA	OPC Unified Architecture	OPC enotna arhitektura
PLC	Programmable logic controller	Programabilni logični kontroler
RAID	Redundant Array of Independent Disks	Standard povezovanja več neodvisnih trdih diskov v redundantno polje

SCADA	Supervisory Control And Data Acquisition	Sistem, namenjen nadzorovanju in krmiljenju različnih tehnoloških procesov z računalnikom
SDM	Splošno Delovno Mesto	
UDT	User Defined Types	Tipi definirani s strani uporabnika
UML	Unified Modelling Language	Poenoteni jezik modeliranja
UPS	Un-interruptable Power Supply	Brezprekinitveno napajanje
URL	Uniform Resource Locator	Enolični krajevnik vira
ZDM	Začetno Delovno Mesto	

ZAHVALA

Najprej se bi rad zahvalil svojemu mentorju, doc. dr. Petru Roglju, ter delovnemu somentorju Damijanu Mihlju za njun trud in čas, ki sta mi ga posvetila za usmerjanje pri izdelavi magistrskega dela. Istočasno bi se rad zahvalil tudi podjetju MAHLE Electric Drives Slovenia, ki mi je omogočilo sodelovanje in dovolilo uporabo teme magistrske naloge v povezavi s podjetjem.

Iz srca se pa predvsem zahvaljujem svoji družini, posebej svojima staršema Agati in Goranu, ki sta mi tekom celotnega šolanja vedno stala ob strani, me vzpodbujala in me podpirala ter svetovala z deljenjem svojih lastnih izkušenj. S finančno podporo sta mi omogočila študij v Kopru ter hkrati tudi vztrajanje pri aikidu – svojem hobiju, ki mi je nudil sprostitev v stresnih časih.

Zahvala gre tudi mojemu dekletu, Teji Sedmak, na katero sem se lahko tekom vseh let študija vedno zanesel. Razumevajoče je vedno odgovarjala na moje telefonske klice in mi vedno nudila oporo v vseh študijskih zadevah pa tudi izven njih.

Na koncu bi se še rad zahvalil svojim sošolcem na magistrskem študiju – Janu Bratini, Blažu Gombaču, Tomažu Grižonu ter Patriku Kocijančiču. Brez njihove pomoči bi se skozi študij nedvomno prebijal počasneje ter se veliko manj zabaval kot sicer. Podobno velja za moje sodelavce, ki so me opremili s potrebnim znanjem in mi bili na voljo za kakršna koli vprašanja glede projekta, predstavljenega v magistrski nalogi. Še posebej bi izpostavil sošolca in sodelavca Jana Bratino, s katerim sva se skupaj podala na pot pisanja vsak svoje magistrske naloge in se pri tem vzpodbujala in si svetovala ter naposled zaključila magistrski študij.

1 UVOD

V sodobni industriji težimo k vedno večji avtomatizaciji in čim manjši odvisnosti proizvodnega cikla določenega izdelka od delavca na proizvodnji liniji. Kljub temu, da je na proizvodni liniji vedno manj ljudi zadolženih za določeno delovno mesto in so delavci primerno izobraženi za delo, ki ga opravljajo, še vedno prihaja do napak delavcev, zaradi katerih lahko izdelek s slabo kakovostjo pride do kupca. Podobno lahko pride tudi do napak, ki jih zakrivi naprava zaradi slabega delovanja, saj proizvajalec naprave namreč težko zagotovi stoodstotno zanesljivost.

Veliko od zgoraj omenjenih napak razrešujejo že same naprave z različnimi testiranji in tako delavca obvestijo o neustrezni kvaliteti izdelka, vendar to velikokrat ni dovolj. Kajti lahko se zgodi, da delavec spregleda opozorilo in izdelek preda na naslednjo delovno postajo. Na enak način pa tudi v primeru, ko delavec ravna pravilno, omenjeno uniči ritem proizvodnje (delavec mora izdelek odstraniti in zapisati, zakaj je bil izdelek odstranjen) in se tako podaljša cikel proizvodnje posameznega izdelka.

Iz omenjenega je precej razvidno, zakaj je težnja k čim večji avtomatizaciji proizvodnje vedno višja. K avtomatizaciji posameznega delovnega mesta na proizvodni liniji večinoma prispeva stroj sam, potrebujemo pa sistem, ki bo povezoval delovna mesta in odločanja o ustrezno opravljeni operaciji na delovnem mestu ne bo prepuščal le delavcu samemu.

Na enak način pa nam veliko naprav na posameznih delovnih mestih nudi tudi veliko drugih podatkov, ki vplivajo na kakovost izdelka in nam tako lahko s primerno analizo le-teh dolgoročno pomagajo priti do marsikaterih ugotovitev. Prav zato je lahko koristno, če se omenjene podatke shranjuje in naknadno analizira. S shranjevanjem teh podatkov bi lahko tako omogočili tudi boljši nadzor nad izmetom iz redne proizvodne linije brez nepotrebne obremenitve delavca z zapisovanjem razloga za izmet – kar bi med drugim pospešilo tudi proizvodni cikel.

Problem shranjevanja podatkov iz proizvodne linije in problem nadzora nad proizvodno linijo sta tako samo dve plati iste medalje. Obstaja veliko rešitev, ki rešujejo omenjena problema, zato se soočamo tudi s problematiko, katero od teh rešitev uporabiti. Izbira rešitve je seveda odvisna od veliko faktorjev znotraj podjetja – zahtev podjetja, razpoložljive delovne sile, finančne zmogljivosti in že utečenih praks ter sistemov v podjetju. Vsekakor pa je končen cilj pridobiti rešitev, ki bom čim bolj univerzalna in bo pripomogla k čim večjemu dobičku podjetja.

Vse zgoraj omenjene probleme rešuje SCADA sistem na proizvodni liniji. V tej magistrski nalogi je opisana implementacija takega prototipnega SCADA sistema na eni izmed proizvodnih linij v podjetju.

1.1 DEFINICIJA PROBLEMA

V podjetju je na veliko linijah že implementiran SCADA sistem, ki je bil postavljen samostojno, brez orodij za implementacijo SCADA sistema, ki bi olajšala postavljanje in predvsem vzdrževanje takega SCADA sistema. Prednost takega, v podjetju razvitega SCADA sistema je ta, da imamo popolno svobodo implementacije kakršnih koli funkcionalnosti ter želenih modifikacij sistema, ki jih morebiti potrebujemo in nismo omejeni z omejitvami SCADA orodja. Velika slabost pa je pomanjkanja kadra v podjetju, ki bi ta sistem lahko podpiral. Hišni SCADA sistem namreč potrebuje zadostno število vzdrževalcev ter integratorjev, ki pa bi podjetje preveč stali. Prav tako potrebujemo programerje za nadaljnji razvoj SCADA sistema, ki je prav tako nujno potreben. Ko govorimo o nadaljnjem razvoju, s tem mislimo tako na razvoj že obstoječih komponent SCADA sistema, ki jih je potrebno razvijati, da ostanejo v koraku s časom, kot dodajanje morebitnih novih funkcionalnosti, ki jih sistem še nima.

Zato iščemo alternativo obstoječemu SCADA sistemu, ki bo zahtevala manj dela s strani programerjev, zaposlenih v podjetju, ter omogočala lažjo integracijo na nove proizvodne linije, ki jih bo podjetje v prihodnosti nabavilo, hkrati pa bo še vedno vsebovala vse funkcionalnosti, ki so sedaj že implementirane v hišnem SCADA sistemu. Želeli bi tudi, da nadaljnji razvoj ne bi več potreboval toliko delovne sile kot jo je do sedaj, ter da bi bilo dodajanje novih funkcionalnosti kar se da enostavno. Alternativa lahko vsebuje uporabo namembne programske opreme za implementacijo SCADA sistemov.

Prototip izbrane alternative je potrebno implementirati na novo zgrajeni liniji v podjetju, na kateri se izdeluje dotičen izdelek. Proizvodna linija vsebuje štirinajst delovnih mest, ki glede SCADA sistema delujejo več ali manj enako in jih lahko zato obravnavamo na enak način. Izjema je začetno delovno mesto, ki se malce razlikuje od delovanja drugih.

1.2 UMESTITEV SCADA SISTEMA

SCADA sistem je en izmed gradnikov informacijskega sistema podjetja. Tesno je povezan s pojmoma ERP in MES [20].

Sistem ERP (*Enterprise Resource Planing*) pomaga pri načrtovanju virov v organizaciji. Vključuje nakup materiala, upravljanje z zalogami in upravljanje s strankami ter delovnimi nalogi. Prav tako se preko ERP načrtuje proizvodnja in upravlja z logistiko ter tudi upravlja s kadrom. ERP tako predstavlja nekakšen most med dobavitelji in strankami ter informacijskim sistemom podjetja [18].

Naloga sistema MES (*Manufacturing Execution System*) je analiziranje ter prikaz podatkov, ki jih SCADA sistem pridobi iz proizvodne linije. S pomočjo tega se lahko

optimizira proizvodni proces ter pripravlja razna poročila in nadzoruje vhodne ter izhodne podatke v SCADA sistem. Glavni cilj takega sistema je s povezovanjem proizvodnega ter poslovnega okolja zagotoviti učinkovito izvajanje proizvodnih operacij in izboljšati donos proizvodnje [13].

Funkcije SCADA (*Supervisory Control And Data Acquisition*) sistema so že opisane v poglavju 1, vendar je pomembno, da vemo, kje je SCADA sistem umeščen v podjetju. Na najnižjem nivoju imamo različne senzorje in aktuatorje, ki samo zajemajo podatke. S skupkom le-teh, ki tvorijo napravo, upravljajo PLC-ji (*Programmable Logic Controller*), ki so sprogramirani, da na določene načine uporabljajo senzorje in aktuatorje. Podatke, ki jih PLC-ji pridobijo s pomočjo teh senzorjev, SCADA sistem zajema in beleži v podatkovno bazo. Hkrati tudi odgovarja in s tem nadzoruje delovanje PLC-ja. Preko podatkovne baze nato SCADA sistem interaktira s sistemom MES, ki opravi svoje delo, omenjeno v prejšnjem odstavku, in sicer analizira podatke in pripravi poročila ter jih pošlje v sistem ERP. V nasprotni smeri pa sistem MES zagotovi SCADA sistemu razne podatke (delovni nalogi, navodila napravam…), ki jih SCADA potrebuje za učinkovito nadziranje, odvzem in beleženje podatkov iz linije. ERP od sistema MES prejme podatke, ki jih zatem pripravi za interakcijo z drugimi sistemi in upravljanjem z viri, MES sistemu pa zagotovi ustrezna navodila, ki jih oblikuje glede na zahteve kupcev ter razpoložljive vire [20].

1.3 OPIS OBSTOJEČEGA SISTEMA

Kot smo že omenili v poglavju 1.1, imamo v podjetju na veliko linijah že implementiran hišni SCADA sistem. *Back-end* oz. strežniški del tega sistema je implementiran v javanskem ogrodju *Spring*, medtem ko je *front-end* oz. odjemalec, sprogramiran s pomočjo platforme *Angular*. Vsa glavna logika za upravljanje posameznega delovnega mesta se odvija na odjemalcu. Na vsakem delovnem mestu je lociran mini računalnik *Raspberry Pi*, ki preko brskalnika prejema podatke iz strežnika, ki jih gosti na strežniku *Apache* [11].

Odjemalec lahko tako preko strežnika dostopa do podatkovne baze ter tudi do PLC-jev naprav. Slednje je mogoče le zaradi orodja KEPServerEX, ki služi kot strežnik OPC za povezavo na naprave PLC . Poleg tega obstoječi sistem, v kolikor je to potrebno, nudi tudi možnost branja in ustvarjanja ter pošiljanja datotek za izmenjavo informacij z napravami [11].

2 ŠTUDIJA IZVEDLJIVOSTI

V študiji izvedljivosti moramo preučiti, ali se je smiselno lotiti reševanja definiranega problema [21]. Pri iskanju rešitve se moramo zavedati, da se zaradi časa in financ, ki jih imamo na razpolago, ne moremo zanašati na to, da nam bo uspelo zagotoviti rešitev problema za celotno proizvodno linijo. Zato se lahko v prototipnem projektu osredotočimo samo na nekaj delovnih mest, ki bi nam lahko dala nekakšen vpogled v to, ali je problem SCADA sistema, smiselno na celotni proizvodni liniji reševati na način, ki ga bomo aplicirali na teh nekaj delovnih mestih.

Prav zaradi zgoraj omenjenega problema bomo študijo izvedljivosti, kjer je to potrebno, razdelili na dva dela. V takšnih primerih bomo najprej izvedli študijo izvedljivosti celotnega projekta, nato pa še prototipnega projekta. S prototipnim projektom je namreč smiselno začeti samo, če je rezultat študije izvedljivosti celotnega projekta zadovoljiv.

2.1 TEHNIČNA IZVEDLJIVOST

Pri tehnični izvedljivosti lahko istočasno obravnavamo celoten projekt in prototipni projekt, saj ne bi prišlo do razlik, če bi omenjeno obravnavali posebej.

V tem razdelku se moramo zavedati glavnih komponent, ki jih moramo implementirati. Lahko rečemo, da imamo dve, in sicer podatkovno bazo ter SCADA komponento.

Okvirno torej vemo, da potrebujemo nekakšen strežnik od koder bo potekalo upravljanje linije s pomočjo komponente SCADA. Na neki oddaljeni lokaciji ali tudi na tem strežniku imamo lahko tudi podatkovno bazo, ki je lahko implementirana z enim od več možnih sistemov za upravljanje podatkovne baze (DBMS).

Poleg strežnika na liniji na vsaki delovni postaji potrebujemo tudi prikazovalnike, ki bodo informacije delovne postaje prikazovali delavcu na tej operaciji. Zasloni bodo seveda morali biti povezani vsak s svojim računalnikom, na katerem bo tekel program, ki bo pridobival informacije za prikazovanje.

Za implementacijo komponente SCADA bomo uporabili programsko orodje **Ignition** podjetja **Inductive Automation** [8]. To orodje potrebuje specifično znanje, ki ga je potrebno osvojiti. Imamo možnosti udeležbe vodenega izobraževanja ali tudi samoizobraževanja s spletno učilnico podjetja, ki razvija *Ignition* [9]. Pri samoizobraževanju lahko pomaga tudi bogata dokumentacija [6]. Tehnično sta izvedljivi obe varianti, prav tako tudi sama uporaba *Ignition* orodja.

Da bo projekt tehnično izvedljiv, potrebujemo tudi sodelovanje proizvajalcev proizvodnje linije. Od njih moramo pridobiti informacije o postopku, po katerem delujejo naprave na liniji ter zahtevati, da so na voljo za kakršnakoli vprašanja o poteku operacij na delovnih

mestih. Ker proizvajalci delujejo v sklopu podjetja, vprašanje pripravljenosti za sodelovanje ni problem, saj je tudi njim v interesu zagotoviti čim večjo uspešnost delovanja proizvodnih linij.

2.2 OPERATIVNA IZVEDLJIVOST

Pri izvajanju operativne izvedljivosti se moramo zavedati, da imamo v podjetju že implementiran SCADA sistem in da bomo v projektu skušali implementirati nov SCADA sistem, ki bi dolgoročno bolje zadovoljil potrebe podjetja ter mu tako prinesel več zaslužka. Torej je na mestu vprašanje, ali in kako bo podjetje sprejelo nov sistem, ki je cilj tega projekta. Glede na to, da preizkušamo nov SCADA sistem, ki prej še nikjer ni bil implementiran s strani podjetja, se moramo tudi zavedati, da se bo bržkone večina težav pojavljala zaradi pomanjkanja izkušenj z orodjem *Ignition*.

Želeno je, da bo prehod na nov sistem čim bolj transparenten, tako da bo imel čim manj vpliva na končne uporabnike. Več ali manj se bo namreč spreminjalo le ogrodje SCADA sistema. Tako bodo vhodni in izhodni podatki sistema več ali manj enaki kot prej, kar pomeni, da se za končne uporabnike ne bo spremenilo tako-rekoč nič, razen malenkosti pri uporabniškem vmesniku, ki pa v osnovi in večini funkcionalnosti ostaja enak.

Istočasno z implementacijo novega SCADA sistema podjetje namerava na isto proizvodno linijo implementirati tudi verzijo že obstoječega SCADA sistema v podjetju. Oba SCADA sistema bosta zasnovana kompatibilno – zagotovljeno je, da bo z dvema različnima delovnima mestoma lahko upravljal različni SCADA sistem, tudi če sta delovni mesti na isti proizvodni liniji. Iz tega torej sledi, da bo imelo podjetje manj izgube v primeru težav pri implementaciji, saj bomo lahko nov sistem testirali vzporedno z že obstoječim in bomo v primeru velikih zapletov lahko uporabljali stari SCADA sistem za kakršnakoli druga testiranja delovanja nastajajoče proizvodne linije.

Obenem je vzporedno implementiranje obeh SCADA sistemov dobro tudi v primeru, če bo potrebno nov SCADA sistem karkoli spreminjati, saj bomo vedno imeli star SCADA sistem, ki bo lahko začasno zamenjal novega, ki bo v testiranju. Še ena dobra stran je tudi ta, da si lahko v tem primeru privoščimo več testiranja in planiranja pri prototipnem projektu in ni potrebna takojšnja, prehitra evalvacija prototipnega sistema ter posledično prehiter prehod na gradnjo končnega celotnega sistema.

2.3 EKONOMSKA IZVEDLJIVOST

Pri ekonomski izvedljivosti imamo najjasnejšo ločnico med prototipnim projektom in projektom gradnje celotnega sistema. Okvirni izračun stroškov zahteva najprej približno

oceno licenciranja ter oceno števila ljudi, ki bodo sodelovali pri projektu, ki pa se razlikuje glede na to, za katero različico projekta gre.

2.3.1 Ekonomska izvedljivost celotnega sistema

Programiranje in načrtovanje celotnega SCADA sistema zahteva delo več projektantov in programerjev. Tako moramo poleg stroška programerja avtorja te magistrske naloge prišteti še stroške najmanj enega projektanta programerja. Zraven moramo prišteti še strošek dela zunanjega integratorja *Ignition* SCADA sistema, ki bo pomagal pri prehodu na nov sistem ter manjše stroške pomoči programerjev naprav PLC, od katerih bomo želeli pridobiti razne, prej še neznane informacije glede proizvodne linije.

Integrator *Ignition* SCADA sistema omogoča tudi delavnice za pridobitev osnov dela z omenjenim orodjem. Smiselno bi bilo, da bi se izobraževanja udeležili vsi neposredno ter tudi posredno vpleteni v gradnjo novega SCADA sistema, saj bi tako vsi najhitreje pridobili ustrezno znanje za uporabo orodja in integracijo novega SCADA sistema ter se seznanili s prakso uporabe orodja. Takšno dvodnevno izobraževanje bi podjetje stalo približno 1.400 €.

Naslednji finančni zalogaj je cena licence programskega orodja *Ignition*. *Ignition* je modularno orodje, ki nam omogoča popolno svobodo pri izbiri modulov, ki jih hočemo uporabiti – torej lahko kupimo samo tiste module, ki jih bomo resnično potrebovali tekom projekta. Tako se je torej v fazi načrtovanja potrebno odločiti, katere module uporabiti, da nam ti ne bodo preveč zvišali cene licence, hkrati pa ne bomo zaradi pomanjkanja le-teh imeli veliko preveč dela – kar bi torej pomenilo zakasnitev zaključka projekta in dodatne stroške za delovno silo, ki finančno povsem odtehtajo katero od dodatnih licenc za določen modul.

Če hitro ocenimo želje podjetja na podlagi SCADA sistema, lahko privzamemo, da bo naša verzija dokaj osnovna. Podjetje namreč nima interesa porabe finančnih sredstev za nekatera orodja, ki neprogramerjem olajšajo delo z orodjem, prav tako pa nima zahteve po uporabi nekaterih dodatnih modulov, ki ne spadajo k osnovnem delovanju SCADA sistema. Tako bi se cena licence gibala okoli 10.000 dolarjev (okoli 9.000 €) [7].

Za implementacijo podatkovne baze si lahko privoščimo brezplačni sistem za upravljanje podatkovne baze, dodaten strošek pa je še nakup strojne opreme za projekt. Potrebujemo strežnik z vsaj 8 GB notranjega pomnilnika, 250 GB SSD-diskom ter 3 GHz procesorjem, na katerem bo lahko brezhibno deloval strežnik *Ignition* ter podatkovna baza, hkrati pa še nekateri deli sistema MES. Tak računalnik skupaj s tipkovnico, miško ter enostavnim zaslonom, stane od 1000 do 1500 €. Poleg tega pa potrebujemo tudi računalnike, na katerih bodo prikazovalniki na vsakem od 14 delovnih mest. SCADA sistem, ki je trenutno v uporabi v podjetju, za prikazovalnike uporablja *Raspberry Pi 3* računalnike [11], kar bi

lahko bilo dovolj tudi za prikazovalnike v sistemu *Ignition,* vendar je potrebno testiranje, ali so omenjeni računalniki dovolj zmogljivi. Cena takih računalnikov je okoli 50 €, cenejših prikazovalnikov na dotik pa okoli 250 €. V primeru, da bo testiranje na *Raspberry Pi* računalnikih neuspešno, pa bomo potrebovali majhne industrijske računalnike, katerih cena je najmanj štirikratnik cene *Raspberry Pija*.

Vsem stroškom navkljub se je treba zavedati tudi koristi izvedbe projekta. Pričakujemo, da bomo z uspešno izvedenim projektom pridobili novo prakso implementacije SCADA sistema v podjetju s pomočjo orodja *Ignition*. Tako bomo lahko ta nov sistem v prihodnje uporabljali tudi na novo nastalih proizvodnih linijah v podjetju in bomo posledično lahko opuščali starega. Star sistem namreč za vsako implementacijo zahteva veliko dodatnega dela s strani programerjev, kar pričakujemo, da v primeru novega ne bo veljalo. Pri novem sistemu pričakujemo tudi večjo stabilnost sistema, tudi zaradi podpore, ki jo omogoča proizvajalec orodja *Ignition,* kar pomeni, da bo podjetje lahko zagotovilo tudi višjo kvaliteto pri produktih, proizvedenih pod okriljem novega sistema ter večjo količino produktov v določenem časovnem obdobju, ker bo prihajalo do manj zastojev. Prav tako zaradi stabilnosti pričakujemo tudi manjšo potrebo podjetja po dežurnih delavcih. Navkljub relativno velikim začetnim stroškom pri prvem projektu implementacije SCADA sistema s pomočjo orodja *Ignition,* lahko tako pri vseh nadaljnjih implementacijah pričakujemo veliko manjše stroške zaradi delovne sile, kar je v podjetju dobrodošlo, saj programerjev primanjkuje in so preobremenjeni. Stroški licenciranja bodo medtem za vsako proizvodno linijo ostali isti.

2.3.2 Ekonomska izvedljivost prototipnega sistema

Ekonomska izvedljivost prototipnega sistema je v marsičem skladna z ekonomsko izvedljivostjo celotnega sistema. Razlike so v tem, da se lahko prototipnega projekta loti le ena oseba samostojno, brez integratorja in le z nasveti drugih programerjev. Prav tako je za spoznavanje z orodjem *Ignition* dovolj le bogata spletna učilnica tega področja [9]. Omenjeno torej pomeni, da lahko odštejemo stroške dodatne delovne sile ter izobraževanja.

Zmanjšani so tudi stroški glede licenciranja orodja *Ignition*. To orodje namreč omogoča brezplačno licenco z vsemi moduli, ki jo je treba podaljševati vsake dve uri, kar pa je dovolj za prototipni projekt in posledično odločitev o nadaljevanju gradnje projekta v celotni projekt.

