Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

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

Št. naloge: 00514/2010
Datum: 05.04.2010

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

Kandidat: **TINE SVETE**

Naslov: **RAZVOJ SISTEMA ZA PROCESNO UPRAVLJANJE AVTOMATIZIRANEGA SKLADIŠČA**

THE DEVELOPMENT OF WAREHOUSE CONTROL SYSTEM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Naredite analizo in načrt sistema za procesno upravljanje avtomatiziranega skladišča. Na podlagi tega sistem tudi razvijte. Pri tem upoštevajte zahtevo po uporabi WCF (Windows Communication Foundation) ter ostalih najsodobnejših tehnologij.

Mentor:

doc. dr. Rok Rupnik

Fakulteta za računalništvo in informatiko 2 UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Franc Solina

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Tine Svete

Razvoj sistema za procesno upravljanje avtomatiziranega skladišča

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Rok Rupnik

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

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

Št. naloge: 00514/2010
Datum: 05.04.2010

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

Kandidat: **TINE SVETE**

Naslov: **RAZVOJ SISTEMA ZA PROCESNO UPRAVLJANJE AVTOMATIZIRANEGA SKLADIŠČA**

THE DEVELOPMENT OF WAREHOUSE CONTROL SYSTEM

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Naredite analizo in načrt sistema za procesno upravljanje avtomatiziranega skladišča. Na podlagi tega sistem tudi razvijte. Pri tem upoštevajte zahtevo po uporabi WCF (Windows Communication Foundation) ter ostalih najsodobnejših tehnologij.

Mentor:

doc. dr. Rok Rupnik

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Franc Solina

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

diplomskega dela

Spodaj podpisani **Tine Svete**,

z vpisno številko **63000279**,

sem avtor diplomskega dela z naslovom:

RAZVOJ SISTEMA ZA PROCESNO UPRAVLJANJE AVTOMATIZIRANEGA SKLADIŠČA

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  doc. dr. Rok Rupnik

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

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

V Ljubljani, dne **5.7.2010** Podpis avtorja:

Zahvala

Zahvala gre v prvi vrsti staršem, ki so mi omogočili študij in ženi za izkazano potrpežljivost v času študija. Iskrena hvala tudi mentorju doc. dr. Roku Rupniku za strokovno pomoč pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic in simbolov

- **ERP** – informacijski sistem za upravljanje sredstev poslovnega sistema (ang. Enterprise Resource Planning).
- **WMS** – informacijski sistem za logistično upravljanje skladiščnih prostorov (ang. Warehouse Management System).
- **WCS** – informacijski sistem za procesno upravljanje avtomatiziranega skladišča (ang. Warehouse Control System).
- **PLC** – programabilni logični krmilnik (ang. Programmable Logical Controller).
- **HMI** – vmesnik človek-stroj (ang. Human Machine Interface).
- **SCADA** – nadzor in zajem podatkov (ang. Supervisory Control And Data Aquisition)
- **Skladiščna enota** (ang. Storage Unit) – najmanjša enolično označena enota tovora z vidika sistema WCS, tipično je to ena evro paleta
- **Skladiščno mesto** (ang. Storage Bin) – fizično mesto, na katerem se lahko v danem trenutku nahaja največ ena skladiščna enota z vidika sistema WCS
- **VRS** – (avtomatizirano) visoko-regalno skladišče (ang. High-Bay Warehouse)
- **ARD** – avtomatizirano regalno dvigalo
- **WCF** – programsko ogrodje za razvoj komunikacijskih povezav med procesi podjetja Microsoft (ang. Windows Communication Foundation)
- **WPF** – programsko ogrodje za razvoj bogatih interaktivnih uporabniških vmesnikov (ang. Windows Presentation Foundation)
- **IDE** – integrirano razvojno okolje (ang. Integrated Development Environment)
- **API** – vmesnik za izdelavo aplikacij (ang. Application Programming Interface)
- **CLR** – skupno izvajalno okolje (ang. Common Language Runtime)
- **UI** – uporabniški vmesnik (ang. User Interface)
- **XAML** – razširljiv označevalni jezik za aplikacije (ang. eXtensible Application Markup Language)

Povzetek

Sistem WCS je informacijski sistem namenjen procesnemu upravljanju avtomatiziranega skladišča. Zapolnjuje informacijsko vrzel med sistemi WMS in programabilnimi logičnimi krmilniki naprav v avtomatiziranem skladišču.

V diplomskem delu predstavljam možno zasnovo in implementacijo takega sistema s pomočjo nekaterih najnovejših Microsoft tehnologij za izdelavo programske opreme.

Posebno pozornost sem namenil ogrodju WCF in izdelavi podatkovne baze, ki predstavlja temelj vsakega informacijskega sistema. Ključna lastnost ogrodja WCF je zagotovitev enotnega programskega modela za izdelavo porazdeljenih sistemov. Ogrodje namreč enakovredno obravnava komunikacijo med dvema procesoma na istem računalniku, v istem omrežju ali med dvema storitvama, ki komunicirata preko HTTP.

Ključne besede:

- ERP,
- WMS,
- WCS,
- PLC,
- WCF.

Abstract

Warehouse Control System is information system for automated warehouse process management. It fills the gap between WMS systems and programmable logic controllers of automated warehouse devices.

The presented work offers possible design and implementation of such system with the help of some of the latest Microsoft technologies for software development.

Special attention was devoted to WCF framework and database design, which is the foundation of any information system. WCF framework key feature is to provide a single programming model for distributed production systems. WCF treats communication between two processes on the same computer, on the same network or between two services that communicate via HTTP, equally.

Keywords:

- ERP,
- WMS,
- WCS,
- PLC,
- WCF.

1 Uvod

Logistika je ena najhitreje rastočih panog gospodarstva. Tovrstna rast pa zahteva tudi sproten razvoj podporne programske opreme. Že nekaj let delujem na področju vzdrževanja in vpeljevanja sistemov za procesno upravljanje avtomatiziranih skladišč (ang. Warehouse Control System). V tem času sem se dodobra seznanil z omejitvami obstoječega sistema s katerim delam, kakor tudi željami končnih uporabnikov ter poslovnimi praksami. Zaradi tega sem se odločil v okviru diplomskega dela na podlagi dosedanjih izkušenj in s pomočjo nekaterih najnovejših Microsoft tehnologij zasnovati in izdelati informacijski sistem za procesno upravljanje avtomatiziranega skladišča.

Namen in cilji diplomskega dela so:

- se podrobneje seznaniti z nekaterimi najnovejšimi tehnologijami za razvoj programske opreme,
- na podlagi preteklih izkušenj zasnovati in izdelati sistem za procesno upravljanje avtomatiziranega skladišča.

V uvodnem delu bom predstavil umeščenost sistema ter funkcionalne zahteve, ki jim mora sistem zadostiti.

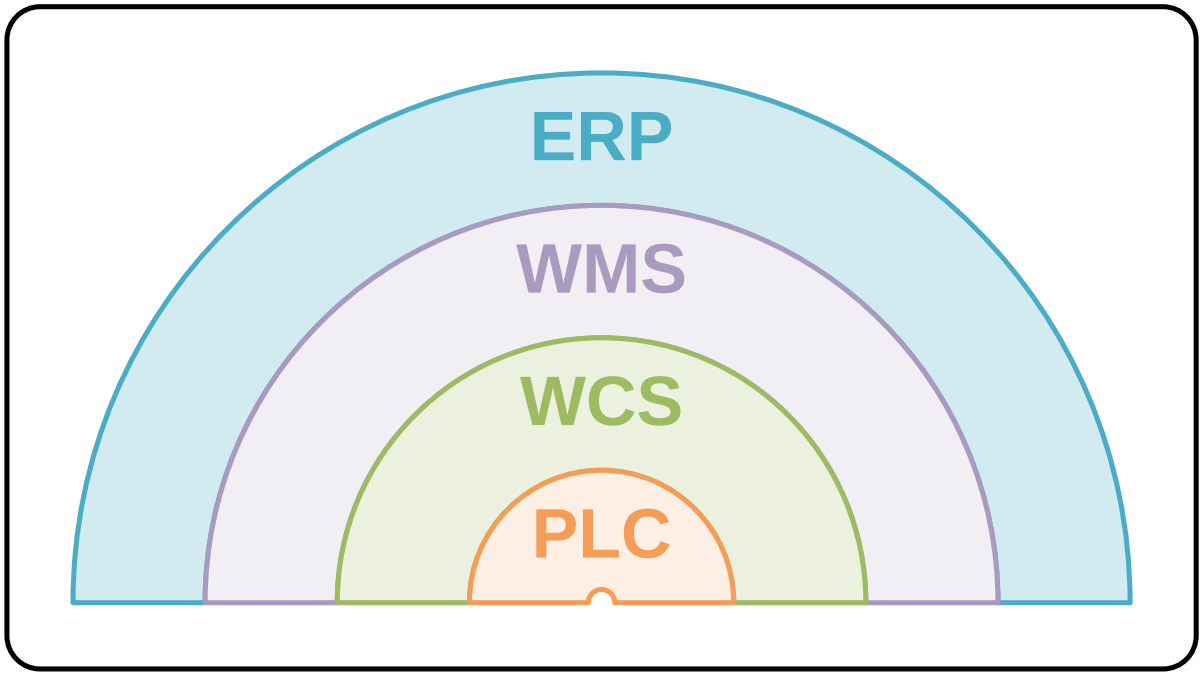
V nadaljevanju bom predstavil fizično in logično topografijo tipičnega avtomatiziranega skladišča, uporabljena orodja in tehnologije ter zamišljeno zasnovo oziroma arhitekturo sistema.

Osrednji del je namenjen predstavitvi izdelave posameznih elementov sistema.

V zaključnem delu sem izpostavil še nekatere probleme s katerimi sem se soočil med izdelavo ter ideje za nadaljni razvoj sistema.