Pri strojni opremi so stroški prototipnega projekta enaki, s to razliko, da bomo sistem testirali le na dveh delovnih mestih in torej potrebujemo le dva prikazovalnika namesto štirinajstih. Pričakovano so seveda ob uspešno zasnovanem prototipnem projektu manjše tudi finančne koristi. Poleg pridobljenega znanja o gradnji sistema s pomočjo orodja *Ignition,* dokončan prototipni projekt namreč pomeni le, da lahko začnemo z gradnjo

celotnega projekta. Ti dve dejstvi pa prinašata dovolj veliko korist, saj imamo potemtakem ob neuspelem projektu manj izgub, kot če bi iz začetka gradili celotni projekt in bi prišli do ugotovitve o neustreznosti le-tega. Ob neuspelem prototipnem projektu se seveda lahko tudi strojna oprema uporabi drugje v podjetju (ves čas se namreč gradijo nove linije, ki delujejo na podoben princip), torej podjetje v tem primeru izgubi le, kar je prispevalo za plačilo projektanta (avtorja magistrske naloge).

2.4 KONČNE UGOTOVITVE

Po opravljeni študiji izvedljivosti ugotovimo, da bo poleg izvedbe faz programskega inženirstva, potrebne tudi veliko organizacije. Ustrezno se moramo namreč povezati z vzdrževalci linije ter s programerji naprav na liniji, prav tako pa moramo podjetju utemeljiti želje po nakupu nove strojne opreme in licenc ter čakati na njihovo odobritev. Sklenemo lahko, da je celoten projekt tehnično izvedljiv, zahteval pa bo finančni vložek podjetja v vrednosti približno 16.500 €, brez stroškov financiranja dveh projektantov programerjev. Koristi, ki jih prinese projekt, zadevajo predvsem zmanjšanje stroškov pri vzdrževanju po uspešno izvedenem projektu ter zmanjšanje stroškov pri prenosu sistema na neko novo proizvodno linijo.

Končna odločitev za zagon projekta je tako prepuščena vodstvu. Vsekakor pa je bolj smiselno, če najprej začnemo s prototipnim projektom. Skupni finančni vložek pri gradnji prototipnega projekta znaša namreč le okoli 2.500 €, če izključimo stroške financiranja enega projektanta programerja. Tekom gradnje prototipnega projekta pa bomo ugotovili ustreznost novega sistema ter pridobili nova znanja in brez veliko stroškov pridobili argumente, ki nam bodo olajšali odločitev o nadaljevanju gradnje sistema. Prav tako bodo argumenti lahko prepričali vodstvo podjetja v lažjo odločitev o večjem vložku v gradnjo celotnega sistema - nakupu licenc, izobraževanj in dodatne strojne opreme.

3 SPECIFIKACIJA ZAHTEV

Pri specifikacij zahtev obravnavamo funkcijske ter nefunkcijske zahteve [21]. Pri analizi zahtev smo uporabili že obstoječ SCADA sistem na proizvodnih linijah v podjetju. Zaradi izkušenj s SCADA sistemom nam je bila torej analiza zahtev lažja kot sicer.

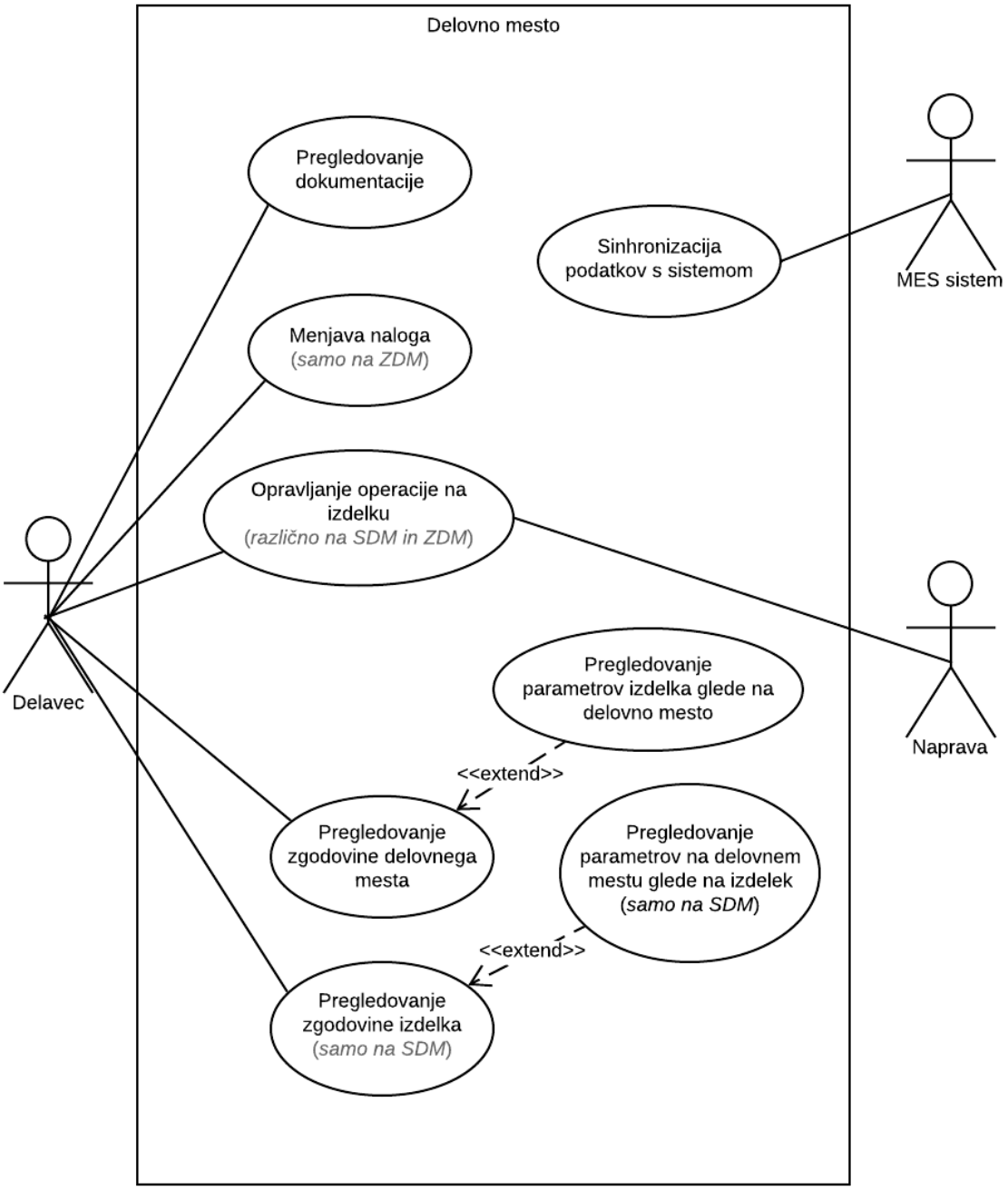
Odločili smo se, da bomo poskusili implementirati SCADA sistem na dveh napravah. Ena izmed njih je naprava, katere komunikacija s SCADA sistemom je dokaj reprezentativna za celo linijo. V nadaljevanju bo to delovno mesto poimenovano kot splošno delovno mesto (SDM). Tudi večina drugih naprav deluje na enak način, kar se tiče SCADA sistema. S tem, ko implementiramo SCADA sistem na tem delovnem mestu, lahko predvidoma brez večjih naporov prenesemo dobljeno rešitev tudi na več drugih delovnih mest.

Naslednje delovno mesto je začetno delovno mesto (ZDM) na liniji. Osnova delovanja tega delovnega mesta je enaka SDM s tem, da ima to še nekatere dodatne funkcije. Izbrali smo ga kot eno izmed delovnih mest, ki predstavlja tista delovna mesta, ki so drugačna od SDM. Več ali manj je splošno delovanje vseh podobno SDM, s tem da imajo še nekatere dodatne korake, ki se med seboj razlikujejo. Lahko sklepamo, da se ti koraki v delovanje SDM vključijo na podoben način kot koraki ZDM. Prav zato smo dobili zamisel, da bi poskusili implementirati SCADA sistem na tem ZDM.

3.1 FUNKCIJSKE ZAHTEVE

V tem razdelku bomo skupaj obravnavali obe delovni mesti. V tistih funkcijah sistema, kjer se SDM in ZDM razlikujeta, bomo to poudarili in razlikovali med delovanjem enega in drugega. Osnovne funkcije sistema zajemajo menjavo delovnega naloga, pregledovanje dokumentacije, sinhronizacijo podatkov s sistemom, pregledovanje zgodovine izdelka ter zgodovine delovnega mesta in pripadajočih procesnih parametrov na posameznem delovnem mestu ter opravljanje osnovne operacije na izdelku na delovnem mestu skupaj z upravljanjem z recepti. Recepti so navodila PLC-ju posamezne naprave, kako naj deluje v odvisnosti od izdelka, ki se trenutno nahaja na postaji.

Funkcijske zahteve bomo predstavili s pomočjo enega diagrama primera uporabe za obe delovni mesti ter opisa le-tega. Nekateri primeri uporabe bodo izraženi samo na ZDM nekateri pa le na SDM.



Slika 1: Diagram primera uporabe za delovno mesto

Diagram primera uporabe na sliki 1 prikazuje interakcijo sistema z akterji - delavcem, napravo na liniji ter MES sistemom podjetja – ter načine uporabe, na katere lahko akterji uporabljajo sistem. Akter delavec predstavlja osebo, ki je na proizvodni liniji in upravlja s posameznim delovnim mestom. Akter MES sistem predstavlja že implementiran MES sistem podjetja, ki na različne načine interaktira z našim SCADA sistemom. Akter naprava pa predstavlja napravo, ki jo ima vsako od delovnih mest in ki navadno opravi zahtevno operacijo na izdelku, ki jo sproži delavec.

Tako na SDM kot na ZDM so izraženi primeri uporabe *sinhronizacije podatkov s sistemom, pregledovanje dokumentacije, pregledovanje zgodovine delovnega mesta* in *pregledovanje parametrov izdelka glede na delovno mesto*. Vsak od teh navedenih primerov uporabe je nespremenjen ne glede na to, o katerem od dveh delovnih mest govorimo. To pa ne velja za glavni primer uporabe - *opravljanje operacije na izdelku*. Ta se razlikuje glede na delovno mesto, na katerem je izražen. *Opravljanje operacije na izdelku* na ZDM ima tako nekoliko več korakov od enakega primera uporabe na SDM. Slednji primer uporabe bo tako opisan povsem ločeno glede na to ali se ta nahaja na enem ali drugem delovnem mestu.

Ostali primeri uporabe, ki niso navedeni v prejšnjem odstavku, pa so izraženi le na enem delovnem mestu. Na SDM so to pregledovanje zgodovine izdelka in pregledovanje parametrov na delovnem mestu glede na izdelek, na ZDM pa menjava naloga.

3.1.1 Opisi primerov uporabe

V tem podpoglavju sledi opis primerov uporabe iz diagrama primera uporabe, ki je prikazan v prejšnjem poglavju (3.1) na sliki 1. Vsak primer uporabe bo najprej na kratko obrazložen za boljše razumevanje celotnega sistema. Posamezen primer uporabe je izražen tako na SDM kot na ZDM, razen, če je podatek o izraženosti eksplicitno napisan v obrazložitvi. Po vsaki obrazložitvi bo sledil podroben opis pripadajočega primera uporabe, v kolikor je le-ta bil predmet tega prototipnega projekta.

3.1.1.1 Sinhronizacija podatkov s sistemom

V dani primer uporabe je vpleten le akter MES. Preko tega sistem MES sinhronizira in dostavi v sistem vse potrebne podatke, ki jih od njega potrebujemo. Za podatke, kot so nalogi in vse informacije o nalogih ter recepti, to naredi tako, da jih shrani v podatkovno bazo. Pri dostavi dokumentacije je to malo drugače. Dokumentacijo gosti na določenem strežniku. Do spletne aplikacije, ki prikazuje in omogoča pregledovanje dokumentacije, naš sistem dostopa preko URL-ja.

Do teh podatkov, ki jih je MES na ta način dovedel do našega sistema, torej dostopamo in jih uporabljamo v drugih primerih uporabe. Sistem MES je upravljan s strani drugih programerjev in tako ni bil predmet tega projekta, zato nam njegovo natančno delovanje ni popolnoma znano. Pri projektu vemo le, kako uporabljati nekatere njegove izhodne podatke.

3.1.1.2 Pregledovanje dokumentacije

Pregledovanje dokumentacije služi delavcu na posameznem delovnem mestu, da lahko ta pregleda različna navodila, kako obdelovati izdelek na želenem delovnem mestu na proizvodni liniji. Tako npr. pridobi informacije, kakšen vijak priviti v katero luknjo, kam vstaviti kakšen material, kam zalepiti nalepko ipd. Dokumentacijo SCADA sistemu priskrbi MES sistem preko primera uporabe *sinhronizacija podatkov s sistemom*.

Preglednica 1: Opis primera uporabe pregledovanja dokumentacije

Naziv	Pregledovanje dokumentacije
Akterji	Delavec
Zahteve	Uporabniški vmesnik je na začetnem zaslonu in delavec želi pregledati dokumentacijo tipa izdelka. Sistem mora zagotoviti ustrezno šifro naloga, katerega dokumentacijo se pregleduje. Na ZDM se to zagotovi preko naloga, ki ga je izbral delavec, na SDM pa šifro naloga pridobimo s pomočjo DMC kode izdelka, ki je bila nazadnje odčitana na tem delovnem mestu.
Prožilec	Delavec pritisne na gumb »Dokumentacija«.
Osnovni potek	1. Na zaslonu se prikaže eden od dokumentov iz dokumentacije za trenutno delovno mesto 2. Delavec s pritiskom na enega od gumbov na desni strani zaslona izbere dotičen dokument iz trenutno odprte dokumentacije za pregled 3. Delavec pregleduje dokumentacijo z uveljavljenimi gestami za upravljanje s slikami na zaslonih na dotik (približevanje in premikanje slike) 4. Delavec se s pritiskom na gumb »Nazaj« vrne na začetni zaslon ter zaključi primer uporabe
Stanje po zaključku	Sistem je spet na začetnem zaslonu uporabniškega vmesnika.
Alternativni poteki	4a: Delavec želi pregledati dokumentacijo drugega delovnega mesta od trenutnega 4a1. Delavec na spodnjem delu zaslona izbere delovno mesto, katerega dokumentacijo hoče pregledati s pritiskom na gumb s šifro želenega delovnega mesta 4a2. Primer uporabe se nadaljuje na 2. koraku osnovnega poteka 4b: Delavec želi pregledati drugi dokument iz trenutno odprte dokumentacije 4b1. Delavec s pritiskom na drugi gumb na desni strani zaslona izbere drugi dokument iz dokumentacije za pregled 4b2. Primer uporabe se nadaljuje na koraku 3 osnovnega poteka

3.1.1.3 Pregledovanje zgodovine delovnega mesta

Delavec lahko s pomočjo tega primera uporabe pregleda osnovne podatke izdelkov, ki so bili pred trenutnim na delovnem mestu, na katerem se nahaja. S pritiskom na določen izdelek lahko tudi preide na primer uporabe *pregledovanje parametrov izdelka glede na delovno mesto*.

Preglednica 2: Opis primera uporabe pregledovanja zgodovine delovnega mesta

Naziv	Pregledovanje zgodovine delovnega mesta
Akterji	Delavec
Zahteve	Uporabniški vmesnik je na začetnem zaslonu in delavec želi pregledati zgodovino delovnega mesta.
Prožilec	Delavec pritisne na gumb »Zgodovina delovnega mesta«.
Osnovni potek	1. Na zgornji polovici zaslona se prikažejo osnovni podatki (serijska številka, čas prihoda na delovno mesto, šifra tipa izdelka, šifra naloga, kakovost, šifra in opis napake in naslednje delovno mesto) zadnjih 20 izdelkov, katerih koda DMC je bila odčitana na tem delovnem mestu 2. Delavec pregleduje osnovne podatke zadnjih 20 izdelkov 3. Delavec se s pritiskom na gumb »Nazaj« vrne na začetni zaslon ter zaključi primer uporabe
Stanje po zaključku	Sistem je spet na začetnem zaslonu uporabniškega vmesnika.

3.1.1.4 Pregledovanje parametrov izdelka glede na delovno mesto

Delavec želi ob pregledu zgodovine delovnega mesta pregledati še podrobnosti izbranega izdelka, ko je ta bil na tem delovnem mestu, ter pritisne na zapis o želenem izdelku. Tedaj se mu prikažejo vsi parametri izdelka na delovnem mestu, na katerem se nahaja. Tako lahko nato pregleduje parametre in iz njih potegne razne zaključke glede izdelka ali glede naprave na tem delovnem mestu.

Preglednica 3: Opis primera uporabe pregledovanja parametrov izdelka na delovnem mestu

Naziv	Pregledovanje parametrov izdelka na delovnem mestu
Akterji	Delavec
Zahteve	Delavec pregleduje zgodovino delovnega mesta in želi pregledati parametre določenega izdelka na tem delovnem mestu.
Prožilec	Delavec pri pregledu zgodovine delovnega mesta s pritiskom na enega od izdelkov izbere izdelek, za katerega hoče pregledati parametre na trenutnem delovnem mestu.
Osnovni potek	1. Na spodnji polovici zaslona se prikažejo vsi procesni parametri izbranega izdelka, ki so bili zapisani tekom obdelovanja izbranega izdelka na delovnem mestu na katerem se delavec nahaja 2. Delavec pregleduje parametre izbranega izdelka na trenutnem delovnem mestu. 3. Delavec se s pritiskom na gumb »Nazaj« vrne na začetni zaslon ter zaključi primer uporabe
Stanje po zaključku	Sistem je spet na začetnem zaslonu uporabniškega vmesnika.
Alternativni poteki	2a: Delavec bi rad pregledal parametre drugega izdelka namesto izbranega 2a1. Delavec na zgornjem delu zaslona izbere drugi izdelek, katerega parametre hoče videti 2a2. Delavcu se na spodnjem delu zaslona prikažejo procesni parametri na novo izbranega izdelka, ki so bili zapisani tekom obdelovanja izbranega izdelka na delovnem mestu na katerem se delavec nahaja 2a3 Delavec pregleduje parametre na novo izbranega izdelka na trenutnem delovnem mestu (zopet je na 2. koraku osnovnega poteka)

3.1.1.5 Pregledovanje zgodovine izdelka

Delavec lahko s pomočjo tega primera uporabe pregleda osnovne podatke izdelka iz vseh delovnih mest na katerih je do zdaj ta že bil. S pritiskom na določeno delovno mesto lahko nato tudi preide v primer uporabe *pregledovanje parametrov na delovnem mestu glede na izdelek.* Ta primer uporabe je izražen le na SDM.

Preglednica 4: Opis primera uporabe pregledovanja zgodovine izdelka

Naziv	Pregledovanje zgodovine izdelka
Akterji	Delavec
Zahteve	Uporabniški vmesnik je na začetnem zaslonu.
Prožilec	Delavec pritisne na gumb »Zgodovina izdelka«.
Osnovni potek	1. Na zgornji polovici zaslona se prikažejo osnovni podatki (šifra in opis delovnega mesta, čas prihoda na delovno mesto, kakovost, šifra in opis napake in naslednje delovno mesto) izdelka, katerega koda DMC je bila odčitana na tem delovnem mestu. Ti podatki se prikažejo za vsako od delovnih mest, na katerih je do zdaj izdelek že uspešno ali neuspešno opravil operacijo 2. Delavec pregleduje osnovne podatke iz vseh prejšnjih delovnih mest izdelka, katerega koda DMC je bila odčitana na tem delovnem mestu 3. Delavec se s pritiskom na gumb »Nazaj« vrne ne začetni zaslon ter zaključi primer uporabe
Stanje po zaključku	Sistem je spet na začetnem zaslonu uporabniškega vmesnika.

3.1.1.6 Pregledovanje parametrov na delovnem mestu glede na izdelek

Delavec ob pregledu zgodovine izdelka želi pregledati še podrobnosti tega izdelka, ko je bil ta na izbranem delovnem mestu, ter pritisne na zapis o želenem izdelku. Tedaj se mu prikažejo vsi parametri izdelka na izbranem delovnem mestu. Tako lahko nato pregleduje parametre in iz njih pride do raznih zaključkov glede izdelka na različnih delovnih mestih.

Preglednica 5: Opis primera uporabe pregledovanja parametrov izdelka na delovnem mestu glede na izdelek

Naziv	Pregledovanje parametrov izdelka na delovnem mestu glede na izdelek
Akterji	Delavec
Zahteve	Delavec pregleduje zgodovino izdelka in želi pregledati parametre tega izdelka na poljubnem delovnem mestu.
Prožilec	Delavec pri pregledu zgodovine izdelka izbere delovno mesto s pritiskom na enega od delovnih mest, na katerih je bil izdelek. S tem želi pridobiti parametre izdelka, ki ga trenutno pregleduje, na izbranem delovnem mestu.
Osnovni potek	1. Na spodnji polovici zaslona se prikažejo vsi procesni parametri, ki so bili zabeleženi tekom obdelovanja izdelka na izbranem delovnem mestu 2. Delavec pregleduje parametre izdelka na izbranem delovnem mestu. 3. Delavec se s pritiskom na gumb »Nazaj« vrne na začetni zaslon ter zaključi primer uporabe
Stanje po zaključku	Sistem je spet na začetnem zaslonu uporabniškega vmesnika.
Alternativni poteki	2a: Delavec želi pregledati parametre izdelka na nekem drugem delovnem mestu, namesto na prej izbranem 2a1. Delavec na zaslonu izbere drugo delovno mesto 2a2. Delavec pregleduje parametre izdelka na novo izbranem delovnem mestu (zopet je na 2. koraku osnovnega poteka)

3.1.1.7 Menjava naloga

Preko tega primera uporabe akter delavec izbere nalog izdelka, ki ga trenutno obdeluje na delovnem mestu. Razpisane naloge, ki jih lahko izbere, mu s primerom uporabe *sinhronizacija podatkov s sistemom* priskrbi sistem MES, preko izbranega naloga pa se nato pridobi ustrezen recept, ki se pošlje PLC-ju naprave. Spodnji osnovni potek velja le za ZDM, kjer je ta primer uporabe izražen.

Preglednica 6: Opis primera uporabe menjave naloga

Naziv	Menjava naloga
Akterji	Delavec, Naprava
Zahteve	Delavec med preverjanjem ujemanja trenutno izbranega naloga z nalogom, pod katerim mora biti obdelan naslednji izdelek, ki se bo obdeloval, ugotovi, da se ta dva ne ujemata.
Prožilec	Na začetnem zaslonu delavec zahteva menjavo naloga s pritiskom na gumb na zaslonu.
Osnovni potek	1. Sistem na zaslonu prikaže delavcu odprte naloge iz podatkovne baze 2. Delavec izbere želen nalog 3. Sistem preveri, ali je šifra tipa izdelka zapisana v na novo izbranem nalogu enaka šifri tipa izdelka iz prejšnjega naloga 4. Sistem pri preverjanju iz 3. koraka pride do ugotovitve, da sta primerjani šifri enaki 5. Sistem zaključi s primerom uporabe
Stanje po zaključku	Sistem ima zabeležen ustrezen nalog pod katerim naj obdeluje izdelke do naslednje menjave naloga, naprava pa je ustrezno konfigurirana glede na tip izdelka, ki je zapisan v izbranem nalogu.
Alternativni poteki	2a: Delavec po ponovnem premisleku ugotovi, da mu ni potrebno izbrati novega naloga 2a1. Delavec se s pritiskom na gumb »Nazaj« vrne zopet na začetni zaslon 2a2. Sistem zaključi s primerom uporabe 4a: Sistem pri preverjanju iz 3. koraka pride do ugotovitve, da sta primerjani šifri različni

	4a1. Sistem iz podatkovne baze pridobi ustrezen recept za tip trenutnega izdelka 4a2. Sistem iz recepta prebere podatke ter jih posreduje napravi, ki jih potrebuje za pravilno delovanje 4a3. Primer se nadaljuje na koraku 5 osnovnega poteka

3.1.1.8 Opravljanje operacije na izdelku na SDM

Opravljanje operacije na izdelku na posameznem delovnem mestu je glavni način uporabe celotnega sistema. Ta primer uporabe spremlja izdelek od pričetka opravljanja operacije na delovnem mestu do zaključka tamkajšnjega obdelovanja izdelka. Večinoma gre za interakcijo med sistemom in PLC-jem naprave na trenutnem delovnem mestu, kjer je delavec več ali manj le akter, ki je o poteku primera uporabe obveščen ter sproži in zaključi operacijo.