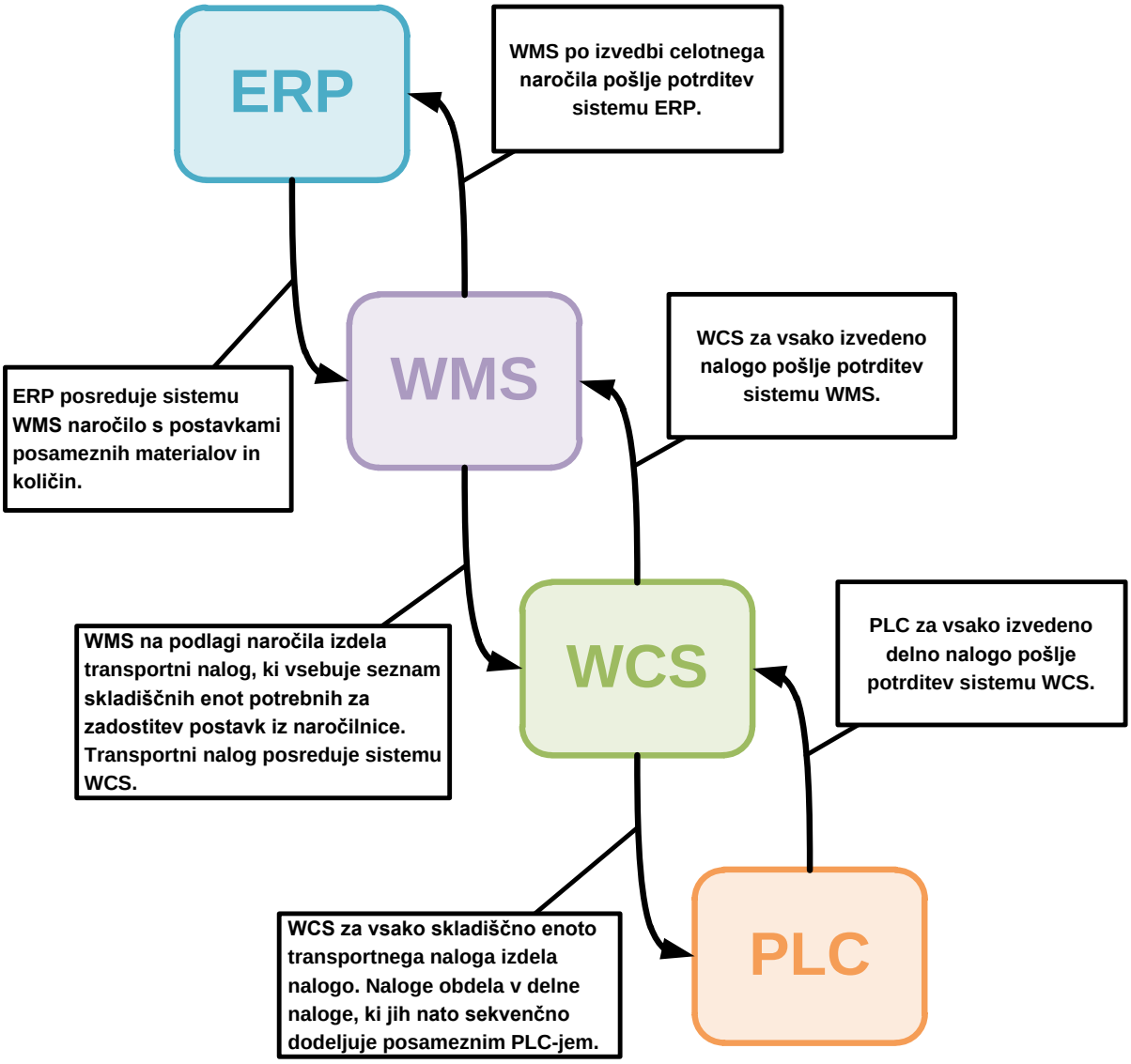
1.1 Umeščenost sistema

Informacijski sistem za procesno upravljanje avtomatiziranega skladišča ni samozadosten. Informacijsko podporo logistiki znotraj poslovnega sistema sestavlja več nivojev, kot prikazuje Slika 1. Sistem WCS je neposredno podrejen nivoju WMS in posredno nivoju ERP ter neposredno nadrejen nivoju PLC.



Slika 1: Nivoji poslovno-procesnega sistema

Med vsemi soležnimi nivoji poteka dvosmerna izmenjava podatkov. Nivoja ERP in WMS si izmenjujeta podatke na nivoju dobav in odprem, WMS in WCS na nivoju uskladiščnih in izskladiščnih transportnih nalogov ter nalog, WCS in PLC pa na nivoju nalog in delnih nalog (Slika 2).



Slika 2: Izmenjava podatkov med sistemi

1.2 ERP

ERP je informacijski sistem za upravljanje notranjih in zunanjih sredstev, vključno s premoženjem, finančnimi sredstvi, materiali in kadri. Omogoča učinkovit in celovit pretok informacij med poslovnimi funkcijami znotraj poslovnega sistema [1].
Z vidika logistike poslovnega sistema gre posebna pozornost nabavni in prodajni poslovni funkciji. Dogodki v okviru teh dveh funkcij namreč neposredno vplivajo na WMS ter posledično na WCS.

1.3 WMS

WMS je ključni element dobavne verige. Namenjen je nadzoru logičnih premikov tovora med skladiščnimi prostori in obdelavi s tem povezanih transakcij: uskladiščenje, izskladiščenje, preskladiščenje, dobava, odprema. WMS je lahko samostojen sistem ali del sistema ERP [2].
Informacijska meja med WMS in WCS ni univerzalno določena, tipično pa je VRS z vidika WMS črna škatla (ang. Black box) oziroma množica območij razdeljenih glede namembnost (polizdelki, izdelki, surovine, embalaža, ipd.). WMS ne obravnava topologije skladišča v smislu naprav in posameznih skladiščnih mest temveč v smislu namembnosti. Posamezna skladiščna enota se vedno nahaja v enem od območij glede na namembnost in ni pomembno, ali je to avtomatizirano visoko-regalno skladišče, komisionirnica, ali morda tovornjak na poti iz enega obrata v drugega. Skladiščna enota za WMS tipično predstavlja množico ene ali več enot materiala z definirano količino, šaržo, datumom izdelave in podobno.