Preglednica 7: Opis primera uporabe opravljanja operacije na izdelku na SDM

Naziv	Opravljanje operacije na izdelku na SDM
Akterji	Naprava, Delavec
Zahteve	Delavec mora dostaviti izdelek na predvideno mesto za začetek dela na napravi.
Prožilec	Naprava zazna prisotnost izdelka na predvidenem mestu za začetek dela na napravi.
Osnovni potek	1. Naprava s čitalcem kod prebere DMC kodo izdelka 2. Naprava pošlje serijsko številko izdelka (iz kode DMC) sistemu ter zahteva dovoljenje za delo na tem izdelku 3. Sistem iz lokalne podatkovne baze pridobi podatke o izdelku, katerega kodo DMC je prejel (šifro tipa izdelka, šifro zadnjega delovnega mesta na katerem je bil, šifro delovnega mesta za katerega je namenjen, šifro naloga …) 4. Sistem preveri, ali je izdelek namenjen za delo na tem delovnem mestu 5. Sistem pri preverjanju iz 4. koraka ugotovi, da je izdelek namenjen za delo na delovnem mestu, na katerem se nahaja 6. Sistem preveri, ali je šifra tipa trenutnega izdelka enaka šifri tipa izdelka, ki je bil na tem delovnem mestu pred trenutnim izdelkom

	7. Sistem pri preverjanju iz 6. koraka, pride do ugotovitve, da sta primerjani šifri enaki 8. Sistem sporoči napravi, da lahko začne z opravljanjem osrednje operacije. O tem obvesti tudi delavca z izpisom dogodka na uporabniškem vmesniku. 9. Sistem zapiše v lokalno podatkovno bazo, da je izdelek prispel na to delovno mesto in je na njem bil sprožen začetek operacije 10. Naprava začne z operacijo in po izvedeni operaciji sporoči sistemu, da je uspešno zaključila operacijo ter mu posreduje procesne parametre. 11. Sistem v lokalno podatkovno bazo zapiše prejete procesne parametre, podatek, da je izdelek uspešno zaključil operacijo na tem delovnem mestu, ter delovno mesto, na katerega naj izdelek gre naprej. 12. Sistem napravi potrdi, da je zabeležil podatke, ki jih je prejel v koraku 10 13. Sistem obvesti delavca, da je bila operacija na tem delovnem mestu uspešno zaključena z izpisom dogodka na uporabniškem vmesniku. 14. Sistem zaključi ta primer uporabe in se spet vrne v stanje pred začetkom
Stanje po zaključku	Sistem je spet v prvotnem stanju in lahko začne z operacijo na novem izdelku. Izdelek je uspešno ali neuspešno zaključil operacijo. Glede na uspešnost operacije je v podatkovni bazi ustrezno označen, na katero delovno mesto je namenjen po tej operaciji. V kolikor pride do začetka izvajanja operacije na izdelku, so vsi podatki izdelka pridobljeni ob zaključku izvajanja operacije na tem delovnem mestu zabeleženi v lokalni bazi.
Alternativni poteki	1a: Napravi ne uspe prebrati kode DMC izdelka, ki je bil postavljen na mesto za pričetek operacije 1a1. PLC naprava preko svojega uporabniškega vmesnika to sporoči delavcu 1a2. Primer uporabe se zaključi in se spet vrne v stanje pred začetkom 3a: Sistem ne pridobi nobenih podatkov o izdelku iz lokalne podatkovne baze 3a1. Sistem sporoči napravi in delavcu, da izdelek ni namenjen za to operacijo 3a2. Sistem zaključi ta primer uporabe in se spet vrne v stanje

	pred začetkom le-tega 5a: Sistem pri preverjanju iz 4. koraka ugotovi, da izdelek ni namenjen za delo na delovnem mestu, na katerem se nahaja 5a1: Sistem sporoči napravi in delavcu, da izdelek ni namenjen za to operacijo 5a2: Sistem zaključi ta primer uporabe in se spet vrne v stanje pred začetkom 7a: Sistem pri preverjanju iz 6. koraka, pride do ugotovitve, da sta primerjani šifri različni 7a1. Sistem iz podatkovne baze pridobi ustrezen recept za tip trenutnega izdelka 7a2. Sistem iz recepta prebere podatke ter jih posreduje napravi, ki jih potrebuje za pravilno delovanje 7a3. Primer se nadaljuje na koraku 8 osnovnega poteka 10a: Naprava začne z operacijo in po izvedeni operaciji sporoči sistemu, da je neuspešno zaključila operacijo na izdelku ter mu posreduje procesne parametre ter kodo in opis napake, do katere je prišlo 10a1. Sistem v lokalno podatkovno bazo zapiše prejete procesne parametre, podatek, da je izdelek neuspešno zaključil operacijo, kodo in opis napake ter delovno mesto, na katerega naj izdelek gre naprej (delovno mesto reparature) 10a2. Sistem napravi potrdi prejem podatkov, ki jih je prejel v koraku 10a 10a3. Sistem preko izpisa na uporabniškem vmesniku obvesti delavca, da je bila operacija na tem delovnem mestu neuspešno zaključena 10a4. Sistem zaključi ta primer uporabe in se spet vrne v stanje pred začetkom

3.1.1.9 Opravljanje operacije na izdelku na ZDM

Opravljanje operacije na izdelku na ZDM se od enakega primera uporabe na SDM razlikuje predvsem v tem, da izdelek ob prihodu na ZDM še nima serijske številke, temveč se tukaj le-ta določi in vgravira v izdelek. Drugače je namen tega primera uporabe enak istoimenskem primeru uporabe na SDM, in sicer spremljanje izdelka skozi izvajanje operacije na delovnem mestu.

Preglednica 8: Opis primera uporabe opravljanja operacije na izdelku na ZDM

Naziv	Opravljanje operacije na izdelku na ZDM
Akterji	Naprava, Delavec
Zahteve	Delavec mora preveriti ustreznost naloga za izdelek, ki ga bo dal v obdelavo in ga po potrebi zamenjati s pomočjo primera uporabe »Menjava naloga«. Nato mora dostaviti izdelek na predvideno mesto za začetek dela na napravi.
Prožilec	Naprava zazna prisotnost izdelka na predvidenem mestu za začetek dela na napravi.
Osnovni potek	1. S poizvedbo po podatkovni bazi se pridobi novo unikatno serijsko številko za nov izdelek na liniji 2. Sistem pošlje napravi novo generirano serijsko številko ter dovoljenje, da lahko začne z graviranjem serijske številke in opravljanjem osrednje operacije. O tem obvesti tudi delavca preko izpisa dogodka na uporabniškem vmesniku 3. Sistem zapiše v lokalno podatkovno bazo, da je izdelek prispel na to delovno mesto in je bil na njem sprožen začetek operacije 4. Naprava začne z operacijo in po izvedeni operaciji sporoči sistemu, da je uspešno zaključila operacijo ter mu posreduje procesne parametre 5. Sistem v lokalno podatkovno bazo zapiše prejete procesne parametre, podatek, da je izdelek uspešno zaključil operacijo na tem delovnem mestu, ter delovno mesto, na katerega naj gre izdelek naprej 6. Sistem napravi potrdi, da je zabeležil podatke, ki jih je prejel v koraku 4 7. Sistem z izpisom dogodka na uporabniškem vmesniku obvesti delavca, da je bila operacija na tem delovnem mestu uspešno zaključena 8. Sistem zaključi ta primer uporabe in se spet vrne v stanje pred začetkom

Stanje po zaključku	Sistem je spet v prvotnem stanju in lahko začne z operacijo na novem izdelku. Izdelek je uspešno ali neuspešno zaključil operacijo. Glede na uspešnost operacije je v podatkovni bazi ustrezno označen, na katero delovno mesto je namenjen po tej operaciji. V kolikor pride do začetka izvajanja operacije na izdelku, so vsi podatki izdelka pridobljeni ob zaključku izvajanja operacije na tem delovnem mestu zabeleženi v lokalni bazi.
Alternativni poteki	4a: Naprava začne z operacijo in po izvedeni operaciji sporoči sistemu, da je neuspešno zaključila operacijo na izdelku ter mu posreduje procesne parametre ter kodo in opis napake, do katere je prišlo 4a1. Sistem v lokalno podatkovno bazo zapiše prejete procesne parametre, podatek, da je izdelek neuspešno zaključil operacijo, kodo in opis napake ter delovno mesto, na katerega naj gre izdelek naprej (delovno mesto reparature) 4a2. Sistem napravi potrdi prejem podatkov, ki jih je prejel v koraku 4a 4a3. Sistem preko izpisa na uporabniškem vmesniku obvesti delavca, da je bila operacija na tem delovnem mestu neuspešno zaključena 4a4. Sistem zaključi ta primer uporabe in se spet vrne v stanje pred začetkom

3.2 NEFUNKCIJSKE ZAHTEVE

Ker je namen tega pilotnega projekta iskanje boljše rešitve od že obstoječe v podjetju, se moramo v veliki meri posvetiti nefunkcijskim zahtevam. Čeprav obstoječi in nov sistem nista zastavljena identično, lahko povlečemo vzporednico med obema sistemoma. Obstoječi sistem dobro opravlja svoje funkcije, ki so specificirane v funkcijskih zahtevah, kajti v nasprotnem primeru bi bil povsem neuporaben in ne bi mogel voditi proizvodne linije. Enako pričakujemo, da bo tudi nam uspelo zgraditi sistem, ki bo enako dobro zadovoljeval funkcijske zahteve. Razlike med sistemoma, po katerih bomo lahko ocenjevali uspešnost projekta, pa se bodo pojavile predvsem na področju nefunkcijskih zahtev.

Nefunkcijske zahteve so predstavljene v naslednjih podpoglavjih.

3.2.1 Prenosljivost

Pri programiranju sistema se moramo osredotočiti na to, da bo sistem kar se da modularen. Biti mora enostavno prenesti rešitev za SDM na ostala delovna mesta, ki imajo zastavljeno enako komunikacijo s SCADA sistemom. Hkrati moramo tudi razmišljati, da lahko sistem, zgrajen na tej proizvodnji liniji, s čim manj dela prenesemo tudi na druge linije, kar je še dodatna motivacija pri težnji za čim večjo prenosljivost sistema.

3.2.2 Večjezičnost

Podjetje ima tudi stranke iz drugih držav po vsem svetu, ki prihajajo na ogled proizvodnje produktov. Le-te jih odkupujejo in so zato zadovoljne, da razumejo uporabniški vmesnik na delovnih mestih. Po drugi strani pa je prav, da zagotovimo delavcem v proizvodnji uporabniški vmesnik v svojem jeziku. Tako jim olajšamo delo, se izognemo kakršnimkoli napakam zaradi slabega razumevanja ter jim omogočimo lažjo komunikacijo z vzdrževalci. Tekom pilotnega projekta se bomo trudili omogočiti uporabniški vmesnik v slovenskem in angleškem jeziku. Omogočiti pa seveda želimo tudi, da se v prihodnosti lahko po želji doda še več jezikov.

3.2.3 Odzivnost

Ena izmed najpomembnejših nefunkcijskih zahtev v našem primeru je čim krajši odzivni čas na pozive akterjev. SCADA sistem je namreč velikokrat ozko grlo sistema na liniji – PLC-ji na napravah so namreč večkrat hitrejši od SCADA sistema ter čakajo na njegov odziv. Od hitrosti sistema je odvisna dolžina proizvodnega cikla izdelka na liniji in posledično dobiček, ki ga linija lahko prinaša. V primeru poziva PLC-ja na enostaven odgovor SCADA sistema pričakujemo odgovor v času 1 sekunde, v primeru poziva na zahtevnejšo operacijo SCADA sistema pa odgovor v času pod 2 sekundama.

3.2.4 Preglednost

Delavec na liniji mora biti dobro obveščen o tem, v kakšni fazi delovanja je SCADA sistem na posameznem delovnem mestu. Tako je lahko sposoben sam diagnosticirati morebiten problem do katerega lahko pride in ga predstaviti svojemu nadrejenemu, ta pa ga lahko razrešuje oz. o tem obvesti vzdrževalce. Delavca je potrebno preko uporabniškega vmesnika čim pogosteje in podrobneje obveščati o poteku operacije ter ga obvestiti, če in do kakšnega problema je prišlo. Istočasno mora biti tudi grafični vmesnik čim bolj minimalističen in tako tudi pregleden. Delavec bo tako na začetnem zaslonu videl le podatke, ki jih nujno potrebuje.

3.2.5 Robustnost

Poleg odzivnosti je tudi ta ena pomembnejših nefunkcijskih zahtev. Vsak izpad zaradi tehnologije na liniji prinaša namreč ogromne finančne izgube. Torej se moramo prepričati, da ne bo prišlo do tega, v primeru, da pride do okvare oz. hrošča, pa mora biti zagotovljeno vzdrževanje v vseh časih, ko proizvodna linija obratuje. Vsemu navkljub je na kratki rok zanesljivost težko testirati – potrebno je testiranje tekom normalnega dneva na proizvodni liniji, kar pa je težko izvedljivo.

Istočasno mora biti sistem tudi zanesljiv v primeru izpada električne energije ali podobnih izpadov. Sposoben se mora biti čim bolj samostojno zopet postaviti, brez posebnih motenj za vzdrževalce. V primeru, da je delo za vzdrževalce v takih primerih neizogibno, pa je potrebno to delo čim bolj zmanjšati in ga ustrezno dokumentirati v navodila za postopanje v takšnih primerih. Izdelke, ki so bili morebiti v obdelavi tekom takšnega izpada v neki napravi, moramo označiti kot slabe in jih lahko na linijo ponovno pošljemo šele, ko jih pristojna oseba podrobno pregleda oz. popravi in jim dodeli delovno mesto, na katerem lahko izdelek nadaljuje s procesom izdelovanja. Evidentiranje takih izjemnih dogodkov je že implementirano v podjetje preko MES sistema in torej ni potrebno tega implementirati še v SCADA sistemu.

3.2.6 Razširljivost

Ker se proizvodna linija nahaja v lastnem podjetju, se velikokrat lahko zgodi, da prihaja do raznih dodelav proizvodne linije zaradi ugotovitev o pomanjkanju nekaterih funkcij delovnih mest. Do dodajanj funkcij lahko pride zaradi ugotovitve o možnostih, ki bi morebiti skrajšale proizvodni cikel ali omogočale dodaten nadzor nad kakovostjo. Prav zaradi dodajanj funkcij na delovnih mestih pa mora v našem SCADA sistemu obstajati tudi enostaven način za dodajanje oz. razširjanje SCADA komponente glede na nove funkcije. Torej mora biti sistem kar se da lahko razširljiv. Zaradi želene razširljivosti je potrebno tudi ustrezno evidentiranje različice SCADA sistema ter možnost povratka SCADA sistema na starejšo verzijo.

3.2.7 Razpoložljivost

Če hočemo izkoristiti novo zgrajeno proizvodno linijo v vsej svoji polnosti, je najbolje, da delavci opravljajo delo na njej v treh izmenah, torej 24 ur na dan, najmanj pet dni v tednu. Prav zato je nujno potrebno, da je tudi SCADA sistem razpoložljiv ves čas obratovanja proizvodne linije, brez tega namreč slednja ne sme obratovati, saj potem strankam podjetja ne moremo zagotoviti sledljivosti izdelkov in enako tudi ne zagotoviti dobre kvalitete.

3.2.8 Strojna oprema in omejitve podjetja

Kot je že omenjeno v študiji izvedljivosti v poglavju 2, imamo določene omejitve glede strojne opreme, ki jo lahko uporabljamo v pilotnem projektu. Zaradi pogodbe s proizvajalcem strojne opreme HP velja, da kjer to podjetje nudi svojo strojno opremo mora biti le-ta uporabljena. Torej moramo v primeru strežnika na liniji uporabiti HP-jev računalnik, enako pa velja tudi za vso periferijo tega računalnika (zaslon, tipkovnica, miška).

Zaželeno je, da ima računalnik dva diska, ki sta med sabo povezana v RAID 1 za boljšo varnost pred izgubo podatkov zaradi okvare diska. Zaradi varnosti pred izgubo podatkov ob izpadu električne energije oz. lažjem ponovnem zagonu po omenjenem dogodku, pa je potrebno tudi, da je računalnik priklopljen na UPS.

Za prikazovalnike na delovnih mestih podobno kot za strežnike velja, da v primeru če računalniki *Raspberry Pi* 3 ne bodo zadovoljili naših potreb, je zaželeno nabaviti HP-jeve majhne, industrijske računalnike, v kolikor je to smotrno. Če dovolj dobro utemeljimo dejstvo, da nabavimo računalnike kakšnega drugega proizvajalca, je seveda sprejemljiva tudi ta odločitev.

Zahteva podjetja je tudi, da ima SCADA sistem ločeno omrežje od preostanka podjetja. Omenjeno je potrebno zaradi varnostnih razlogov in čim manj motečih dejavnikov iz zunanjosti.

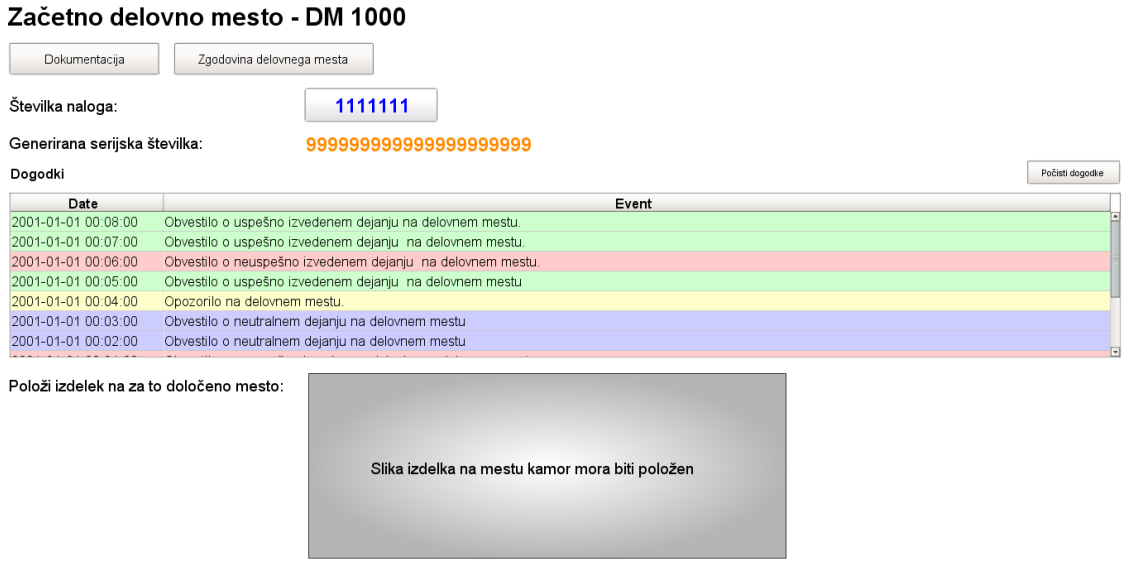
3.3 UPORABNIŠKI VMESNIK

Specifikacija zahtev uporabniškega vmesnika je, čeprav večkrat zanemarjen, tudi zelo pomemben del funkcijske specifikacije. Z dobrim uporabniškim vmesnikom namreč delavcu dodatno olajšamo delo na liniji in se izognemo morebitnim dodatnim napakam zaradi nejasnosti. Podobno velja tudi za vzdrževalce; njim lahko namreč prek uporabniškega vmesnika sporočimo dodatne informacije o stanju na delovnem mestu, kar jim omogoča lažje reševanje težav. Pri zasnovi uporabniškega vmesnika moramo slediti osnovnim smernicam načrtovanja videza uporabniškega vmesnika. Uporabniški vmesnik smo se odločili obravnavati kot posebno podpoglavje, kajti tukaj bomo navajali tako funkcijske zahteve vmesnika (kako mora delovati) kot nefunkcijske zahteve (kakšne so njegove lastnosti).

Na slikah 2 in 3 sta orisa oblike začetnih zaslonov SDM in ZDM. S tega zaslona poteka vodenje opravljanja operacije na izdelku ter pregled večine osnovnih informacij delovnega mesta in izdelka, ki se trenutno nahaja na delovnem mestu. Videza začetnih zaslonov pri SDM in ZDM sta si precej podobna. Skupne točke obeh so: naziv delovnega mesta, gumbi za vodenje po primerih uporabe delovnega mesta, serijska številka izdelka na delovnem

mestu, komponenta z dogodki ter navodila s sliko delavcu, kaj mora storiti za opravljanje operacije na delovnem mestu.

Slika 2: Oris videza uporabniškega vmesnika na splošnem delovnem mestu



Slika 3: Oris videza uporabniškega vmesnika na začetnem delovnem mestu

Naziv delovnega mesta vsebuje poimenovanje delovnega mesta, skupaj z njegovo šifro. Ta služi tako informiranju delavca in kakršnega koli drugega obiskovalca proizvodne linije kot lažji komunikaciji delavca z vzdrževalci.

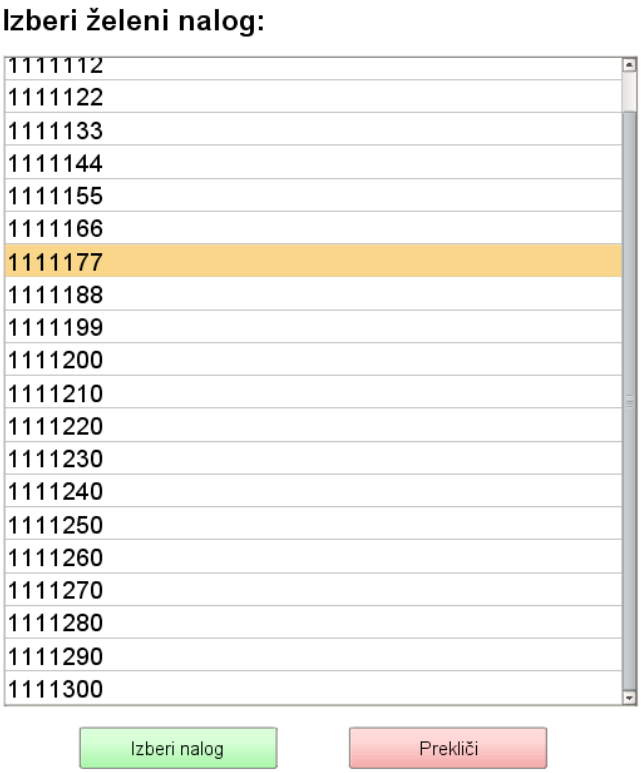
Gumbi za vodenje po primerih uporabe se razlikujejo pri SDM in ZDM. ZDM omogoča namreč le vpogled v dokumentacijo in zgodovino delovnega mesta, medtem kot SDM nudi tudi vpogled v zgodovino izdelka, ki je trenutno na delovnem mestu. Vsak od gumbov odpre nov pogled na katerem so uporabniški vmesniki, ki so opisani v nadaljevanju tega poglavja.