1.4 WCS

WCS operira s skladiščnimi mesti in skladiščnimi enotami. Z vidika WCS je skladiščna enota s črtno kodo ali oznako RFID enolično označena enota, ki se lahko v danem trenutku nahaja samo na enem skladiščnem mestu.
Skladiščna enota ima za WCS naslednje lastnosti:

- enolična oznaka (črtna koda ali RFID),
- velikost,
- izmerjena teža,
- enolična oznaka skladiščnega mesta, na katerem se v danem trenutku nahaja.

Skladiščno mesto ima z vidika WCS naslednje pomembne lastnosti:

- enolična oznaka v koordinatnem sistemu,
- velikost,
- maksimalna dovoljena teža skladiščne enote,
- status zasedeno/nezasedeno (fizično stanje),
- status omogočeno/onemogočeno (logično stanje).

WCS izvaja naslednje operacije:

- uskladiščenje,
- izskladiščenje in
- preskladiščenje.

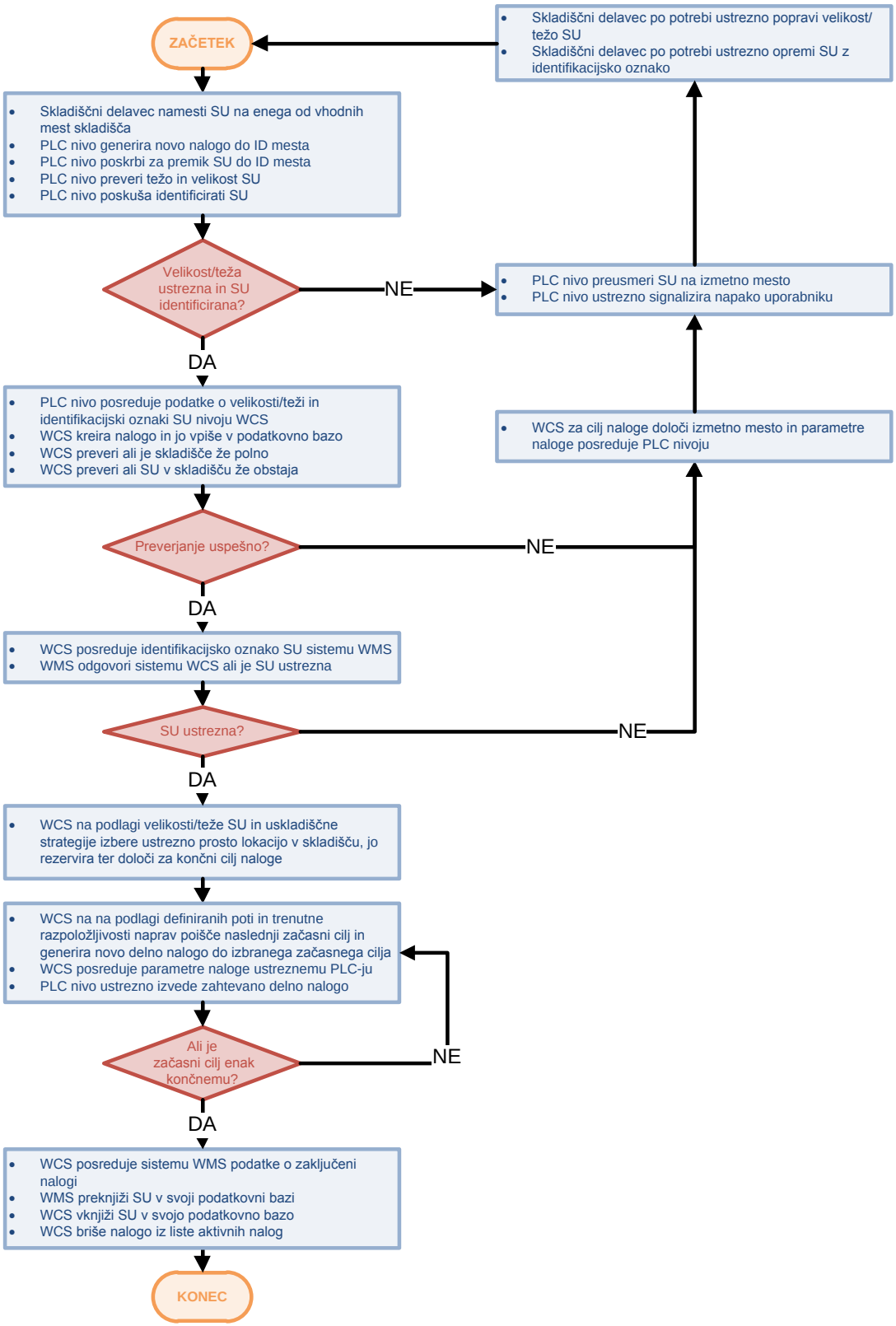
Zahtevke za posamezne operacije prejema od WMS, jih ustrezno obdela ter sekvenčno posreduje PLC-jem. Med avtomatskim obratovanjem skladišča posegi oseb niso potrebni, v primeru zastojev pa uporabniški vmesnik pooblaščenim osebam nudi ustrezna orodja za odpravo težav.