Na začetnem zaslonu imamo izpisano tudi serijsko številko izdelka, ki je trenutno v obdelavi na delovnem mestu. Pri SDM je to serijska številka, ki jo odčita naprava iz vgravirane DMC kode, medtem ko je pri ZDM to serijska številka, ki jo je generiral SCADA sistem. S pomočjo te številke uporabnik grafičnega vmesnika ve, kateri izdelek je v obdelavi in tudi na kateri izdelek se bo nanašala zgodovina izdelka v primeru, da bo pritisnil ustrezen gumb za uporabo te funkcionalnosti.

Tu so še dogodki, ki služijo predvsem za prikaz uporabniku, v kakšnem stanju je trenutno opravljanje operacije na izdelku ter kako je skozi čas potekalo odvijanje operacije. Uporabnik je obveščen tudi o času posameznega dogodka, uspešnosti opravljanja operacije ter serijskih številkah izdelkov, ki so opravljali operacijo na tem delovnem mestu. Dogodki so razvrščeni od najnovejšega do najstarejšega ter so za lažje razumevanje informacij barvno kodirani. Če dogodek prikazuje nekakšno dejanje, ki je bilo uspešno, vrstico obarvamo zeleno. Če prikazuje napako oz. dejanje, ki je bilo neuspešno, je vrstica rdeče obarvana, v primeru opozorila rumeno, če pa je v dogodku napisana le splošna informacija ali navodila delavcu, je vrstica modre barve. Poleg tega ima uporabnik tudi možnost dogodke počistiti s pritiskom na gumb za izbris dogodkov: »Počisti dogodke«.

Poleg vseh že omenjenih komponent uporabniškega vmesnika so tu še kratka navodila s sliko. S pomočjo slike lahko dosežemo, da delavec lažje razume svojo nalogo na delovnem mestu.

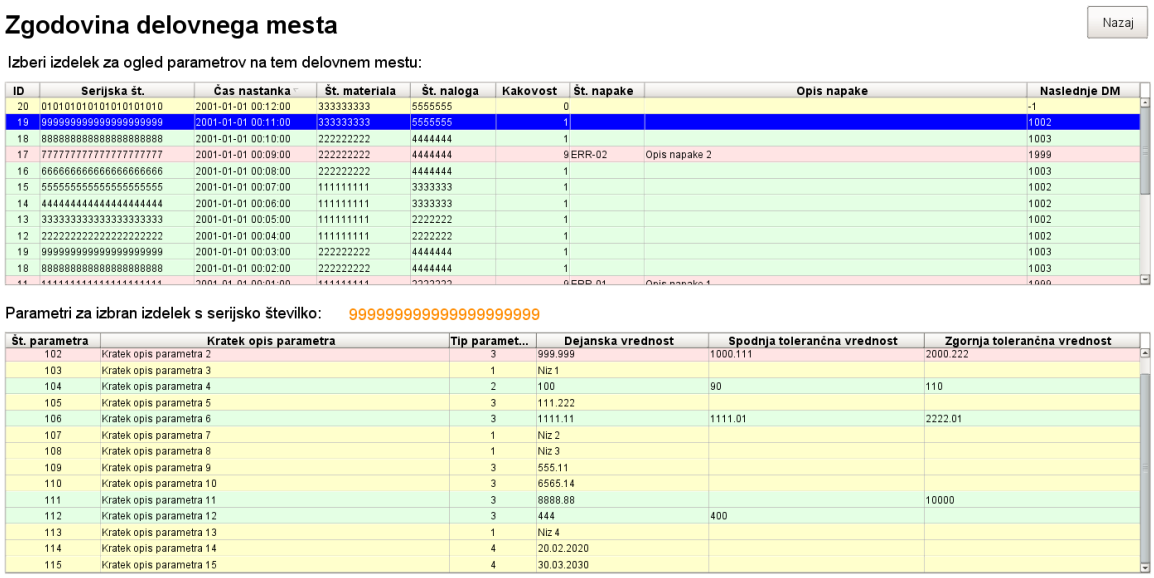
Za razliko od SDM imamo na ZDM na začetnem zaslonu napisano še šifro naloga pod katerim se dela izdelek. Na to šifro naloga lahko pritisnemo in s tem se nam odpre pojavno okno, prek katerega lahko potem izberemo nov nalog. Izgled pojavnega okna je preprosta tabela, na katero lahko pritiskamo in potrdimo izbiro naloga ali zapustimo pojavno okno. Oris videza tega grafičnega vmesnika je na sliki 4.



Slika 4: Oris videza uporabniškega vmesnika za izbiro naloga

Ogleda zgodovine izdelka in zgodovine delovnega mesta imata precej podoben uporabniški vmesnik, ki je orisan na slikah 5 in 6. Vmesnik zgodovine izdelka ima na vrhu napisano serijsko številko določenega izdelka, v zgornji tabeli pa vse podatke o delovnih mestih, na katerih je dotični izdelek že bil – šifra in opis delovnega mesta, čas ko je izdelek prispel na to delovno mesto, kakovost (uspešnost opravljanja operacije na tem delovnem mestu), šifro in opis napake do katere je morebiti prišlo ter delovno mesto, na katerega je bil poslan izdelek po opravljeni operaciji na tem delovnem mestu. S pritiskom na enega od teh opisov delovnih mest se nam spodaj odpre še ena tabela, v kateri pa so opisani procesni parametri, ki so bili zajeti na izbrani operaciji za izdelek, katerega zgodovino pregledujemo. Ta del uporabniškega vmesnika je enak kot pri zgodovini delovnega mesta. Opis posameznega parametra vključuje šifro in kratek opis parametra, tip parametra, dejansko vrednost ter spodnjo in zgornjo tolerančno vrednost parametra.

Slika 5: Oris videza uporabniškega vmesnika za ogled zgodovine izdelka in pripadajočih procesnih parametrov



Slika 6: Oris videza uporabniškega vmesnika za ogled zgodovine delovnega mesta in pripadajočih procesnih parametrov

Pri zgodovini delovnega mesta imamo v zgornji tabeli navedenih zadnjih 20 izdelkov, ki so bili na tem delovnem mestu. Navedena je njihova identifikacijska številka na tem delovnem mestu, serijska številka, čas, ko je izdelek začel z operacijo na tem delovnem mestu, nalog izdelka, številka oz. šifra materiala – torej šifra tipa izdelka, kakovost kot uspešnost opravljanja operacije na tem delovnem mestu ter šifra in opis morebitne napake. Na koncu je še navedeno delovno mesto, na katerega je bil poslan izdelek po zaključku operacije na tem delovnem mestu. S pritiskom lahko nato izberemo enega od izdelkov in spodaj se nam odpre nova tabela, kjer so na enak način kot pri zgodovini izdelka prikazani parametri za izbran izdelek.

Barva vrstice v tabelah pri grafičnem vmesniku zgodovine izdelka in zgodovine delovnega mesta je odvisna od kakovosti izdelka na delovnem mestu. Tako v zgornji tabeli teh dveh uporabniških vmesnikov obarvamo vrstico zeleno, če je operacija bila uspešno opravljena, rdeče, če je bila neuspešno, rumeno pa če se izdelek trenutno nahaja na delovnem mestu. V tabeli parametrov jo obarvamo zeleno, če je dejanska vrednost parametra znotraj tolerančnih mej, rdeče, če je izven, rumeno pa če nimamo znanih tolerančnih mej.

Dokumentacija bo vključena v sistem kot zunanja komponenta in ji zato ne bomo določali uporabniškega vmesnika. Dodali bomo samo gumb »Nazaj« v zgornjem desnem kotu za povratek na začetni zaslon.

4 NAČRTOVANJE

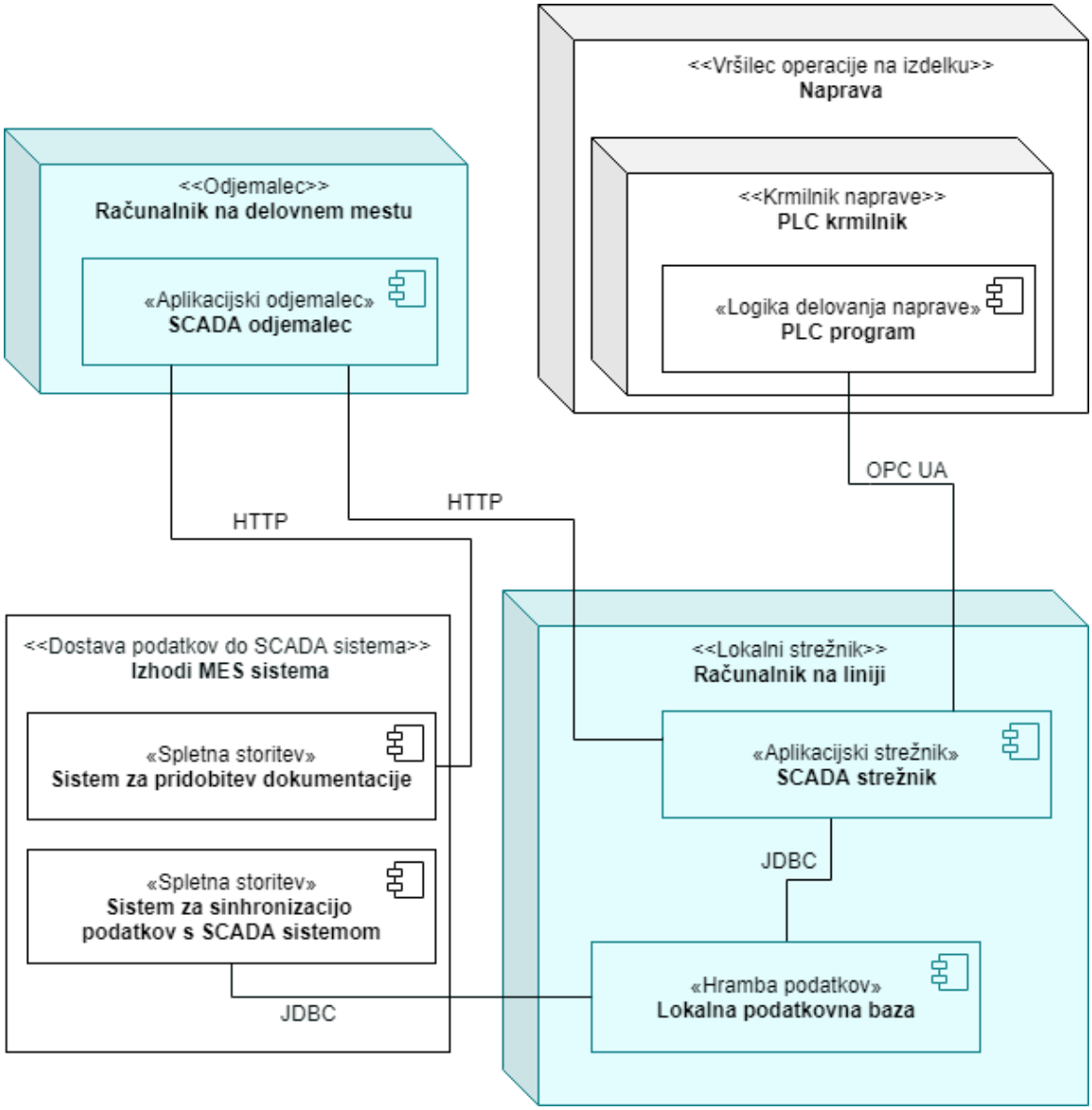
V fazi načrtovanja je naša naloga najprej določiti cilje načrtovanja [21]. Ti so nemalokrat v veliki meri povezani z nefunkcijskimi zahtevami iz specifikacije zahtev. Glede na to, da imamo kar nekaj nefunkcijskih zahtev, moramo najprej določiti, katere od njih bodo imele pomembnejšo vlogo pri načrtovalskih ciljih in katere zanemarljivejšo. Tako smo pri načrtovanju stremeli predvsem k čim večji modularnosti sistema, ki omogoča večjo robustnost in razširljivost, zanemariti pa nismo smeli niti želene odzivnosti ter zanesljivosti in razpoložljivosti. Edina nefunkcijska zahteva, ki jo lahko malce bolj zanemarimo je večjezičnost, vendar se jo vsekakor lahko potrudimo kot cilj vključiti v faze načrtovanja, saj načeloma ne bi smela ovirati drugih pomembnejših ciljev.

4.1 ARHITEKTURNO NAČRTOVANJE

Sistem najprej razdelimo na komponente in ga umestimo v okolico. Istočasno prikažemo tudi povezavo, razmerja ter vmesnike med temi komponentami. Omenjeno naredimo z diagramom postavitve, ki je viden na sliki 7. Z modro barvo so označene komponente, ki so fokus tega projekta – torej odjemalec in strežnik.

Vse tri na diagramu opisane strojne komponente (strežnik, odjemalec in naprava) se fizično nahajajo na proizvodni liniji. Tam je en strežnik in več delovnih mest; vsako od njih ima računalnik, ki služi kot odjemalec, ter napravo, ki opravlja operacijo na delovnem mestu. Na en strežnik je torej vezanih 14 delovnih mest, ki se z njim povezujejo na enak način, kot je skicirano na sliki 7. Tehnologije, ki se uporabljajo na vseh komponentah ter vmesnikih, ki so fokus tega projekta, so podrobneje opisane v poglavju 4.5.

Že v prejšnjih poglavjih smo omenjali, da imamo na vsakem delovnem mestu prikazovalnik. Preko HDMI in USB kabla je zaslon na dotik povezan z mini računalnikom na delovnem mestu. Prikazovalnik je torej na voljo delavcu za informacije o delovnem mestu in morebitno upravljanje le-tega preko odjemalca. Odjemalec preko HTTP protokola komunicira s sistemom za pridobitev dokumentacije, ki je del MES sistema in se nahaja na nekem strežniku ter s strežnikom SCADA, kjer se odvija vsa logika SCADA sistema. Ta mrežni prehod se nahaja na glavnem strežniku (znamke HP), na katerem je nameščen operacijski sistem *Windows 10*. Poleg tega je na istem računalniku nameščena tudi podatkovna baza. SCADA strežnik izvaja poizvedbe v lokalni podatkovni bazi preko JDBC javanskega API-ja, na isti način pa tudi MES sistem dostavlja vse potrebne podatke do SCADA sistema. Lokalna podatkovna baza in umestitev te baze v sistem baz je podrobneje opisana v naslednjem poglavju (4.2), opis JDBC-ja pa bomo izpustili, saj gre prav tako kot pri HTTP za dobro znan protokol.



Slika 7: Diagram postavitve sistema za eno delovno mesto – modra barva prikazuje komponente, ki so fokus te magistrske naloge

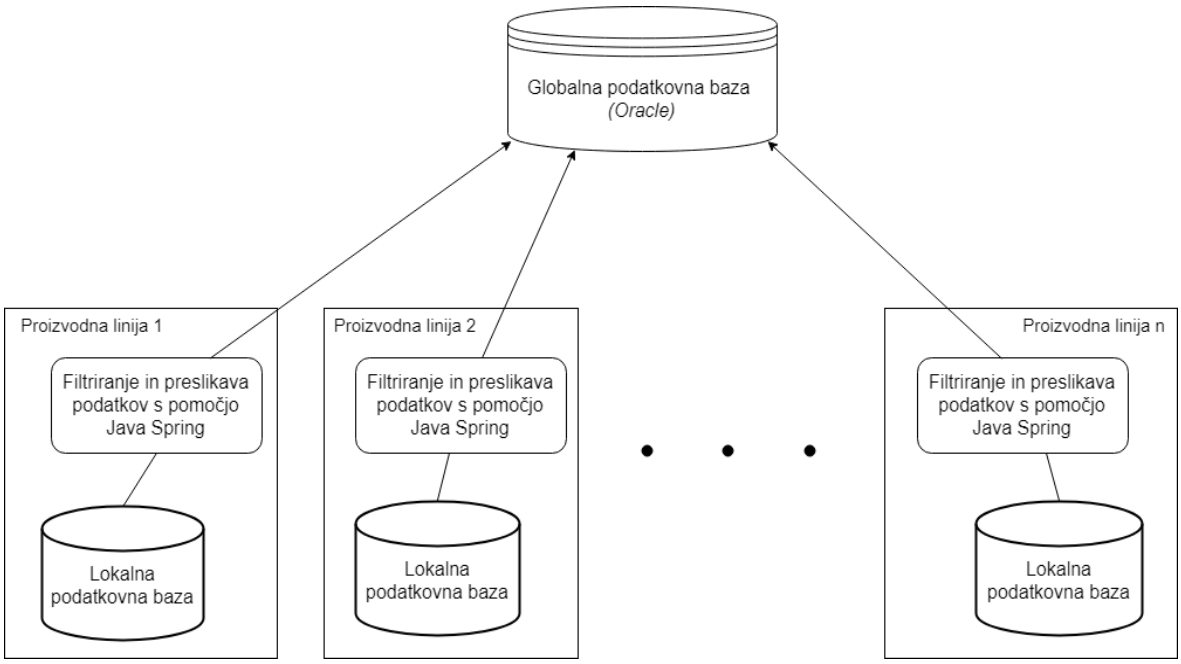
Aplikacijska strežnik in odjemalec sta oba aplicirana s pomočjo orodja *Ignition*, ki je podrobneje opisano v poglavju 4.5.1. Vemo, da med seboj komunicirata preko HTTP protokola, vendar nam je točna narava vmesnika med odjemalcem in strežnikom neznana, kajti z vmesnikom ne upravljamo mi sami, temveč ga v ozadju upravlja orodje *Ignition*. Tako lahko torej pri programiranju sistema obe omenjeni komponenti smatramo kot eno, saj se s funkcijami, ki se kličejo med strežniško in odjemalsko aplikacijo, ne ukvarjamo. Še vedno pa vemo, da se celotna logika, ki upravlja delovno mesto, odvija na strežniku, medtem ko odjemalec le s klicem funkcij strežnika temu posreduje različne parametre za ustrezno delovanje za dotično delovno mesto. Poleg tega ima odjemalec tudi samostojno funkcijo (nepovezano s strežnikom) dostopanja do dokumentacije preko URL-ja v *Web browser* komponenti v orodju *Ignition*.

Naprava in s tem njen programabilni logični krmilnik (PLC) sta v domeni elektroinženirjev, ki so programirali delovanje naprave. Z delovanjem naprave se torej ne ukvarjamo mi, temveč s Siemensovimi PLC-ji le komuniciramo preko aplikacijskega strežnika. Komunikacija poteka preko vmesnika s protokolom OPC UA, kar nam omogoča pridobivanje in posredovanje informacij napravi za delovanje. Podroben opis te tehnologije lahko najdemo v poglavju 4.5.2. Tudi kot OPC strežnik nam služi orodje *Ignition*, saj ima ta modul že vgrajen in ga moramo le ustrezno konfigurirati. Preko tega modula lahko nato preko protokola OPC UA dostopamo do posameznih podatkovnih blokov v PLC-ju in s pomočjo znanja o natančni lokaciji spremenljivk v podatkovnem bloku (te podatke nam posredujejo programerji PLC naprav) tudi do dotičnih značk, ki jih potrebujemo za komuniciranje z napravo.

4.2 PODATKOVNA BAZA

Iz specifikacije zahtev izhaja potreba, da se mora SCADA sistem dobro inkorporirati v že obstoječi sistem v podjetju. Torej je zaradi praktičnosti in univerzalnosti dobro uporabiti tudi sistem baz, ki je že v uporabi na drugih linijah v podjetju.

Imamo globalno in lokalno podatkovno bazo. Na sliki 8 lahko vidimo, kako deluje sistem baz. SCADA sistem poizveduje po lokalni bazi, medtem ko MES sistem uporablja globalno bazo za analizo podatkov iz vseh proizvodnih linij, ki so povezane v tak sistem.

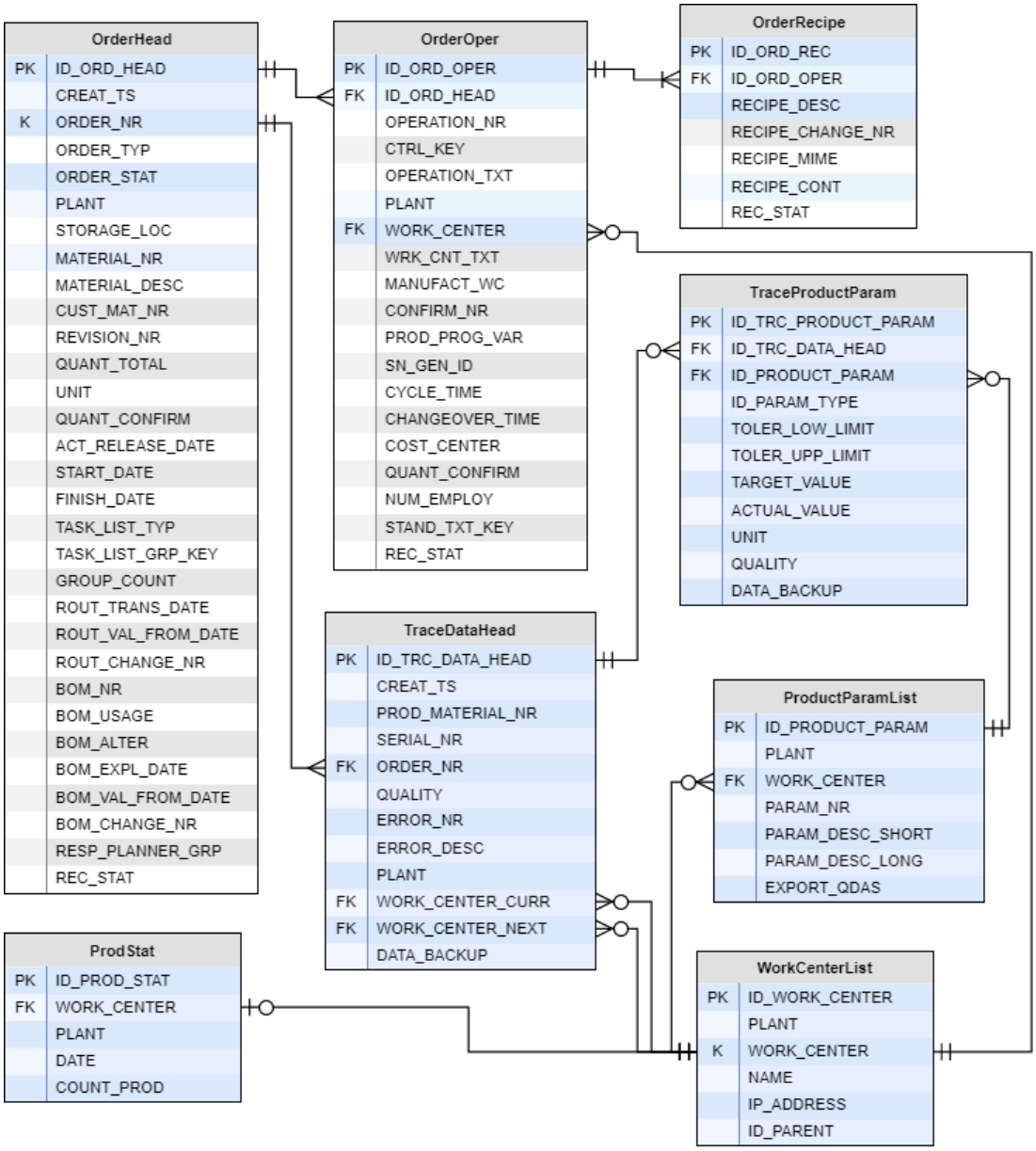


Slika 8: Organizacija in kopiranje med podatkovnimi bazami v podjetju

Vse linije v podjetju imajo lokalne baze s podobno strukturo, kar potem omogoča karseda enostavno kopiranje v globalno podatkovno bazo. Tako je torej nemudoma jasno, da strukture globalne baze seveda ne moremo spreminjati, ker je ta vključena v sistem v celotnem podjetju.

Iz tega seveda sledi, da je smiselno uporabiti tudi karseda enako strukturo lokalne podatkovne baze kot na drugih linijah. Tako uporabimo strukturo, ki je preverjena, istočasno pa lahko uporabimo enak sistem kopiranja med podatkovnima bazama (s pomočjo javanskega ogrodja *Java Spring*).

Na sliki 9 lahko vidimo predvideno strukturo lokalne podatkovne baze prikazano s pomočjo relacijskega diagrama.



Slika 9: Relacijski diagram lokalne podatkovne baze – odtenki modre barve prikazujejo atribute, ki so pomembni za razumevanje in izvedbo projekta

V nadaljevanju so opisane posamezne tabele. Podrobneje so opisane tiste, katerih struktura je bolj pomembna za razumevanje in izvedbo projekta. Nekatere tabele ohranijo strukturo iz globalne podatkovne baze in v tem projektu ne potrebujemo vseh njihovih atributov; atributi, ki so za nas pomembni so na diagramu označeni z odtenki modre barve.

Nekatere atribute lahko opišemo vnaprej, saj imajo v vseh tabelah enak oz. podoben pomen. Tako je najprej vredno omeniti, da je v vseh tabelah primarni ključ neko celo število, ki sicer nima nobene druge funkcije, razen funkcije primarnega ključa. Podobno ima več tabel atribut *DATA_BACKUP*, ki nam pove, ali je že bil določen zapis v tabeli kopiran v globalno podatkovno bazo. Dogovorjene vrednosti za ta atribut so:

- ***0***, če ta vnos šele čaka, da bo bil prekopiran v globalno podatkovno bazo (ni še bil kopiran)
- ***1***, če ta vnos ni več namenjen za kopiranje v globalno podatkovno bazo (je že bil kopiran)

Še en od takih atributov je *CREAT_TS,* v katerega se zapiše natančen datum in čas, ko je bil zapis v tabeli ustvarjen, ostali atributi pa so več ali manj unikatni in so zato opisani pod opisi posameznih tabel.

4.2.1 TraceDataHead

Osrednja tabela, ki je namenjena sledenju posameznega izdelka po posameznih delovnih mestih. Vsaka vrstica v tej tabeli predstavlja izdelek na posameznem delovnem mestu. Opis pomembnejših atributov je v preglednici 9.

Preglednica 9: Opis atributov v tabeli TraceDataHead

Ime atributa	**Tip atributa**	**Opis atributa**
ID_TRC_DATA_HEAD	bigint	*Primarni ključ*
CREAT_TS	datetime	*Atribut opisan zgoraj*
PROD_MATERIAL_NR	varchar	Šifra za določen tip izdelka – vsak tip izdelka ima svojo šifro
SERIAL_NR	varchar	Serijska številka izdelka – to je številka, ki je unikatna za vsak posamezni izdelek
ORDER_NR	varchar	Tuj ključ - šifra naloga, pod katerim je delan izdelek na delovnem mestu, ki je napisano pod

		atributom *WORK_CENTER_CURR*
QUALITY	int	Kakovost izdelka na delovnem mestu, ki je napisano pod atributom *WORK_CENTER_CURR*. Dogovorjene vrednosti za ta atribut so: • ***0***, če je izdelek trenutno v obdelavi na tem delovnem mestu • ***1***, če se je operacija na izdelku na tem delovnem mestu uspešno zaključila • ***9***, če se operacija na izdelku na tem delovnem mestu ni uspešno zaključila
ERROR_NR	varchar	V primeru, da je pri obdelavi izdelka na delovnem mestu, ki je napisano pod atributom WORK_CENTER_CURR prišlo do napake, potem se v ta stolpec vpiše koda napake, ki jo pridobimo od PLC-ja na liniji
ERROR_DESC	varchar	Podrobnejši opis napake, katere koda je zapisana pod atributom *ERROR_NR*
PLANT	varchar	Šifra obrata, na katerem se izdelek nahaja – ta je vedno enak za določen izdelek
WORK_CENTER_CURR	varchar	Ko pride izdelek na delovno mesto, se ustvari zapis v tabeli s šifro tega delovnega mesta iz ERP-ja. Ta šifra je tuj ključ, ki se nanaša na vnos v tabeli *WorkCenterList* in je zapisana v temu atributu.
WORK_CENTER_NEXT	varchar	Tuji ključ nanašajoč na vnos v tabeli *WorkCenterList* - š*ifra* delovnega mesta iz ERP-ja, na katerega naj izdelek nadaljuje po opravljeni obdelavi izdelka na trenutnem delovnem mestu. Vrednosti tega atributa so odvisne od rezultata operacija na trenutnem delovnem mestu: • V primeru, da se operacija na trenutnem delovnem mestu šele odvija, postavimo ta atribut na ***-1*** • V primeru, da se operacija na trenutnem delovnem mestu ustrezno zaključi,

		dodelimo temu atributu šifro delovnega mesta, na katerem se nadaljuje začrtana obdelava izdelka • V primeru, da pri obdelavi izdelka na trenutnem delovnem mestu pride do napake, dodelimo temu atributu število delovnega mesta za reparaturo
DATA_BACKUP	varchar	*Atribut opisan zgoraj.*

4.2.2 ProductParamList

Seznam vseh parametrov, ki so uporabljeni na določeni liniji. To so parametri, ki jih dobimo z različnimi meritvami od naprave na posameznem delovnem mestu oz. lahko tudi parametri, ki jih SCADA sistem zapiše v bazo. Vse te parametre lahko nadalje MES sistem analizira oz. jih uporabimo za nadzor nad kakovostjo. Opis tabele je v preglednici 10.

Preglednica 10: Opis atributov v tabeli ProductParamList

Ime atributa	**Tip atributa**	**Opis atributa**
ID_PRODUCT_PARAM	bigint	*Primarni ključ*
PLANT	varchar	Šifra obrata, na katerega se nanaša parameter
WORK_CENTER	varchar	Šifra delovnega mesta, na katerega se nanaša parameter. Ta šifra je tuj ključ, ki se nanaša na vnos v tabeli *WorkCenterList*.
PARAM_NR	int	Šifra parametra – vsak parameter ima svojo unikatno šifro
PARAM_DESC_SHORT	varchar	Kratek opis parametra
PARAM_DESC_LONG	varchar	Daljši opis parametra
EXPORT_QDAS	varchar	Temu atributu dodelimo vrednost ***1*** v primeru, da se ta parameter izvaža v orodje za nadzor nad kakovostjo (to se dogaja na nivoju globalne podatkovne baze)

4.2.3 TraceProductParam

Dotične vrednosti parametrov, katerih seznam je zapisan v tabeli *ProductParamList*. Za vsako odčitano vrednost imamo v tej tabeli en vnos, ki je preko tujega ključa povezan z enim vnosom v *ProductParamList*. Stolpci v tej tabeli so opisani v preglednici 11.

Preglednica 11: Opis atributov v tabeli TraceProductParam

Ime atributa	**Tip atributa**	**Opis atributa**
ID_TRC_PRODUCT_PARAM	bigint	*Primarni ključ*
ID_TRC_DATA_HEAD	bigint	Tuj ključ, ki se nanaša na vnos v tabeli *TraceDataHead* – pove nam, na katero operacijo obdelave katerega izdelka se nanaša parameter v vrstici tabele *TraceProductParam*
ID_PRODUCT_PARAM	bigint	Tuj ključ, ki se nanaša na vnos v tabeli *ProductParamList* – pove nam, na kateri parameter se nanaša vrednost, zapisana v vrstici
ID_PARAM_TYPE	int	Podatkovni tip parametra, ki ga označimo s celim številom od 1 do 5: • ***1*** označuje niz • ***2*** označuje celo število • ***3*** označuje decimalno število • ***4*** označuje dateTime (datum in čas) tip • ***5*** označuje boolean tip
TOLER_LOW_LIMIT	float	Spodnja mejna vrednost parametra
TOLER_UPP_LIMIT	float	Zgornja mejna vrednost parametra
TARGET_VALUE	float	Idealna vrednost parametra
ACTUAL_VALUE	nvarchar	Dejanska vrednost parametra
UNIT	nvarchar	Merska enota parametra

QUALITY	int	Kakovost parametra iz vrstice. Dogovorjene vrednosti so: • ***1***, v primeru, da je parameter znotraj tolerančnega območja • ***0***, v primeru, da je parameter zunaj tolerančnega območja
DATA_BACKUP	varchar	*Atribut opisan zgoraj.*

4.2.4 WorkCenterList

Seznam vseh delovnih mest na liniji. Ta tabela ni povezana z nobeno drugo in služi pregledu nad vsemi delovnimi mesti ter shranjevanju nekaterih podatkov o delovnih mestih. Podrobnejši opis atributov tabele sledi v preglednici 12.

Preglednica 12: Opis atributov v tabeli WorkCenterList

Ime atributa	Tip atributa	Opis atributa
ID_WORK_CENTER	bigint	*Primarni ključ*
PLANT	varchar	Obrat, na katerem se nahaja delovno mesto
WORK_CENTER	varchar	Ključ z unikatno šifro delovnega mesta iz ERP-ja
NAME	varchar	Naziv delovnega mesta
IP_ADDRESS	varchar	IP naslov morebitnega prikazovalnika na delovnem mestu
ID_PARENT	bigint	IP delovnega mesta starša opisanega delovnega mesta (če ga le-to ima)

4.2.5 OrderHead

V tej tabeli so opisani posamezni nalogi preneseni iz ERP-ja. Struktura tabele, ki je vidna v preglednici 13, se ohrani iz ERP-ja skupaj z vsemi atributi, zato smo opis atributov, ki niso uporabljeni v tem projektu, izpustili.

Preglednica 13: Opis atributov v tabeli OrderHead

Ime atributa	**Tip atributa**	**Opis atributa**
ID_ORD_HEAD	int	*Primarni ključ*
CREAT_TS	datetime	*Atribut opisan zgoraj*
ORDER_NR	varchar	Ključ z unikatno šifro naloga (iz ERP-ja). To šifro se potem uporablja tudi v drugih tabelah.
ORDER_STAT	varchar	Tukaj so napisane različne oznake, ki nekaj povedo o nalogu. Za nas je pomembna oznaka ***LANS***, ki označuje, da je nalog odprt in da se pod tem nalogom lahko izdeluje izdelke.
PLANT	varchar	Šifra obrata, za katerega je razpisan nalog
MATERIAL_NR	varchar	Šifra tipa izdelka, za katerega je razpisan nalog

4.2.6 OrderOper

V tabeli *OrderOper* so zapisane informacije naloga za določeno operacijo. Tudi tukaj velja enako kot pri *OrderHead* – v preglednici 14 smo opisali le atribute, ki so uporabljeni v našem projektu (označeni z modro barvo na relacijskem diagramu). V bistvu v našem projektu te tabele sploh ne bi potrebovali, če bi tabeli *OrderRecipe* dodali še atribut *WORK_CENTER*, vendar to tabelo potrebuje sistem MES in je preko nje tabela *OrderRecipe* vezana na *OrderHead*.

Preglednica 14 : Opis atributov v tabeli OrderOper

Ime atributa	**Tip atributa**	**Opis atributa**
ID_ORD_OPER	int	*Primarni ključ*
ID_ORD_HEAD	int	Tuj ključ iz tabele *OrderHead*. Tukaj je zapisano, na kateri nalog se nanašajo informacije iz dotične vrstice.
PLANT	varchar	Šifra obrata, na katerega se nanaša vrstica iz *OrderOper*
WORK_CENTER	varchar	Tuj ključ, ki se nanaša na vnos v tabeli *WorkCenterList*. Pove nam šifro delovnega mesta, na katerega se nanaša vrstica iz *OrderOper.*

4.2.7 OrderRecipe

Vsak vnos v tabelo *OrderRecipe* je recept za določen nalog na posamezni operaciji. Natančnejši opis te tabele je viden v preglednici 15.

Preglednica 15: Opis atributov v tabeli OrderRecipe

Ime atributa	Tip atributa	Opis atributa
ID_ORD_REC	int	*Primarni ključ*
ID_ORD_OPER	int	Tuj ključ iz tabele *OrderOper*. Očitno je, da nam torej poda informacijo, pod katerim nalogom in na kateri operaciji se uporablja ta recept.
RECIPE_DESC	varchar	Opis recepta
RECIPE_CONT	varbinary	Vsebina recepta v JSON obliki. V vsebini so navodila PLC-jem za delo za določeno operacijo.

4.2.8 ProdStat

Ta tabela je namenjena štetju kolikim izdelkom v določenem dnevu je bila do sedanjega časa vgravirana unikatna DMC koda. V našem primeru je to število izdelkov, za katere se je v določenem dnevu začel proces proizvajanja na proizvodni liniji (graviranje kode v izdelek se zgodi na začetku proizvodne linije). Opis atributov je viden v preglednici 16.

Preglednica 16: Opis pomembnejših atributov v tabeli ProdStat

Ime atributa	Tip atributa	Opis atributa
ID_PROD_STAT	int	*Primarni ključ*
PLANT	varchar	Šifra obrata, na katerem so izdelki, katerih dnevno količino se šteje
WORK_CENTER	varchar	Šifra delovnega mesta, kjer se to tabelo uporablja (kjer se gravira). Šifra je hkrati tudi tuj ključ, ki se nanaša na atribut WORK_CENTER v tabeli *WorkCenterList*.
DATE	date	Zapis datuma dneva, na katerega je bilo narejeno število izdelkov, ki je zapisano v atributu *COUNT_PROD*
COUNT_PROD	int	Števec izdelkov, ki so do sedanjega trenutka začeli s proizvodnjo na proizvodnji liniji v dnevu, ki je zapisan v atributu *DATE*

4.3 OBNAŠANJE SISTEMA

Pri načrtovanju obnašanja sistema smo si pomagali z diagrami UML. S pomočjo le-teh lahko namreč karseda pregledno in nedvoumno opišemo delovanje sistema. V poglavju 4.1 smo arhitekturo sistema opredelili s pomočjo diagrama postavitve, v tem poglavju pa bomo prikazali obnašanje sistema z dvema diagramoma aktivnosti in dvema sekvenčnima diagramoma. Diagrami, ki ponazarjajo obnašanje sistema, nastopajo v dvojicah, saj vsak od njih prikazuje glavni primer uporabe sistema na vsakem od dveh tipov delovnih mest (torej na SDM in ZDM). Glavni primer uporabe sistema je seveda opravljanje operacije na

izdelku, temu pa smo pri diagramskih tehnikah priključili v primeru ZDM tudi primer uporabe menjave naloga.

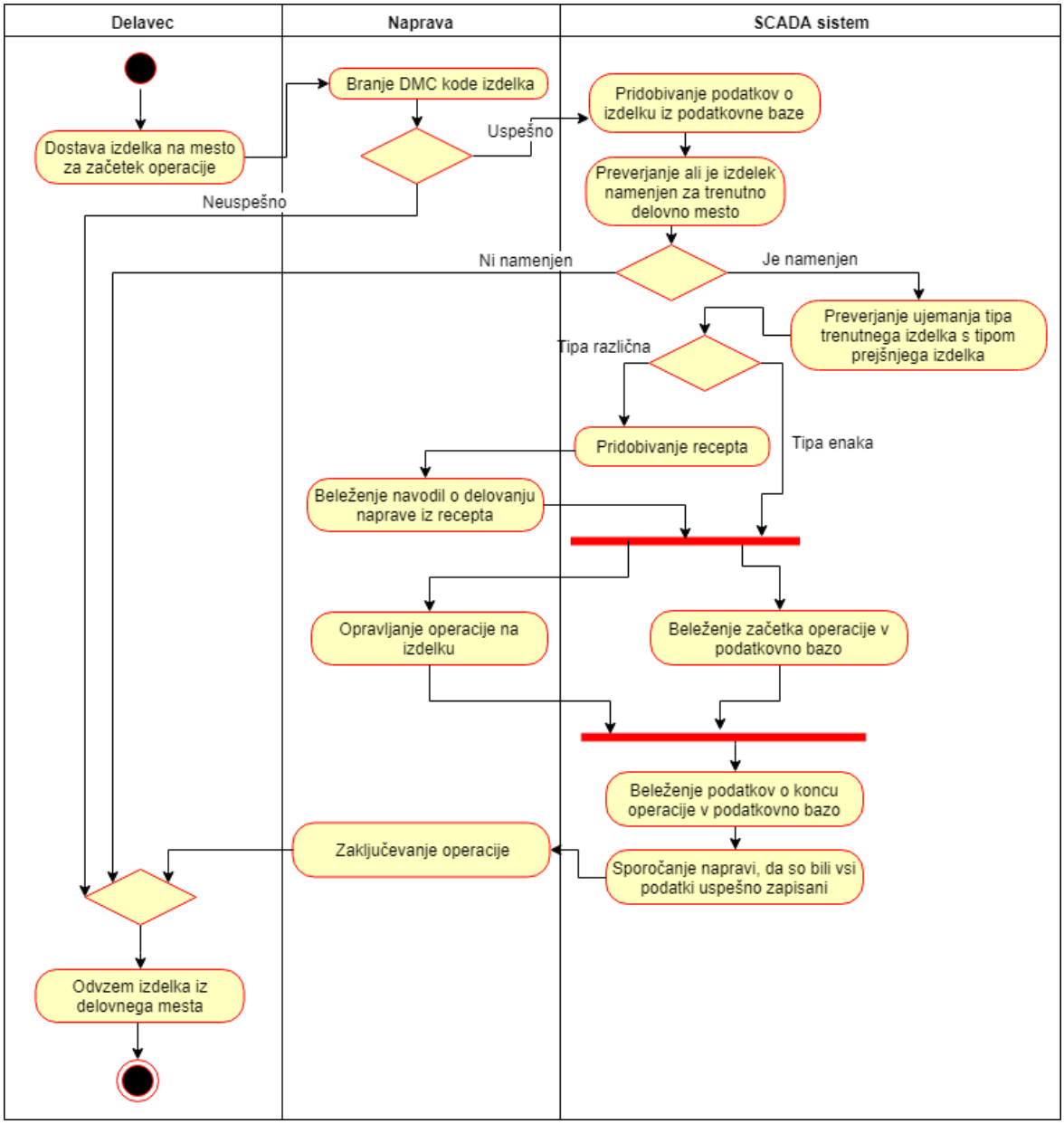
4.3.1 Predstavitev kontrolnega toka pri obratovanju delovnega mesta

Iz praktičnih razlogov smo naredili dva diagrama aktivnosti. Na sliki 10 vidimo diagram aktivnosti za primer uporabe opravljanje operacije na SDM, na sliki 11 pa je enak diagram, le da za ZDM in skupaj z menjavo naloga.

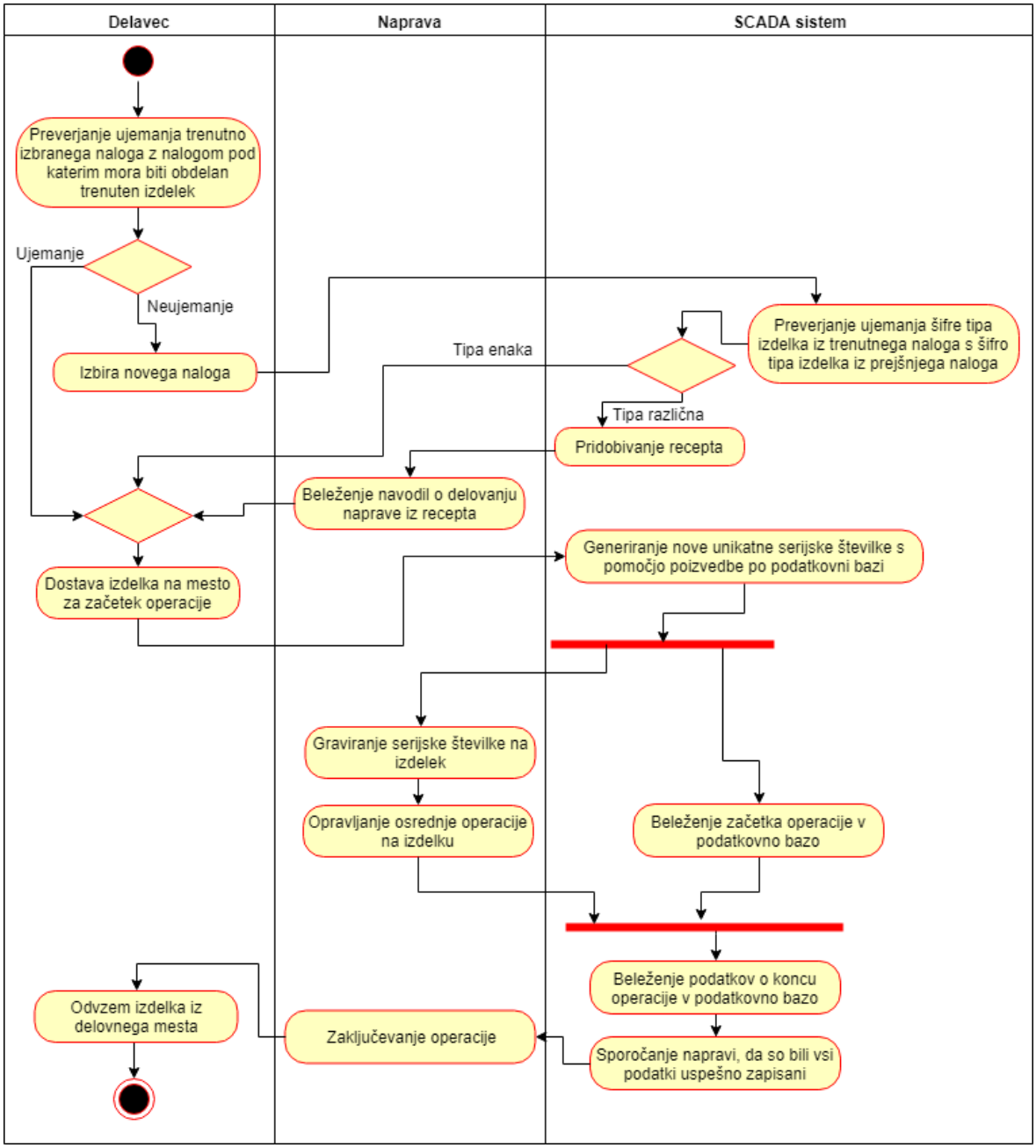
Na diagramih je razvidno, kako mora na ZDM delavec najprej preveriti ustreznost naloga, ter ga po potrebi spremeniti. V kolikor ga spremeni, mora sistem takrat tudi preveriti, ali je šifra tipa trenutnega izdelka različna od šifre tipa prejšnjega izdelka. V kolikor to dejstvo drži, je potrebno pridobiti nov recept in navodila iz recepta poslati napravi. Šele nato delavec dostavi izdelek na delovno mesto, kar je začetna aktivnost na diagramu aktivnosti SDM. V SDM se po dostavi izdelka prebere DMC koda izdelka ter preveri ustreznost izdelka za nadaljevanje operacije, šele nato pa se upravlja z recepti preko tipa skeniranega izdelka po enakem protokolu kot se upravlja na ZDM. Od tu naprej so aktivnosti do zaključka primera uporabe (odvzema izdelka iz delovnega mesta) več ali manj enake tako pri SDM kot tudi pri ZDM. Razlika je le v tem, da mora pred začetkom dejanske operacije na izdelku in beleženjem tega dogodka v podatkovni bazi, sistem na ZDM generirati novo unikatno serijsko številko s pomočjo poizvedbe po podatkovni bazi. To serijsko številko tudi tekom opravljanja operacije na izdelku na ZDM sistem vgravira v izdelek. Pri obeh tipih delovnih mest po opravljeni operaciji sistem zabeleži še podatke o koncu operacije v podatkovno bazo ter sporoči napravi, da je to opravil, ta pa nato zaključi operacijo in preda vajeti delavcu, ki izdelek odvzame iz delovnega mesta.