WCS skrbi za karseda učinkovito izrabo skladiščnih mest in razpoložljivih naprav. To pomeni, da pri uskladiščevanju izbira skladiščna mesta, ki minimalno zadostijo velikosti in teži skladiščne enote ter pri premikih izbira poti in naprave, ki predstavljajo najkrajšo razpoložljivo pot do cilja.

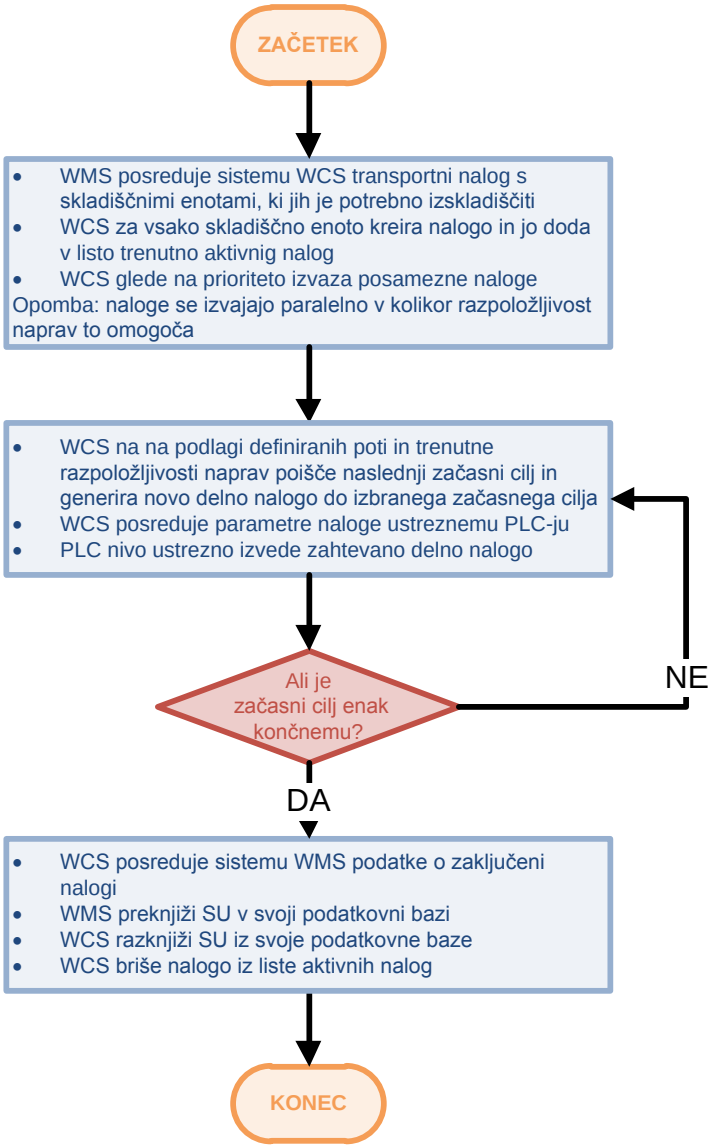
Pri uskladiščenju (Slika 3) je potrebno vsako skladiščno enoto identificirati ter določiti pomembne fizične atribute skladiščne enote – velikost, težo. To so za WCS potrebni in zadostni podatki za delo s skladiščnimi enotami.

Izskladiščenje (Slika 4) je enostavnejše, saj je potrebni in zadostni podatek sistemu WCS identifikacijska oznaka skladiščne enote, na podlagi katere WCS poišče skladiščno mesto iz katerega bo izvedel postopek izskladiščenja.

Z vidika sistema WCS skladiščna enota obstaja od trenutka, ko je identificirana na enem od vhodnih identifikacijskih mest skladišča, do trenutka, ko zapusti skladišče na enem od izhodov skladišča.



Slika 3: Diagram postopka uskladiščenja



Slika 4: Diagram postopka izskladiščenja

1.5 PLC

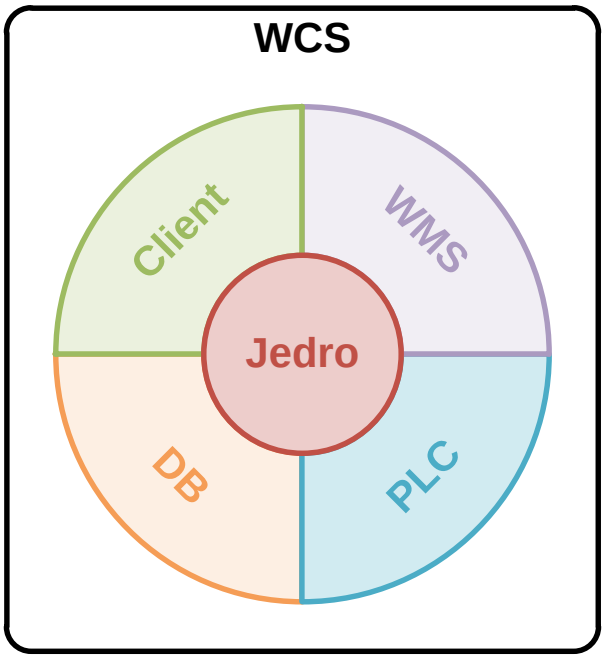
PLC nivo sestavlja eden ali več programabilnih logičnih krmilnikov procesnih naprav. PLC je dejansko digitalni računalnik za potrebe avtomatizacije elektro-mehanskih procesov. Za razliko od splošno-namenskih računalnikov je bolj odporen na klimatske spremembe, elektro-magnetne vplive, vibracije in prah. PLC operira z vhodno/izhodnimi signali, prek katerih je povezan s perifernimi napravami (pogoni, senzorji, frekvenčnimi regulatorji, čitalci črtnih kod in drugimi) [3].