Zgoraj je opisan osnovni potek primera uporabe prikazan z diagramom aktivnosti. Predvsem pri SDM imamo tudi nekatere poteke, ki hitro zaključijo s primerom uporabe. To se zgodi v primeru, ko je branje DMC kode neuspešno (zaradi slabe kvalitete kode) ali ko sistem ugotovi, da izdelek ni namenjen za delovno mesto, na katerega je bil dostavljen.

Med analizo diagramov aktivnosti lahko tudi pridemo do še globljega spoznanja, da se veliko aktivnosti med SDM in ZDM prekriva, kar torej pomeni, da bomo lahko tekom izvedbe sistema veliko aktivnosti implementirali na zelo podoben način in bomo le modularno dodajali funkcije, ki bodo izvajale aktivnosti, ki so med SDM in ZDM različne.



Slika 10: Diagram aktivnosti za primer uporabe opravljanje operacije na SDM



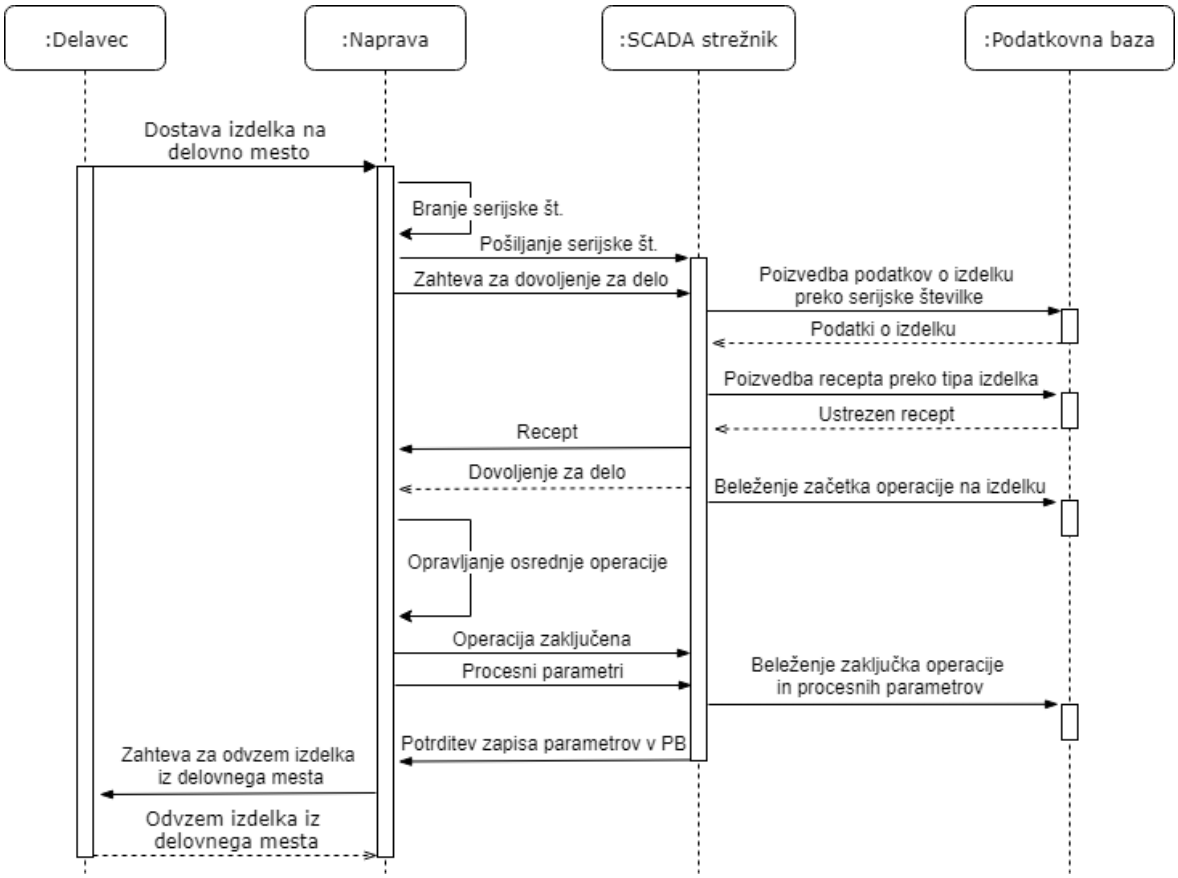
Slika 11: Diagram aktivnosti za primer uporabe opravljanje operacije na ZDM skupaj z menjavo naloga

4.3.2 Izmenjava sporočil med komponentami v časovnem zaporedju

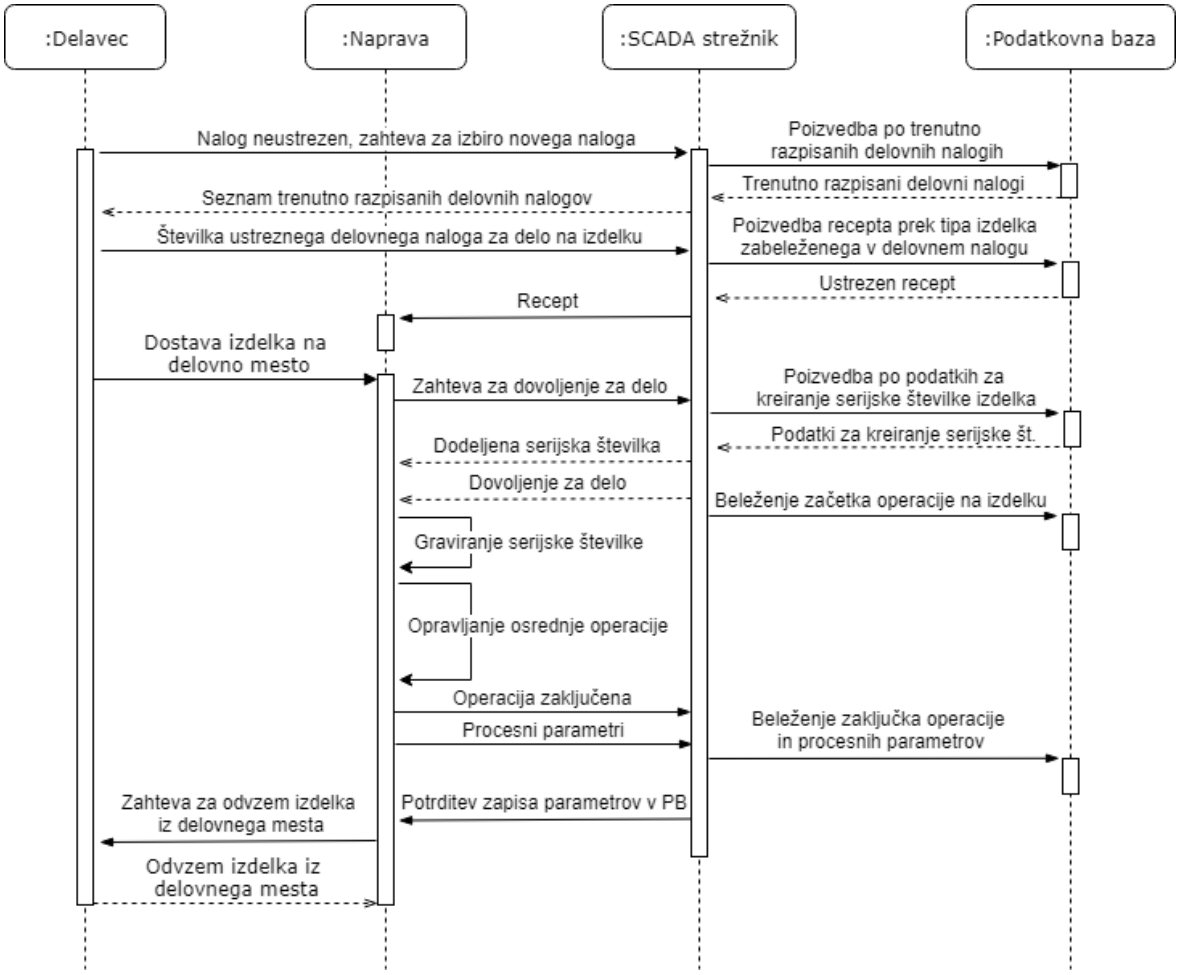
Enako kot pri predstavitvi kontrolnega toka pri obratovanju delovnega mesta imamo tudi tukaj dva diagrama. Sekvenčni diagram za primer uporabe opravljanje operacije na SDM je viden na sliki 12, ekvivalentni sekvenčni diagrama na ZDM skupaj z menjavo naloga pa lahko analiziramo na sliki 13. Na diagramih smo pregledali le osnovni potek primera uporabe. Izjema je le pri koraku 7 opravljanja operacije na SDM, ko smo uporabili le alternativo 7a in pri koraku 4 menjave naloga, ko smo raje prikazali alternativo 4a namesto osnovnih potekov. Te dve alternativi smo prikazali zato, da razjasnimo, kako poteka

upravljanje z recepti na delovnih mestih. Ostalih alternativ pa nismo prikazovali, saj smo hoteli olajšati razumevanje diagramov; težko razumljivi diagrami bi namreč razvrednotili pomen risanja diagramov. Ostalih potekov nismo prikazali tudi zato, ker le-ti zahtevajo bistveno manj izmenjevanja sporočil med komponentami – torej zgolj izpustijo nekatere izmenjave in smo iz tega razloga prikazali najbolj kompleksen potek glede izmenjave sporočil. Celoten osnovni potek in vse alternativne poteke pa lahko vedno vidimo v opisu primera uporabe v poglavju 3.1.1.

Na sekvenčnih diagramih je viden potek skozi čas in interakcije med objekti. Objekti so delavec, naprava s svojim PLC-jem ter SCADA sistem, ki je razdeljen na dva dela, in sicer na SCADA strežnik ter podatkovno bazo. Delitev je narejena z namenom, da še bolje razjasnimo interakcijo tudi med SCADA strežnikom in podatkovno bazo in ne le med sistemom in vhodi ter izhodi v sistemu. Vsa interakcija delavca s SCADA strežnikom seveda poteka preko SCADA odjemalca na delovnem mestu, vendar odjemalec le direktno prenese sporočilo SCADA strežniku, zato te komunikacije nismo vrisovali v diagram, kajti to bi zmanjšalo preglednost.



Slika 12: Sekvenčni diagram za primer uporabe opravljanje operacije na SDM



Slika 13: Sekvenčni diagram za primer uporabe opravljanje operacije na ZDM skupaj z menjavo naloga

Podobno kot pri diagramu aktivnosti lahko tudi v tej diagramski tehniki vidimo skupne korake med SDM in ZDM in tiste, ki so različni. Od dodelitve dovoljenja za delo s strani SCADA strežnika dalje sta diagrama skladna s to razliko, da naprava na ZDM še gravira serijsko številko preden opravi osrednjo operacijo, vendar to ni bistvenega pomena in ne vpliva na interakcijo med komponentami.

Začetek diagramov pa je precej različen in ima nekatere skupne operacije, ki se ne zgodijo v enakem vrstnem redu. Diagram ZDM se začne s preverjanjem ustreznosti naloga delavca. V kolikor ta ugotovi, da je nalog neustrezen, zahteva izbiro novega naloga, SCADA strežnik pa mu prikaže vse trenutno razpisane delovne naloge, ki jih pridobi preko poizvedbe v podatkovni bazi. Delavec nato izbere ustrezen nalog, ki ga SCADA strežnik shrani kot aktiven nalog. Prek tipa izdelka, ki je zapisan v tem delovnem nalogu (v primeru, da je ta tip izdelka drugačen od tipa izdelka, ki je bil prej v obdelavi na tem delovnem mestu) strežnik naredi poizvedbo po bazi, da pridobi ustrezen recept, ki ga potem posreduje napravi, da se le-ta nastavi, da bo ustrezno obdelovala naslednje izdelke. Zatem delavec dostavi izdelek na delovno mesto. Naprava pošlje zahtevo za dovoljenje za delo, ki ji ga SCADA strežnik dodeli po tem ko s pomočjo poizvedbe po podatkovni bazi

ustvari novo unikatno serijsko številko, ki bo vgravirana v nov izdelek in jo posreduje napravi.

Pri SDM se vse skupaj začne z dostavo izdelka na delovno mesto. Naprava nato z bralcem kod DMC prebere serijsko številko in jo posreduje SCADA strežniku ter zahteva dovoljenje za delo na izdelku s to serijsko številko. Strežnik SCADA preveri, ali lahko dovoli delo na izdelku, in sicer tako, da naredi poizvedbo po podatkovni bazi preko serijske številke, ki jo je prejel. Ko mu podatkovna baza odgovori s podatki o tipu izdelka ter delovnem mestu, na katerega je izdelek s to serijsko številko namenjen, strežnik SCADA opravi upravljanje z recepti po istem postopku, kot je to narejeno na ZDM po izbiri ustreznega naloga. Šele po tem se pošlje napravi dovoljenje za delo na izdelku.

4.4 NAČRTOVANA STROJNA OPREMA

V poglavju 3.2 smo že omenili zahteve za nakup strojne opreme. Pri načrtovanju se seveda skušamo teh načel tudi držati. Na sliki diagrama postavitve (slika 7) v poglavju 4.1 lahko vidimo dve komponenti, na kateri se osredotočamo tekom tega projekta in zahtevajo izbiro primerne strojne opreme – strežnika in odjemalca.

Posebno pri strežniškem računalniku moramo biti pazljivi, da ne pride do izgube podatkov ali kakršnihkoli motenj zaradi počasnosti računalnika; strežnik mora biti dovolj zmogljiv, kajti na njem se izvajajo vsi glavni procesi obdelave podatkov. Računalnik v nobenem primeru ne sme biti ozko grlo sistema, zaradi katerega bi se povečal čas cikla proizvodnje.

Zato načrtujemo uporabo računalnika proizvajalca HP z jedrnim procesorjem Intel i7 6 s taktom 3,2 – 4,6 GHz. Računalnik ima 8 GB rama tipa DDR4 in SSD trdi disk velikosti 256GB na vodilu SATA3. Poleg tega trdega diska potrebujemo še enega popolnoma enakega, preko katerega bomo lahko vzpostavili sistem povezovanja trdih diskov RAID1. Na tem računalniku s 64-bitnim operacijskim sistemom *Windows 10* bomo uporabili kar integrirano grafično kartico, saj ta komponenta nima bistvenega vpliva na naš sistem. Podobno velja tudi za USB priključke, ki jih uporabljamo le za priklop miške in tipkovnice tekom konfiguracije strežnika. Pomembno pa je, da imamo na računalniku dve gigabitni mrežni kartici. Ena kartica namreč povezuje strežnik z omrežjem podjetja (omogoča, da je računalnik dostopen iz sleherne pisarne) druga pa vodi do stikala in ločenega omrežja, ki povezuje vse naprave na liniji ter prikazovalnike (odjemalce) s strežnikom. Strošek takega računalnika znaša okoli 1400 € [2]. Poleg tega je koristno dokupiti še HP-jev UPS, ki bo ohranjal napetost na strežniku za vsaj 30 minut tudi v primeru izpada električne energije (okoli 450 €) [22].

Drugače kot pri strojni opremi strežnika lahko razmišljamo pri strojni opremi odjemalca. Za delovanje odjemalca ne potrebujemo hitrega računalnika, saj ne opravlja nikakršnih

zahtevnih operacij, temveč le komunicira s strežnikom. V študiji izvedljivosti (poglavje 2) smo razmišljali o možnosti uporabe računalnika *Raspberry Pi*, vendar smo pri testiranju orodja *Ignition* prišli ugotovitve, da je odjemalec vseeno prezahteven za normalno delovanje na tem mini računalniku. Zato smo se odločili, da bo na vsakem delovnem mestu ASUS-ov mini računalnik ASUS PN40 s 64-bitnim operacijskim sistemom *Windows 10*. Ta računalnik ima procesor Intel Celeron 1.1 GHz in 4GB notranjega pomnilnika skupaj z majhnim trdim diskom (SSD 120GB). Cena takega računalnika znaša okoli 200€, medtem ko so vse HP-jeve alternative veliko dražje, saj ne ponujajo možnosti nakupa manj zmogljivega mini računalnika, zatorej lahko s tem utemeljimo odločitev za nakup ASUS-ovih računalnikov [1].

Namen odjemalca je tudi ta, da delavcu na delovnem mestu prikazuje ter omogoča upravljanje s sistemom. Obe zahtevi bomo zagotovili z zaslonom na dotik proizvajalca Iiyama. 22-palčni zaslon ima optično zaznavo dotika, kar je idealno za delavce, ki imajo rokavice, saj optični zaslon deluje tako, da zazna dotik ob prekinitvi žarkov iz kamer v dveh vogalih ekrana. Cena takega zaslona se giblje okoli 250 € [3].

Preostala strojna oprema, kot je zgoraj omenjeno stikalo, preko katerega je povezan strežnik z linijo, je že v uporabi na liniji in je ni potrebno dodajati ob implementaciji SCADA sistema, zato je tukaj ne bomo obravnavali.

4.5 PREDVIDENE TEHNOLOGIJE

V tem poglavju so opisane tehnologije, ki jih nameravamo uporabiti tekom projekta. Omenjeni so tudi nekateri razlogi za izbiro teh rešitev ter omejitve teh tehnologij.

Vse tehnologije omenjene v nadaljevanju bomo uporabili na strežniku, na kateremu teče operacijski sistem *Windows 10*.

4.5.1 Ignition SCADA

Programsko orodje *Ignition* je produkt podjetja *Inductive Automation*. Prvo verzijo je podjetje izdalo januarja 2010 [23], sedaj pa smo v primeru našega projekta uporabili verzijo 7.9.10 iz novembra 2018 [14]. Orodje nudi lažjo implementacijo SCADA sistema. Zgrajeno je v programskem jeziku Java in je tako zlahka prenosljivo med platformami [8].

Osnovne komponente *Ignition* orodja so tri: mrežni prehod, orodje za oblikovanje poimenovano *Designer* in odjemalec. Mrežni prehod je spletni strežnik kjer se izvajajo vsi glavni procesi. Tam lahko tudi urejamo nastavitve, se povezujemo z napravami in podatkovnimi bazami ter zaženemo orodje *Designer* in odjemalca. Odjemalec služi kot vmesnik uporabniku, ki lahko potem preko tega upravlja s sistemom. V našem primeru je

mrežni prehod na strežniku na liniji, odjemalec pa na prikazovalnikih na posameznih delovnih mestih [6].

S pomočjo *Ignition Designerja* zgradimo uporabniški vmesnik za odjemalca in vso logiko delovanja SCADA sistema. Logiko večinoma pišemo v skriptah programskega jezika Jython 2.5 (implementacija Python programskega jezika) ter v tako imenovanem *izraznem jeziku* orodja *Ignition*. Skripte se izvedejo ob dogodkih, ki jim jih dodelimo ter tako narekujejo delovanje našega SCADA sistema. Ti dogodki so lahko povezani na primer s pritiskom na komponento uporabniškega vmesnika ali s spremembo vrednosti značke [6].

Značka (angl. *tag*) je nekakšna globalna spremenljivka, ki ohranja svojo vrednost dokler je ne spremenijo tisti, ki imajo dovoljenje za pisanje v značko. Značke lahko uporabljamo kot preslikave vrednosti značk na PLC-jih naprav ali kot svoje značke (značke orodja *Ignition*), v katere lahko shranjujemo poljubne vrednosti različnih podatkovnih tipov. Torej lahko v posamezno značko shranimo tako kompleksen podatek kot je celotna tabela rezultata neke poizvedbe v podatkovni bazi kot tudi enostavno boolovo vrednost. *Ignition Designer* ima tudi možnost kreiranja lastnih podatkovnih tipov značk (t. i. UDT), katerih instance lahko potem kreiramo in uporabljamo [6].

Tudi za poizvedbe po podatkovni bazi imamo svoj modul, ki ga lahko kličemo iz skript oz. ob različnih dogodkih. Imenuje se modul *Named Queries* ali modul imenovanih poizvedb. Preko tega modula lahko vnaprej napišemo poizvedbe in jih neštetokrat kličemo skupaj s parametri, ki jih podamo zraven. Tako nam ni treba vsakič znova pisati enakih poizvedb [6].

Zraven osnovnih komponent imamo na voljo še veliko med sabo neodvisnih modulov, ki prinesejo dodatne funkcionalnosti, kot so na primer: alarmiranje, shranjevanje zgodovine značk, avtomatsko kopiranje značk v podatkovno bazo, podpora mobilnim platformam, implementacija spletnega brskalnika v uporabniški vmesnik itd. Opisali bomo le module, ki jih bomo uporabili v našem projektu [5][6].

Osnovni modul je imenovan *Vision Module*. To je modul, ki omogoča uporabo uporabniškega vmesnika – brez tega modula je možna le uporaba logike v ozadju, brez odjemalcev. Dva modula, ki sta med sabo povezana, sta *OPC UA server module* in ustrezen *Siemensov* gonilnik za PLC-je, ki so na napravah na liniji. Zaradi gonilnika lahko *Ignition* uspešno dostopa do značk PLC-ja naprave na delovnem mestu, s pomočjo standarda OPC UA pa se značke predstavljajo v orodju *Ignition* in lahko z njimi operiramo (beremo in pišemo v značke) [6].

V projektni nalogi bomo uporabili tudi *Web browser module*. Ta nam omogoča, da v uporabniški vmesnik dodamo okno z brskalnikom, preko katerega lahko dostopamo do katerekoli spletne strani. Preko tega modula bomo predvidoma tako lahko implementirali pregledovanje dokumentacije v sistemu. Uporabniški vmesnik za pregled dokumentacije je

namreč na voljo preko internetne strani in zaradi njegove kompleksnosti se nismo odločili za njegovo predelavo v *Ignition* [6].

Majhen problem orodja *Ignition* je ta, da zahteva nekatera specifična znanja za uporabo tega orodja, ki jih je treba pridobiti – a vendarle se moramo tudi zavedati, da to velja za vsa orodja SCADA. Ravno ta specifičnost pa je tudi prednost tega orodja, kajti to pomeni, da je namenjena ravno SCADA sistemu in je zato popolnoma prilagojena njegovim potrebam. Razlog za implementacijo SCADA sistema z orodjem *Ignition* tiči tudi v razširjenosti njegove uporabe. Uporablja ga namreč veliko velikih, vsem znanih podjetij po svetu, kot so npr. *Shell, Coca Cola, Starbucks, Good Year* in *Walt Disney* [10].

4.5.2 OPC UA

OPC je protokol za komunikacijo med več napravami ali napravo in sistemom. Njegova kratica ponazarja *OLE for Process Control*, kar v slovenščini pomeni OLE za nadzor procesov, kjer je OLE Microsoftov mehanizem za med procesno komunikacijo. OPC je bil prvič predstavljen leta 1995 in je v naslednjih desetih letih postal najbolj uporabljen način za komunikacijo na področju avtomatizacije v vseh vrstah industrije. Slabost OPC-ja je ravno ta, da sloni na Microsoftovi tehnologiji OLE in DCOM, kar pomeni, da se ga lahko uporablja le na platformi *Windows* [17].

Imamo več OPC protokolov, ki so med seboj povsem neodvisni, v našem primeru pa uporabljamo protokol OPC DA (Data Access). S pomočjo tega protokola se podatki pridobijo iz kontrolnega sistema v ostale sisteme na proizvodni liniji. Za vsak podatek pridobimo vrednost podatka, naziv podatka ter čas zajema podatka in kakovost podatka [17].

OPC UA je izboljšava OPC, katere prva verzija je izšla leta 2006. Glavna razlika med OPC in OPC UA je ta, da slednja ne več temelji na tehnologiji OLE in DCOM in je tako neodvisna in delujoča na več platformah (*Windows, Linux, Apple*). Druga prav tako pomembna izboljšava je ta, da se lahko pri predstavitvi podatkov uporablja strukture oz. modele – to pomeni, da se lahko posamezne podatke porazdeljuje v skupine in se jim dodeli nekakšen kontekst, kar omogoča uporabnikom lažje razumevanje, upravljanje in vzdrževanje podatkov [17]. V našem primeru bomo torej uporabili protokol OPC UA, ki ga podpira tudi orodje *Ignition*.

4.5.3 Tehnologije lokalne baze

Za izvedbo lokalne podatkovne baze bomo uporabili Microsoftov **SQL Server Express,** in sicer različico 2014. Omenjeno različico, čeprav je starejša, uporabljamo zaradi preverjene kompatibilnosti z že obstoječim sistemom MES (kopiranje lokalne v globalno podatkovno bazo). Zaradi dolgoletne uporabe v sistemu MES v podjetju se lahko zanašamo tudi, da na

dolgi rok rešitev s *SQL Server Express* preverjeno deluje, kar je tudi en od glavnih razlogov za izbiro tega orodja.

Drugi velik razlog za uporabo omenjenega orodja je, da je zastonj hkrati pa ustrezno zadovolji kriterije iz nefunkcijskih zahtev, kot so odzivnost, razpoložljivost in robustnost. Omogoča enostavno izvedbo podatkovne baze z nekaterimi omejitvami. Najbolj omejujoča od teh je maksimalna velikost posamezne podatkovne baze, ki je 10GB [15]. Omenjen problem rešujemo s pomočjo avtomatskega brisanja iz podatkovne baze, ki je implementirano s pomočjo javanskega ogrodja *Java Spring*. Podatkov z brisanjem seveda ne izgubimo, saj so ti na voljo v globalni podatkovni bazi.

4.5.4 Tehnologije razvojnega okolja

Sistem se razvija na računalniku z operacijskim sistemom *Windows 10*. Diagrami uporabljeni tekom faz programskega inženirstva so ustvarjeni z brezplačno spletno aplikacijo *draw.io* [12], ves tekst pa v orodju *Microsoft Office Word 2016*.

Ker *Ignition Designer* sam po sebi omogoča pisanje skript direktno v orodju, načeloma ne bi potrebovali nobenega dodatnega urejevalnika teksta za programiranje sistema. Žal pa je modul za pisanje skript v orodju *Ignition* precej pomanjkljiv in ne omogoča nikakršnih prednosti, ki jih imajo sodobni urejevalniki teksta. Prav tako orodje *Ignition* ne omogoča odpiranja posameznih modulov v več oknih, kar pomeni, da je potrebno stalno preklapljanje med pogledi, kar velikokrat otežuje preglednost dela v orodju. Zato pri pisanju skript uporabljamo *Visual Studio Code* [16], ki je urejevalnik z raznimi dodatki, kot je dodatek za predvidevanje besed ter dodatek, ki obarva sintaktične in oblikovne nepravilnosti programskega jezika Python.

Ignition Designer sloni na ogrodju Java, torej je potrebno imeti za razvoj sistema nameščeno tudi novejšo različico Javanskega JDK (verzija 11) [19]. Za konfiguriranje mrežnega prehoda *Ignition* smo do le-tega dostopali kar prek brskalnika (v našem primeru *Google Chrome*) po IP-ju strežnika.

Za enostavno upravljanje komponent in podatkov iz podatkovne baze uporabljamo *Microsoft SQL Management Studio*, ki je seveda kompatibilen z izvedbo baze. S pomočjo tega orodja lahko tudi poizvedujemo po bazi v jeziku *MS SQL* in tako načrtujemo poizvedbe, ki jih kasneje prenesemo v modul *Named Queries* v orodju *Ignition*.

5 IZVEDBA

V fazi izvedbe se osredotočimo izpolnjevanju načrtovanega v prejšnji fazi. Torej gre za dejansko gradnjo in programiranje sistema [21]. To poglavje smo razdelili na 3 glavne dele. Najprej bomo opisali način uporabe tehnologij, opisanih v načrtovanju, nato konfiguracijo sistema ter kako smo se organizirali v orodju *Ignition Designer*. Na koncu pa bomo opisali še postopek programiranja sistema. V vseh poglavjih bomo sproti razložili tudi način uporabe tehnologij opisanih v načrtovanju.

5.1 KONFIGURACIJA SISTEMA

Predenj smo lahko začeli s programiranjem delovanja SCADA sistema, je bila potrebna premišljena konfiguracija tako strežnika kot odjemalca. V tem poglavju bomo podrobneje opisali tisti del konfiguracije, ki je bolj specifičen za gradnjo dotičnega SCADA sistema in le omenili druge konfiguracije, ki jih je potrebno narediti tekom gradnje vsakršnega sistema.

Konfiguracija strežnika in odjemalca imata skupno namestitev operacijskega sistema ter konfiguracijo mrežnih kartic. Ostalo se več ali manj razlikuje od strežnika do odjemalca, zato bomo konfiguraciji enega in drugega obravnavali v ločenih poglavjih

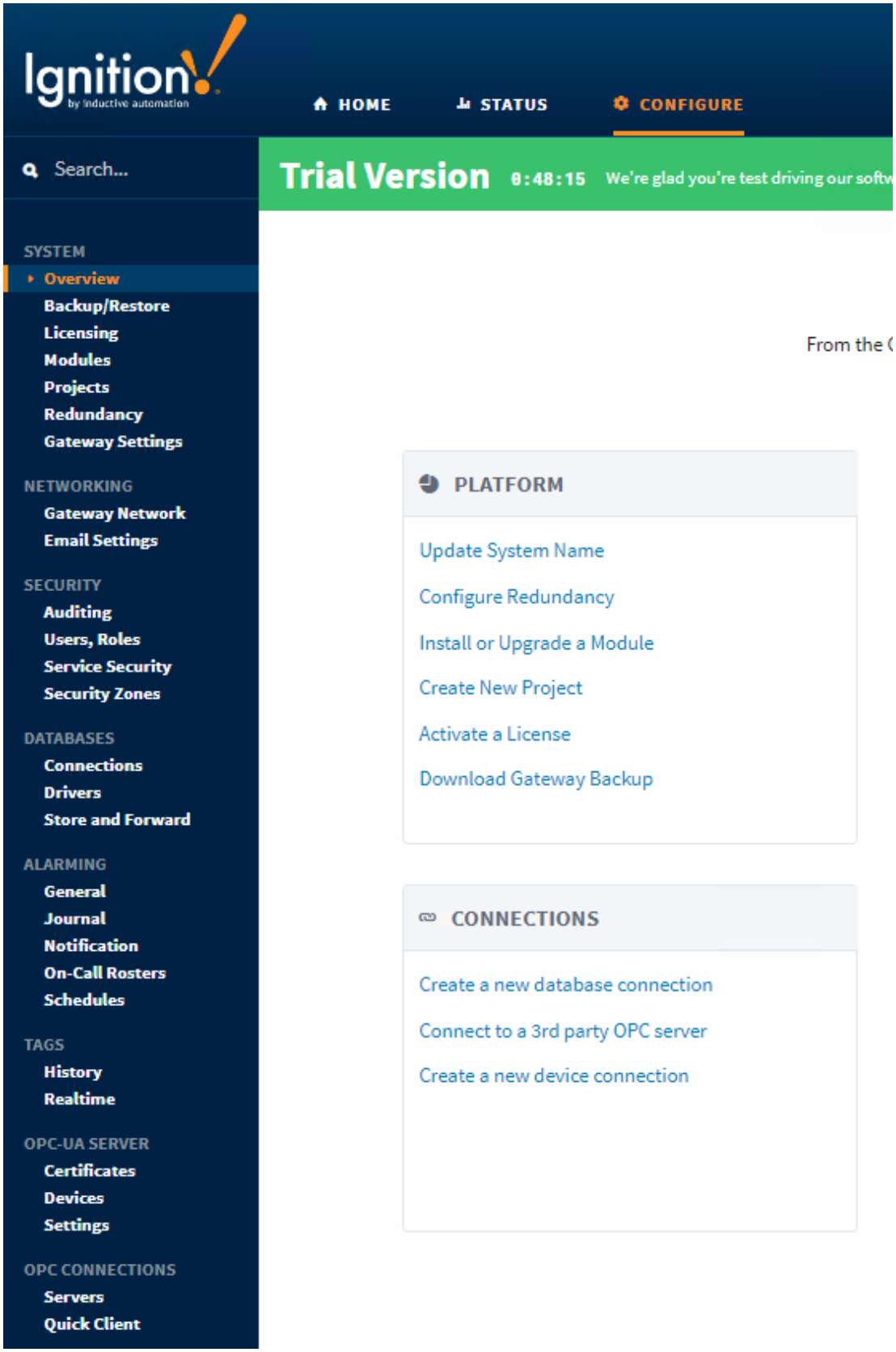
5.1.1 Konfiguracija strežnika

Na strežniku moramo v glavnem namestiti in nastaviti dve stvari – podatkovno bazo in orodje *Ignition*. Posvetili se bomo le slednjemu, saj je nameščenje in konfiguriranje podatkovne baze standardna procedura, ki ni prav nič specifična za ta dotičen projekt.

Za namestitev orodja *Ignition* moramo najprej namestiti *Java* JDK, nato na strežnik z brskalnikom prenesemo verzijo 7.9 tega orodja, ki jo dobimo na njihovi uradni spletni strani [4]. To potem z administratorskimi pravicami namestimo ter zaženemo mrežni prehod. Do konfiguracijske strani mrežnega prehoda lahko tako dostopamo preko IP-ja in privzetih vrat 8088 iz katerega koli računalnika v istem omrežju; tako ostalo konfiguracijo opravimo z delovnega računalnika in ne direktno na strežniku.

Ob prvi povezavi na nastavitveno stran mrežnega prehoda vnesemo privzeto uporabniško ime in geslo in ob pritisku na zavihek »Configure« vidimo meni, ki je prikazan na sliki 14. Na vrhu vidimo zeleno pasico, ki kaže, da gre za testno različico orodja. V pasici se odšteva tudi čas, dokler je ta različica še na voljo; ob preteku tega časa, s klikom na gumb v tej pasici ponastavimo čas na 2 uri do izteka. Ker delamo prototipni projekt in uporabljamo testno različico se nam ni potrebno ukvarjati z licenciranjem pod zavihkom

»Licensing« in z izbiro modulov (zavihek »Modules«), saj so ti vsi pod privzeto aktivirani in ne motijo delovanja našega sistema.



Slika 14: Videz menija konfiguracij na mrežnem prehodu Ignition

V meniju pod »Databases« določimo povezavo na podatkovno bazo. Gonilnika za bazo ni potrebno posebej namestiti, saj je gonilnik, ki ga potrebujemo mi (*Microsoft SQL Server JDBC*) že nameščen, zatorej lahko kar gremo pod zavihek »Connections« in tam s klikom na gumb začnemo z ustvarjanjem nove povezave. V naslednjem pogledu nato izberemo gonilnik naše baze in naposled preidemo na zadnjo stran ustvarjanja povezave na bazo. Tam enostavno določimo ime ter opis povezave na bazo in z URL-jem v zraven predpisani obliki povemo, kje se podatkovna baza nahaja ter vnesemo uporabniško ime in geslo preko katerega gonilnik *Ignition*-a dostopa do baze. Pod predelkom »Extra Connection Properties« nato še določimo instanco podatkovne baze, na katero se hočemo povezati po zraven napisani sintaksi in to zadošča za pritisk na gumb »Create New Database Connection«, ki ustvari povezavo, preko katere lahko nato delamo poizvedbe po podatkovni bazi.

Na mrežnem prehodu moramo dodati tudi naprave (PLC-je), ki so na liniji, da lahko dostopamo do značk teh naprav. V našem primeru imamo dve napravi. Dodamo jih tako, da gremo pod sekcijo »OPC-UA Server« in izberemo zavihek »Devices« ter gumb »Create new Device…«.Tam najprej izberemo tip naprave, ki je v našem primeru za oba PLC-ja *Siemens S7-300*. S pomočjo tega *Ignition* ve, katere gonilnike uporabiti za dostopanje do značk. Ko nadaljujemo na naslednji pogled, tam dodelimo novo dodani napravi ime in opis ter določimo IP in v naprednih nastavitvah vrata, preko katerih naj *Ignition* dostopa do naprave. Ko ta postopek opravimo dvakrat, smo dodali PLC-je obeh naprav, da lahko dostopamo do značk le-teh.

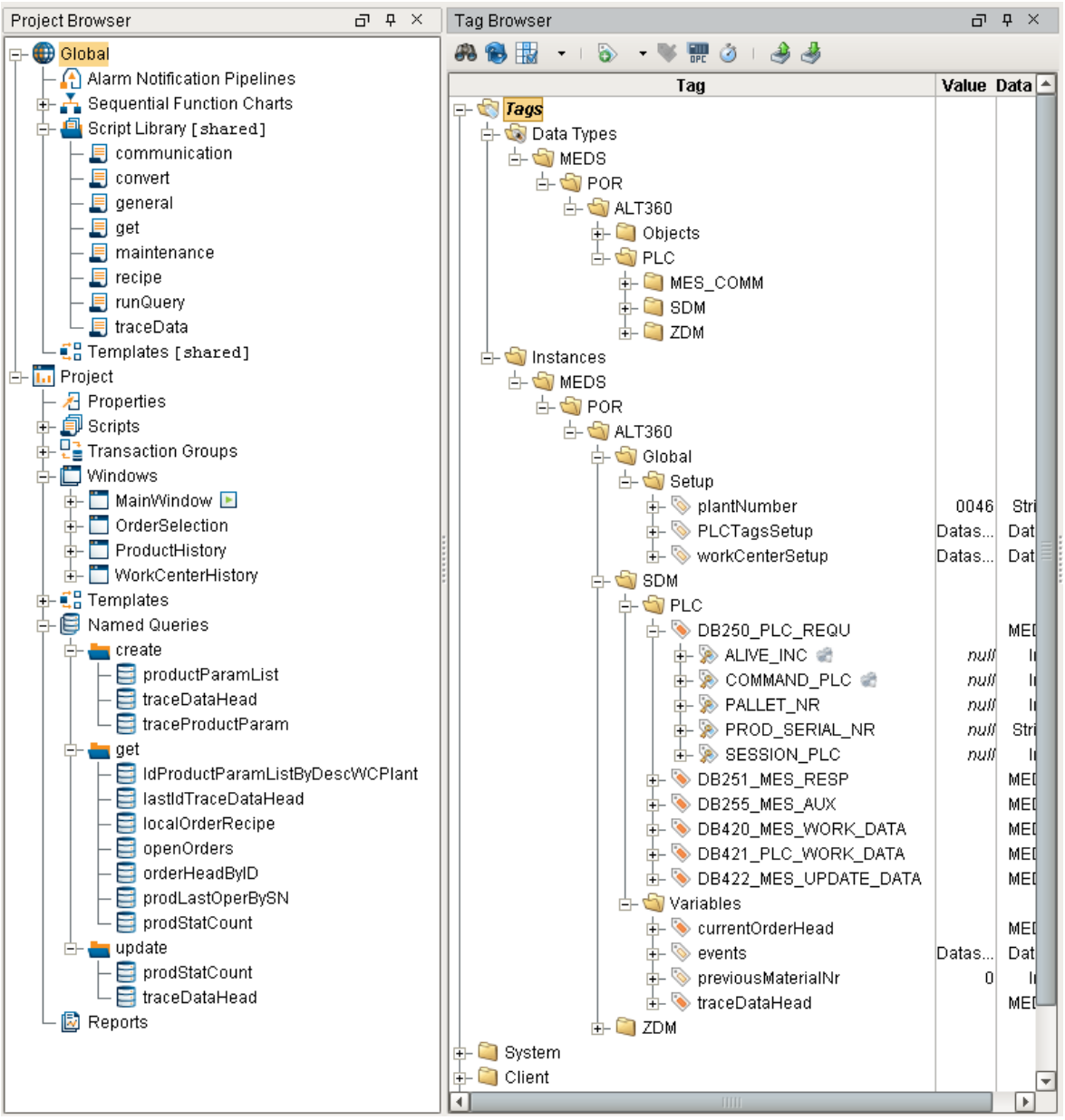
Imamo tudi veliko drugih zavihkov za nastavitve, vendar večina služi v naših očeh le minornim konfiguracijam ali takim, ki jih mi ne potrebujemo. V našem primeru npr. nismo implementirali nobenega sprožanja alarmov, torej ne potrebujemo zavihka »Alarming«, prav tako v zavihku »Security« nismo spreminjali nič, ampak smo le dodali novega uporabnika in privzetemu spremenili geslo. Smo pa dodali nove uporabnike (zavihek »Users, roles«) in zamenjali geslo privzetemu. Po vsem tem smo ustvarili še nov projekt, v katerem smo potem zgradili ta prototipni sistem.

5.1.2 Konfiguracija odjemalca

Konfiguracija odjemalca je precej enostavna. Na računalnik odjemalca namestimo le odjemalsko aplikacijo. To naredimo tako, da gremo zopet na stran za konfiguracijo mrežnega prehoda, kot v poglavju 5.1.1, le da sedaj izberemo zavihek »Home« na vrhu in spodaj prenesemo »Client Launcher for Windows« s pritiskom na gumb »Download«. Po prenosu ga namestimo in ga nato lahko zaganjamo s tega računalnika. Zaželeno je, da se ta program zažene ob vsakem zagonu računalnika (strežnik *Ignition* se kot privzeto vedno zažene) – zato bližnjico do le-tega premaknemo v mapo *startup* v operacijskem sistemu Windows, ki ima to funkcijo, da se ob zagonu vedno zažene vse, kar je znotraj nje.

5.2 SPLOŠNA ORGANIZACIJA *IGNITION DESIGNERJA*

Za čim lažje in čim hitrejše programiranje SCADA komponente v *Ignition Designer*ju je zelo koristno dobro organizirati skripte, imenovane poizvedbe, značke ter okna v okolju tega orodja. S pomočjo tega dosežemo predvsem večjo preglednost, razširljivost pa tudi večjo prenosljivost rešitve za eno delovno mesto na več delovnih mest. Prvo je pomembno predvsem zaradi sicer slabše preglednosti *Ignition Designer*ja. Pri drugem pa se zanašamo predvsem na dejstvo, da večja previdnost in počasnejše programiranje sedaj prinaša veliko manj dela pri prenosu te rešitve na preostala delovna mesta. To smo se trudili izpolniti predvsem s premišljenim poimenovanjem zgoraj omenjenih komponent *Ignition Designer*ja (skript, imenovanih poizvedb, značk, oken), ustreznim parametriziranjem in urejanjem teh komponent v ustrezne hierarhije. Na sliki 15 je viden izgled organiziranih komponent v *Ignition Designer*ju – na levi je vidna hierarhija in poimenovanje skript, oken in imenovanih poizvedb, na desni pa značk.



Slika 15: Videz organiziranih komponent orodja Ignition Designer

Odločili smo se, da bomo večina programske logike izpeljali skozi izvedbo funkcij v skriptah. Torej četudi *Designer* omogoča pisanje skript ob dogodkih različnih komponent (npr. značk, oken…), smo ob teh dogodkih opravili le zelo enostavno logiko odločanja ali kličemo določeno funkcijo v skripti in jo tako pokličem. Za tak pristop smo se odločili predvsem zaradi večje preglednosti; v tem primeru so namreč vse skripte in glavna programska logika na enem mestu. Funkcije v skriptah so razvrščene med osem skript. V preglednici 17 vidimo, po kakšnih kriterijih so funkcije razvrščene po teh skriptah.

Preglednica 17: Opis skript v orodju Ignition Designer

Ime skripte	**Opis funkcij v skripti**
communication	Funkcije, ki vsebujejo vso logiko, ki jo potrebujemo pri komuniciranju s PLC-ji naprav.
convert	Funkcije, s pomočjo katerih potekajo razne kompleksnejše pretvorbe tekom programa. Predvsem so tu pretvorbe nekaterih struktur orodja *Ignition* v Pythonove strukture po nam lastnih pravilih za lažjo inkorporacijo le-teh v ostalo programsko logiko.
general	Splošne funkcije, ki ne spadajo v nobeno od drugih skript.
get	Funkcije, ki služijo pridobivanju nekih vrednosti ali parametrov z nekakšno enostavno logiko.
maintenance	Funkcije, ki služijo vzdrževanju sistema in programa. Teh funkcij se v normalnem delovanju ne uporablja.
recipe	Funkcije, ki služijo upravljanju z recepti. Tukaj so predvsem kompleksne funkcije, ki pretvarjajo zapis recepta iz podatkovne baze v zapis, ki ga lahko posredujemo napravi.
runQuery	Preko teh funkcij kličemo vse imenovane poizvedbe v modulu *Named Queries*. To je opravljeno zato, da lažje kombiniramo nekatere poizvedbe ter da odpravimo nevšečnost *Ignition Designer*ja, ki zahteva, da v nekaterih primerih dodaš dodaten parameter pri klicanju imenovanih poizvedb, v nekaterih primerih pa tega parametra ne smeš dodati.
traceData	Funkcije, ki so namenjene upravljanju s podatki osrednje tabele v podatkovni bazi (*traceDataHead*) in vsej logiki odločanja okoli podatkov iz te tabele.

Tudi pri imenovanih poizvedbah smo ubrali podoben pristop kot pri skriptah. V tem primeru smo med mape razdelili poizvedbe glede na funkcijo poizvedb. Če gre za nekakšno kreiranje nove vrstice v tabeli, smo le te poizvedbe uvrstili v mapo »create«, če gre za posodabljanje vnosa v »update«, če pa gre za poizvedbe katerih osrednji namen je pridobivanje podatkov iz baze pa v mapo »get«.

Pri oknih nismo veliko kompliciral glede hierarhije, smo pa zato vsakemu oknu definirali parametre, preko katerih pošljemo oknu vrednosti. Glavni parameter je delovno mesto, na

katerem je odjemalec na katerem se izvaja program (vsako delovno mesto izvaja svojo instanco programa). Šifro delovnega mesta dobimo preko značke, v kateri je napisan IP, kjer se izvaja program in preko preslikave tega IP-ja v šifro delovnega mesta. Preko tega glavnega parametra se nato prikaže ustrezna vsebina okna za vsako delovno mesto. Podobno smo parametrizirali tudi komponente na oknih, da le-te ustrezno prikazujejo podatke glede na delovno mesto, na katerem se program izvaja.

Za značke smo morali biti posebej premišljeni pri ustvarjanju hierarhije – vsako premikanje značk po mapah namreč predstavlja kar velik problem, saj vse izgubijo vse reference po značkah v programu. Odločili smo se za čim bolj urejeno hierarhijo po mapah za slučaj, če se bo podoben projekt delal tudi na morebitnih drugih linijah tudi izven podjetja ali v podružnicah podjetja. Na sliki hierarhije (slika 15) je vidno, da smo razdelili značke po delovnih mestih; na značke preslikane iz PLC-ja ter na lastne značke, ki nam služijo kot nekakšne globalne spremenljivke. Prav tako smo naredili tudi značke, ki niso odvisne od delovnega mesta, temveč le od linije. Taka globalna značka je npr. *workCenterSetup*, kjer imamo preslikavo šifre delovnega mesta v razne druge lastnosti tega delovnega mesta (tudi npr. IP). Taka struktura značk nam omogoča dobro parametrizacijo in tako karseda lahko prenosljivost skonstruiranega sistema iz enega delovnega mesta na drugo delovno mesto, z malo več truda pa tudi mogoče na drugo proizvodno linijo.