PLC lahko upravlja eno ali več naprav. Tipično imajo bolj kompleksne naprave (ARD, vozički) lasten PLC, medtem ko si množica elementov transportnega sistema deli skupnega.

2 Zasnova

Pri snovanju WCS sem imel v mislih sistem, ki bo karseda prilagodljiv, razširljiv (ang. scalable) in neodvisen od topologije skladišča ter temelječ na modernih tehnologijah in najboljših praksah (ang. best practice). Poleg tega sem si zadal tudi, da mora biti sistem sposoben podpirati večjezičnost.

Prilagodljivost sem dosegel z modularno zgradbo sistema (Slika 5).



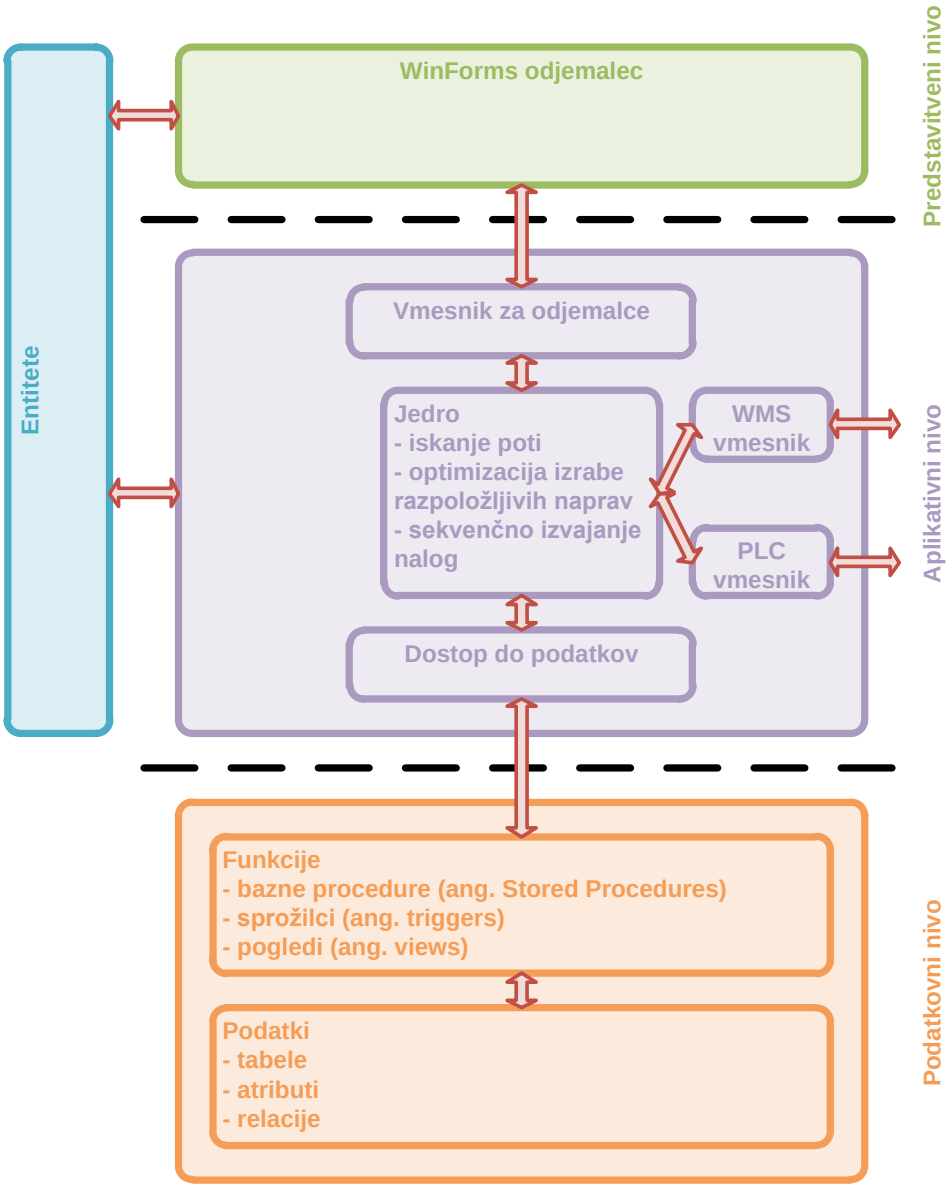
Slika 5: Moduli WCS

Modularna zasnova omogoča prilagoditev posameznega modula zunanjemu sistemu brez vpliva na jedro. To pomeni, da izvorne kode jedra ni potrebno spreminjati v kolikor se spremeni kateri od zunanjih sistemov, saj vmesniki med jedrom in posameznimi moduli ostanejo nespremenjeni. Modul WMS skrbi za komunikacijo s sistemom WMS, modul PLC za komunikacijo s PLC-ji, modul DB za dostop do podatkovne baze ter modul Client za interakcijo z odjemalci.

Razširljivost je zagotovljena z možnostjo naknadnega prenosa posameznih modulov na fizično ločene strežnike, neodvisnost od topologije skladišča pa s tem, da sem vse topološke parametre definiral na nivoju podatkovne baze ter izdelal vnosne maske za spreminjanje le-teh.

2.1 Arhitektura

Modularni zasnovi analogno ustreza t. i. n-nivojska (ang. n-tier) arhitektura. N-nivojska arhitektura se v grobem deli na tri nivoje. Podatkovnega, aplikativnega in predstavitvenega, kateri se lahko nadalje delijo na več podnivojev. Slika 6 prikazuje n-nivojsko arhitekturo, ki sem si jo zamislil za sistem WCS.



Slika 6: Arhitektura sistema WCS

Podatkovni nivo je v bistvu podatkovna baza s pripadajočimi tabelami, relacijami in baznimi procedurami in je implementiran v okviru podatkovnega strežnika. Aplikativni nivo združuje poslovno logiko, vmesnike do zunanjih sistemov in entitete, ki so objektna predstavitev

podatkovne baze. Predstavitveni nivo služi interakciji uporabnika s sistemom ter osnovni validaciji vnešenih podatkov. Predstavitveni nivo nikoli direktno ne dostopa do podatkovnega nivoja.

Predstavljena arhitektura v odvisnosti od potreb in kompleksnosti skladišča omogoča izvedbo na enem računalniku z enim odjemalcem, lahko pa sta podatkovni in aplikativni nivo izvedena na namenskih strežnikih, poljubno število odjemalcev pa na posameznih računalnikih. Poleg tega je arhitektura odprta za morebitne kasnejše razširitve v smislu še večje porazdeljenosti.

2.2 Primeri uporabe

Kljub temu, da sistem tipično deluje avtonomno, so zastoji naprav običajni in pričakovani dogodki. V teh primerih mora sistem uporabniku nuditi ustrezne funkcije za odpravo zastojev in morebitnih nekonsistentnosti med fizičnim in logičnim stanjem v skladišču.

Tipični primer zastoja je prekinitev senzorjev za nadzor velikosti skladiščne enote. Med gibanjem dvigala lahko pride do premikov materiala na skladiščni enoti zaradi česar se senzor prekine in avtomatika ustavi delovanje naprave, saj obstaja nevarnost strojeloma ali poškodovanja materiala. V tem primeru tipično pooblaščen vzdrževalni delavec odpravi fizično napako, uporabnik sistema pa mora zagotoviti konsistentnost med fizičnim in logičnim stanjem.

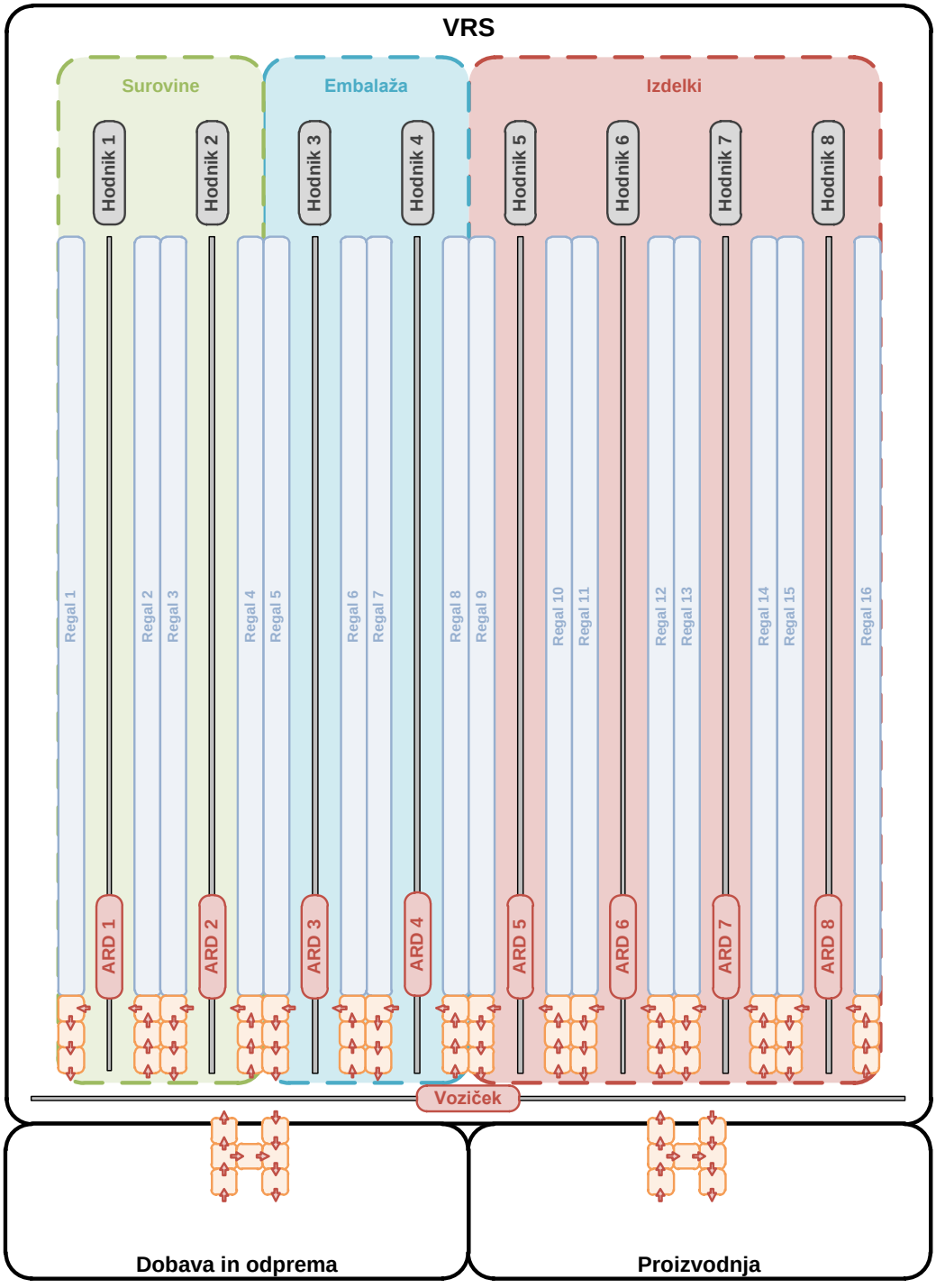
Slika 7 prikazuje diagram primerov uporabe. Nivoja WMS in PLC tudi nastopata kot kreatorja novih nalog. Nivo PLC kadar je nova skladiščna enota identificirana na identifikacijskem mestu, nivo WMS pa kadar zahteva nov premik (izskladiščenje, uskladiščenje ali preskladiščenje). Uporabnik v vlogi kreiranja nove naloge nastopa izjemoma, ker je to ročni poseg v sistem in tipično ni potreben. Preostali primeri uporabe so v domeni uporabnikov za potrebe odpravljanja modebitnih težav ter konfiguracije sistema.