Kreirali smo nekaj lastnih podatkovnih tipov značk (UDT), kjer je bilo smiselno. Svoj UDT ima tudi vsak podatkovni blok preslikan iz PLC-jev. V teh UDT-jih tako določimo točno lokacijo raznih značk na PLC-jih; te lokacije so nam morali posredovati programerji PLC naprav.

V *Ignition*u smo implementirali tudi večjezičnost; *Ignition* ima enostaven modul za prevajanje, ki smo ga uporabili, da bi zadovoljili zahtevo po večjezičnosti. Prav tako je potrebno še omeniti, da smo pri gradnji sistema sistem redno varnostno kopirali, kar omogoča lahko spremljanje verzij projektov ter v primeru nedelovanja povratek na starejšo verzijo.

5.3 PROGRAMIRANJE SISTEMA

Programske kode sistema žal ne moremo deliti, saj je le-ta last podjetja, ki kodo varuje kot poslovno skrivnost. Bomo pa zato opisali, kako smo postopali pri programiranju. Slednje bomo naredili na primeru uporabe opravljanje operacije na ZDM skupaj z menjavo naloga, opisano tudi na aktivnostnem (slika 11) in sekvenčnem diagramu (slika 13), saj je bilo postopanje pri programiranju enako tudi za druge primere uporabe z izjemo pregledovanja dokumentacije (pri slednjem je bil uporabljen modul *Web browser* orodja *Ignition*, ki služi kot brskalnik in prek katerega smo preko URL-ja dostopali in pregledovali dokumentacijo na enak način, kot bi to v navadnem brskalniku). Izpustili bomo interakcijo med napravo in

delavcem, ker pri tem ni vključenega programiranja sistema in se posvetili le interakciji s SCADA sistemom.

Na prikazovalniku odjemalca vedno prikazujemo šifro trenutno izbranega naloga. To enostavno naredimo tako, da vežemo ta prikaz na eno od značk, ki hrani nalog. Preko tega prikaza lahko delavec presoja ustreznost naloga in po potrebi pritisne na gumb, kateremu sledi odprtje novega okna, kjer imamo tabelo, ki prikaže aktivne naloge s tem, da izvede eno od imenovanih poizvedb, ki pridobi šifre aktivnih nalogov. Potem s klikom na enega od polj na tabeli zapremo to novo okno, pridobimo celoten nalog preko druge imenovane poizvedbe in ga zapišemo v značko, ki hrani nalog. V tej znački imamo tako shranjene vse podatke iz naloga, ki so v tabeli *OrderHead* v podatkovni bazi. Ob spremembi vrednosti te značke v skripti le primerjamo šifro tipa izdelka zapisanega v tej znački s šifro tipa izdelka, ki ga imamo zabeleženega v drugi znački, ki hrani šifro tipa prejšnjega izdelka (kar je tudi tip izdelka, za katerega je trenutno nastavljen stroj). V primeru, da sta vrednosti v značkah enaki v tabelo dogodkov (značka tipa tabela) zapišemo nov dogodek, ki ga na odjemalcu vidi delavec, da je le-ta informiran, da je bil nalog uspešno zamenjan in lahko začne s postavljanjem izdelka na delovno mesto.

Če se pa zgodi, da sta primerjani šifri tipa izdelka različni, preko funkcije v skripti najprej pokličemo imenovano poizvedbo, ki vrne ustrezen recept za tip izdelka. Sedaj druga funkcija v skripti obdela recept. Iz tipa JSON, v katerem je recept spravljen, naredi slovar (podatkovni tip jezika Python) poti do različnih značk na PLC-ju skupaj z vrednostmi teh značk. Nato na vsako od teh poti vpiše pripadajočo vrednost in PLC nastavi napravo tako, da ta deluje po teh navodilih. Podobno kot prej funkcija nato še doda dogodek, ki sporoči delavcu, da lahko položi izdelek na delovno mesto, saj sta bila tako nalog kot recept uspešno obdelana.

Ob prejetju izdelka naprava nakaže, da nima serijske številke, tako da pripadajočo značko (PROD_SERIAL_NR) na PLC-ju pusti prazno ter zahteva dovoljenje za delo s tem, da značko COMMAND_PLC postavi na vrednost 10. Ob spremembi te značke na 10 se potem kliče funkcija v skripti, ki preko imenovane poizvedbe najprej iz tabele *ProdStat* pridobi zaporedno številko izdelka v dnevu na gravuri in potem s pomočjo te številke in še nekaterih drugih podatkov kreira novo unikatno serijsko številko. Le-to nato vpiše v značko na PLC-ju PROD_SERIAL_NR, postavi značko PERMISS_WORK na 1, kar pomeni, da dovoli napravi delo na izdelku ter COMMAND_RESP značko izenači z vrednostjo COMMAND_PLC, kar napravi pomeni, da je dobila odgovor in lahko začne z operacijo v kolikor je PERMISS_WORK enak 1. Po tem funkcija še preko imenovane poizvedbe kreira nov vnos v tabeli *TraceDataHead* z vrednostmi, ki povedo, da se je začela in se trenutno opravlja operacija na delovnem mestu. Funkcija tudi izpiše dogodek delavcu, da se je operacija pričela.

Podoben postopek se zgodi, ko naprava zaključi z operacijo. Takrat postavi značko COMMAND_PLC na 20 in v določen podatkovni blok postavi vse značke na vrednosti, ki jih je izmerila tekom opravljanja operacije. Te značke predstavljajo procesne parametre. Ob spremembi COMMAND_PLC na 20 se spet izvede funkcija v skripti, ki preko imenovane poizvedbe zapiše uspešnost opravljanja operacije, tako da posodobi vnos v *TraceDataHead*, ki ga je sistem ustvaril prej, ko je dodelil dovoljenje za delo (primarni ključ tega vnosa se namreč hrani v eni izmed značk v sistemu). Istočasno tudi ta funkcija zgradi *insert* stavka v SQL jeziku, tako da pregleda vse procesne parametre in njihove vrednosti. Prvi *insert* stavek pregleda, če so že vsi parametri v šifrantu parametrov (*ProductParamList*) in po potrebi doda nov parameter, drugi pa zapiše vrednosti vseh parametrov v *TraceProductParam*. Ko se ta stavek uspešno izvede, funkcija najprej postavi COMMAND_RESP na 20, kar naprava zazna kot signal, da so se vsi parametri uspešno zapisali in da je operacija zaključena. Nato ista funkcija doda še dogodek kot informacijo delavcu, da je operacije konec in lahko izdelek odvzame iz delovnega mesta.

V opisu programiranja tega primera uporabe so vse prvine programiranja, ki smo jih uporabljali tekom programiranja sistema; klicanje funkcij, imenovanih poizvedb, grajenje SQL stavkov, dodajanje dogodkov, vezanje okenskih komponent na neko značko ali poizvedbo itd.

6 TESTIRANJE

Sistematično testiranje je običajno pomemben del gradnje vsakršnega sistema [21]. Kot slednje je bilo predstavljeno tudi vodstvu podjetja, vendar se je to zaradi časovne stiske in predvsem pomanjkanja kadra, ki ga bi dodelilo tej funkciji odločilo opustiti strogo sistematično testiranje. Zato je bilo testiranje izvedeno slabše, manj sistematično.

Tekom testiranja smo poskušali ugotoviti čim več napak na sistemu. Večino testiranja je opravil avtor te magistrske sam, je pa tudi sodeloval s sodelavci v podjetju, ki sicer niso bili del tega projekta z namenom, da bi tako odkrili čim več napak ter jih nato tudi odpravili. Iskanje izvora napake je v orodju *Ignition* sicer precej težko. Ta produkt namreč ne omogoča učinkovitega orodja za razhroščevanje.

Pri testiranju smo uporabljali princip tako strukturnega testiranja (*white box testing*) kot vedenjskega testiranja (*black box testing*). Opravljali smo ga v veliki meri že tekom gradnje sistema. Na ta način smo tako lahko že tekom te faze popravili veliko napak ter hkrati spoznali probleme oz. si zapisovali poti, na katere moramo paziti tudi tekom končnega testiranja. Na končno testiranje smo prešli šele, ko smo bili prepričani, da smo zadovoljili vse funkcijske zahteve iz funkcijske specifikacije.

Šele končno testiranje je potekalo dejansko na proizvodni liniji – torej tako, da so bile prisotne dejanske komponente (vhodi in izhodi sistema). Vsa prejšnja testiranja smo zaradi prevelikega oviranja gradnje linije morali opraviti s pomočjo simulacij vhodov in izhodov v sistem; to pa še vedno ne pomeni, da nismo že tekom gradnje testirali povezavo z vhodi in izhodi v sistem. Tako pričakovano ob testiranju na proizvodni liniji nismo naleteli na nove težave glede omenjenega.

6.1 USTREZANJE FUNKCIJSKIM ZAHTEVAM

Pri testiranju ustreznosti funkcijskim zahtevam smo se največ ukvarjali s problemom asinhronega izvajanja nekaterih delov kode. Naleteli smo na problem, ko se je funkcija izvedla preden se je v eno od značk zapisal odgovor iz podatkovne baze in je zato prišlo do napake. Ta problem smo rešili z razdelitvijo funkcije na dva dela (*Ignition* namreč v takih primerih ne omogoča sinhronih klicev). Prvi del je vseboval funkcijo do vključno z zapisom značke, ki nam je povzročala probleme, sinhrono pa smo za zapisom te značke dodelili vrednost še neki drugi znački (pisanje značk lahko izvedemo sinhrono). Ob tej slednji spremembi vrednosti smo nato sprožili drugi del prvotne funkcije. Po uporabi tega načina nismo imeli več problemov z asinhronim izvajanjem.

Pri testnih primeri smo poskušali zaobjeti vse možne poteke iz opisov primerov uporabe v poglavju 3.1.1. Tako smo za vsak primer uporabe naredili svoj testni primer. V

nadaljevanju bomo opisali testni primer pri testiranju opravljanja operacije na izdelku na ZDM skupaj z menjavo naloga. Testirali smo torej delovanje sistema v skladu z diagramom aktivnosti na sliki 11.

Na odjemalcu smo najprej kot akter delavec želeli menjati nalog in smo pritisnili na ustrezen gumb za menjavo naloga. Odprlo se je novo okno, kot je to ustrezno, kjer smo izbrali nov nalog, ki se je zabeležil v naš sistem, avtomatsko se je ob menjavi naloga tudi sprožilo preverjanje ujemanja šifre tipa izdelka iz trenutnega naloga s šifro tipa izdelka iz prejšnjega naloga in v primeru, da sta tipa različna tudi avtomatsko pridobivanje in beleženje recepta v simulator PLC-ja naprave. Sistem je po tem le ustrezno čakal, da mi, kot delavec, sprožimo novo zahtevo sistemu. To tudi smo storili s simuliranjem postavljanja izdelka na mesto za začetek operacije (v znački, ki je ponazarjala to opravilo v *Ignition*u, smo zabeležili ustrezno vrednost). Tedaj je SCADA sistem samostojno začel z poizvedbo po bazi, iz katere je dobil ustrezne podatke za generiranje serijske številke in jo tudi generiral ter ustrezno zapisal v značko simulatorja, kateremu je za tem tudi sporočil, da lahko začne s simuliranjem opravljanja operacije. Sporočanje o zaključku opravljanje operacije smo mi, kot testerji, simulirali z ročnim vpisom procesnih parametrov v značke simulatorja, od tu naprej pa je SCADA sistem ustrezno sam zaključil primer uporabe. V primeru, da smo simulirali slab izdelek, se je v podatkovni bazi zabeležil slab izdelek, v nasprotnem primeru pa dober.

Podobno smo testirali tudi vse druge primere uporabe. Pozneje smo na enak način testirali tudi sistem, ko je bil ta dejansko vpet na proizvodni liniji in prišli do enakih rezultatov kot pri uporabi simulatorja, kar je hkrati tudi potrdilo, da je simuliranje PLC-ja naprave učinkovito pri izvedbi testiranja.

6.2 USTREZANJE NEFUNKCIJSKIM ZAHTEVAM

Ustreznosti nefunkcijskim zahtevam žal nismo uspeli testirati na vseh področjih. Robustnost in razpoložljivost namreč lahko testiramo šele, ko je SCADA sistem v obratovanju, ker pa je to le prototipni projekt tega ne moramo poslati v obratovanje, saj ni kompleten, temveč je zgrajen le za dve delovni mesti. Tudi razširljivosti nismo mogli učinkovito testirati, saj pride zahteva do razširitev šele, ko je sistem v obratovanju. Tako nam preostane večjezičnost, ki smo jo učinkovito implementirali, prenosljivost, preglednost in odzivnost. Za testni primer prenosljivosti smo poizkušali rešitev iz SDM prenesti na neko drugo delovno mesto in to uspešno storili in zatorej bili zadovoljni s prenosljivostjo našega sistema. Testiranja prenosa na druge linije ni bilo, saj le-to ni smiselno dokler ne vzpostavimo polno delujočega sistema na eni proizvodni liniji.

Preglednost sistema smo se odločili testirati s krajšim testiranjem pri uporabnikih ter sodelavcih. Ti so uporabljali razne funkcije sistema (primeri uporabe iz funkcijskih zahtev,

kjer je akter delavec) in nam posredovali povratne informacije o preglednosti uporabniškega vmesnika ter koliko se lahko sredi opravljanja operacije na izdelku orientirajo v kakšnem stanju je to opravilo. Pri preučevanju niso bili izraženi nikakršni veliki pomisleki glede preglednosti (uporabniški vmesnik je pregleden, izpisi dogodkov na zaslonu pa delavca dovolj informirajo o poteku dogodkov na napravi), vseeno pa ne izključujemo možnosti, da se bodo manjše težave pojavile tekom obratovanja sistema. V vsakem primeru pa menimo, da ne bo težav dodelave, kajti upravljanje z elementi uporabniškega vmesnika ter dodajanje izpisov uporabniku je precej enostavno v zgrajenem sistemu.

Največ poudarka smo tako namenili odzivnosti. Čas odziva sistema smo merili v dveh primerih. In sicer čas, ki ga potrebuje sistem, da se odzove PLC-ju na zahtevo za dovoljenje za delo, in čas, ki ga potrebuje sistem, da shrani parametre, ki mu jih posreduje PLC. V specifikaciji zahtev smo za prvi primer zahtevali čas odziva, ki je manjši od 1 sekunde (enostavnejši primer), za drugi pa manjši od 2 sekund (kompleksnejši primer, saj zahteva hranjenje velike količine parametrov). V prvem primeru smo pri prvih meritvah pridobili čas, ki se je gibal od 0,9 do 1,4 sekunde, v drugem primeru pa od 2 do 5 sekund. S temi časi smo bili nezadovoljni, še posebej zaradi velikega nihanja med različnimi meritvami. Po razhroščevanju smo ugotovili, da je razlog za tako velike razlike med časi intervalno zajemanje vrednosti iz značk PLC-jev. Tako smo ta problem odpravili z uporabo modula *Scan class* v *Ignitionu,* s pomočjo katerega osvežujemo na vsakih 10ms le značko COMMAND_PLC, ki služi kot sprožilec za odziv SCADA sistema. Vse ostale značke pa se osvežijo, le, ko se ta značka spremeni. V tem primeru smo prišli do boljših ter veliko bolj konsistentnih časov. V primeru odgovarjanja na zahtevo za delo smo tako potrebovali povprečno 0,74 sekunde, za zapis parametrov pa 1,77 sekunde. Omenjeni časi še vedno niso odlični, vendar glede na ogromno število procesnih parametrov solidni in so zadovoljujoči glede na čase, ki so določeni v specifikaciji zahtev.

7 ZAKLJUČEK

V raziskovalni nalogi je opravljen razvoj in implementacija SCADA sistema po fazah programskega inženirstva. Med študijo izvedljivosti smo odkrili razloge, da se lotimo najprej prototipnega sistema namesto gradnje celotnega sistema. Tako se torej vse druge faze nanašajo na prototipni sistem. Te faze so: specifikacija zahtev, načrtovanje, izvedba in testiranje. Fazo obratovanja in vzdrževanja smo tako izpustili, saj gre za gradnjo prototipnega projekta, iz katerega bomo lahko šele naknadno zgradili celoten sistem, ki bo predan v obratovanje.

Specifika tega sistema je, da smo pri gradnji uporabljali orodje *Ignition* SCADA. V primerjavi s trenutno implementiranim hišnim SCADA sistemom, ki je v podjetju, na tak način dosežemo lažjo implementacijo sistema na novo proizvodno linijo. Ta zahteva tudi manj dela s strani programerjev ter je bolj zanesljiv, saj za orodjem stoji podjetje *Inductive Automation*, ki nudi podporo uporabnikom ter izdaja redne posodobitve te programske opreme.

Z izgradnjo tega prototipnega sistema smo skušali vodstvu podjetja pokazati, da je postavitev sistema z manj delovne sile možna ter da pri tem nimamo omembe vrednih slabosti v primerjavi s trenutno implementiranim sistemom. Ta raziskovalna naloga lahko tako služi vodstvu kot pomoč pri odločanju o vložku v celoten sistem, ki je zgrajen na ta način.

V primeru, da pride do gradnje celotnega sistema, imamo tudi predloge za možno izboljšavo sistema. Ena izmed njih je uporaba modula SFC (*Sequential Function Charts*) v *Ignition*u, ki bi razrešil težave z asinhronim izvajanjem, ki smo jih sicer v gradnji tega prototipnega sistema razreševali drugače. Namesto s postavljanjem značk, kot je opisano v poglavju testiranje, SFC omogoča povezovanje blokov kode med sabo, ti bloki pa se nujno sekvenčno izvajajo. Ta rešitev je veliko ustreznejša s strani preglednosti programske kode, hkrati pa prinaša še eno prednost – boljše spremljanje poteka opravljanje operacije na delovnem mestu. Delavcu na odjemalcu lahko prikazujemo, kateri blok kode se trenutno izvaja in tako je ta obveščen, kje v procesu opravljanja operacije se trenutno nahaja sistem. SFC modul je sicer plačljiv (dodatnih 3.000 €) in zato ni bil uporabljen v prototipnem projektu, vendar smo tekom gradnje projekta spoznali uporabno vrednost tega modula. Drugi predlog izboljšave je vpisovanje v odjemalca z različnimi profili (delavec, preddelavec, vzdrževalec) in na podlagi tega prikazovanje različnih možnosti na zaslonu odjemalca, kar bi omogočalo lažje vzdrževanje. Vendar so vse te izboljšave predmet obravnavanja le v primeru, da se vodstvo v prihodnosti odloči vložiti v gradnjo celotnega sistema.

8 LITERATURA IN VIRI

[1] Asus, *Mini PC PN40*, n.d. [Na spletu]. Dostopno na: https://www.asus.com/Mini-PCs/Mini-PC-PN40/. [Datum ogleda: 18. junij 2019]

[2] HP, *HP Z2 G4 Workstation - Customizable*, n.d. [Na spletu]. Dostopno na: https://store.hp.com/us/en/pdp/hp-z2-g4-workstation-customizable-2yw27av-1. [Datum ogleda: 12. junij 2019]

[3] Iiyama, *PROLITE T2253MTS-B1*, n.d. [Na spletu]. Dostopno na: https://iiyama.com/gb_en/products/prolite-t2253mts-b1/. [Datum ogleda: 12. junij 2019]

[4] Inductive Automation, *Current Ignition Release*, n.d. [Na spletu]. Dostopno na: https://inductiveautomation.com/downloads/ignition/7.9.11. [Datum ogleda: 13. junij 2019]

[5] Inductive Automation, *Fully Integrated Modules for SCADA, IIoT, MES, and More*, n.d. [Na spletu]. Dostopno na: https://inductiveautomation.com/ignition/modules. [Datum ogleda: 31. marec 2019]

[6] Inductive Automation, *Ignition Documentation*, n.d. [Na spletu]. Dostopno na: https://docs.inductiveautomation.com/. [Datum ogleda: 31. marec 2019]

[7] Inductive Automation, *Ignition Software Pricing for SCADA, IIoT, MES and More*, n.d. [Na spletu]. Dostopno na: https://inductiveautomation.com/pricing/ignition. [Datum ogleda: 13. marec 2019]

[8] Inductive Automation, *Inductive Automation Software Solutions*, n.d. [Na spletu]. Dostopno na: https://inductiveautomation.com/. [Datum ogleda: 8. marec 2019]

[9] Inductive Automation, *Inductive University*, n.d. [Na spletu]. Dostopno na: https://www.inductiveuniversity.com/. [Datum ogleda: 8. marec 2019]

[10] Inductive Automation, *Our Customers*, n.d. [Na spletu]. Dostopno na: https://inductiveautomation.com/about/customers. [Datum ogleda: 31. marec 2019]

[11] J. Bratina, *Razvoj sistema za sledljivost izdelkov in vpeljava na proizvodno linijo*, FAMNIT, Koper, 2019

[12] JGraph Ltd., *draw.io*, n.d. [Na spletu]. Dostopno na: https://draw.io/. [Datum ogleda: 13. junij 2019]

[13] M. Rouse, *What is manufacturing execution system (MES)?*, n.d. [Na spletu]. Dostopno na: https://searcherp.techtarget.com/definition/manufacturing-execution-system-MES. [Datum ogleda: 17. junij 2019]

[14] M. Taloa, *Ignition 7.9.10 Now Available*, Inductive Automation, 29 november 2018. [Na spletu]. Dostopno na: https://support.inductiveautomation.com/index.php?/News/NewsItem/View/69/ignition-7910-now-available. [Datum ogleda: 31. marec 2019]

[15] Microsoft, *SQL Server Documentation*, December 2018. [Na spletu]. Dostopno na: https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation?view=sql-server-2017. [Datum ogleda: 8. marec 2019]

[16] Microsoft, *Visual Studio Code*, n.d. [Na spletu]. Dostopno na: https://code.visualstudio.com/. [Datum ogleda: 13. junij 2019]

[17] Novotek, *OPC and OPC UA explained*, n.d. [Na spletu]. Dostopno na: https://www.novotek.com/en/solutions/kepware-communication-platform/opc-and-opc-ua-explained. [Datum ogleda: 7. april 2019]

[18] ORACLE Netsuite, *What is ERP*, n.d. [Na spletu]. Dostopno na: http://www.netsuite.com/portal/resource/articles/erp/what-is-erp.shtml. [Datum ogleda: 17. junij 2019]

[19] ORACLE, *Java SE Development Kit 11*, n.d. [Na spletu]. Dostopno na: https://www.oracle.com/technetwork/java/javase/downloads/jdk11-downloads-5066655.html. [Datum ogleda: 13. junij 2019]

[20] Ordinal Software, *SCADA and MES : the pyramids 'secret*, n.d. [Na spletu]. Dostopno na: https://www.ordinal.fr/en/scada-and-mes-the-pyramids-secret.htm [Datum ogleda: 17. junij 2019]

[21] P. Rogelj, *Skripta za predmet programsko inženirstvo*, FAMNIT, Koper, 2014

[22] Tehnox, *HPE T1000 G5 INTL Tower UPS*, n.d. [Na spletu]. Dostopno na: https://www.tehnox.si/hpe-t1000-g5-intl-tower-ups-q1f50a-p-63758.html. [Datum ogleda: 29. junij 2019]

[23] Wikipedia, *Ignition SCADA*, n.d. [Na spletu]. Dostopno na: https://en.wikipedia.org/wiki/Ignition_SCADA. [Datum ogleda: 31. marec 2019]