Slika prikazuje privzete primere uporabe, saj so uporabniške skupine in z njimi povezane vloge konfigurabilne.

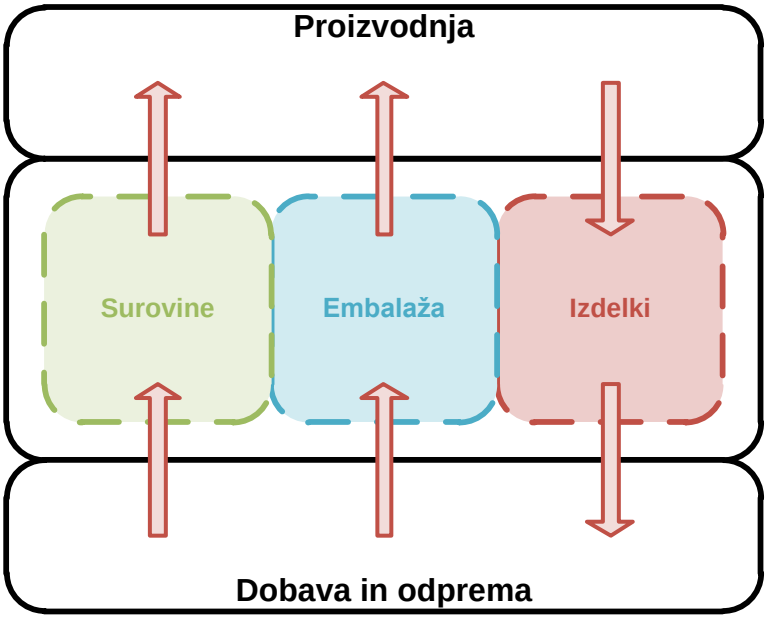
Slika 7: Diagram primerov uporabe

2.3 Skladišče

Pri implementaciji sistema sem si pomagal z namišljenim primerom skladišča, ki ga prikazuje Slika 8. Primer vsebuje nekaj tipičnih situacij, ki se pojavljajo v realnih sistemih in služi kot primerna osnova za razvoj z vidika kompleksnosti. Slika 9 prikazuje isto skladišče z vidika sistema WMS.



Slika 8: Topologija VRS z vidika sistema WCS

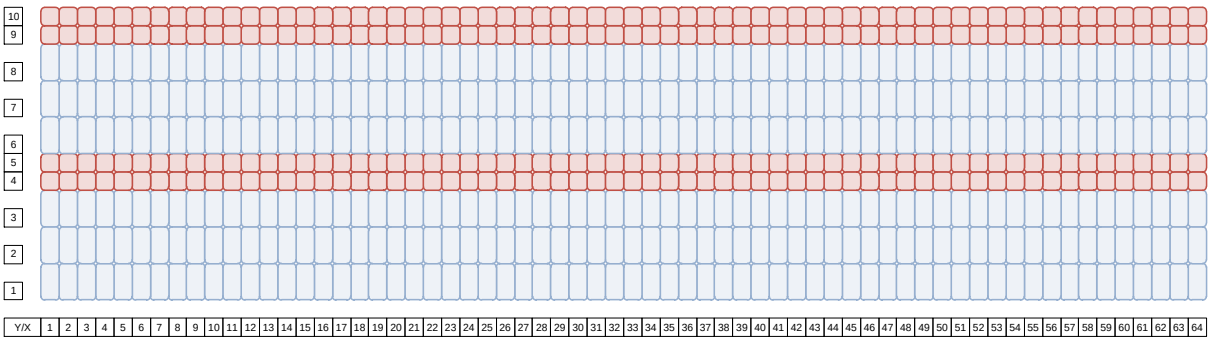


Slika 9: Topologija VRS z vidika sistema WMS

VRS fizično sestavlja 8 regalnih hodnikov. Vsakemu hodniku pripadata dva regala in ARD. Hodnike in dve vhodno-izhodni transportni liniji povezuje voziček.

VRS logično sestavlja 5 področij:

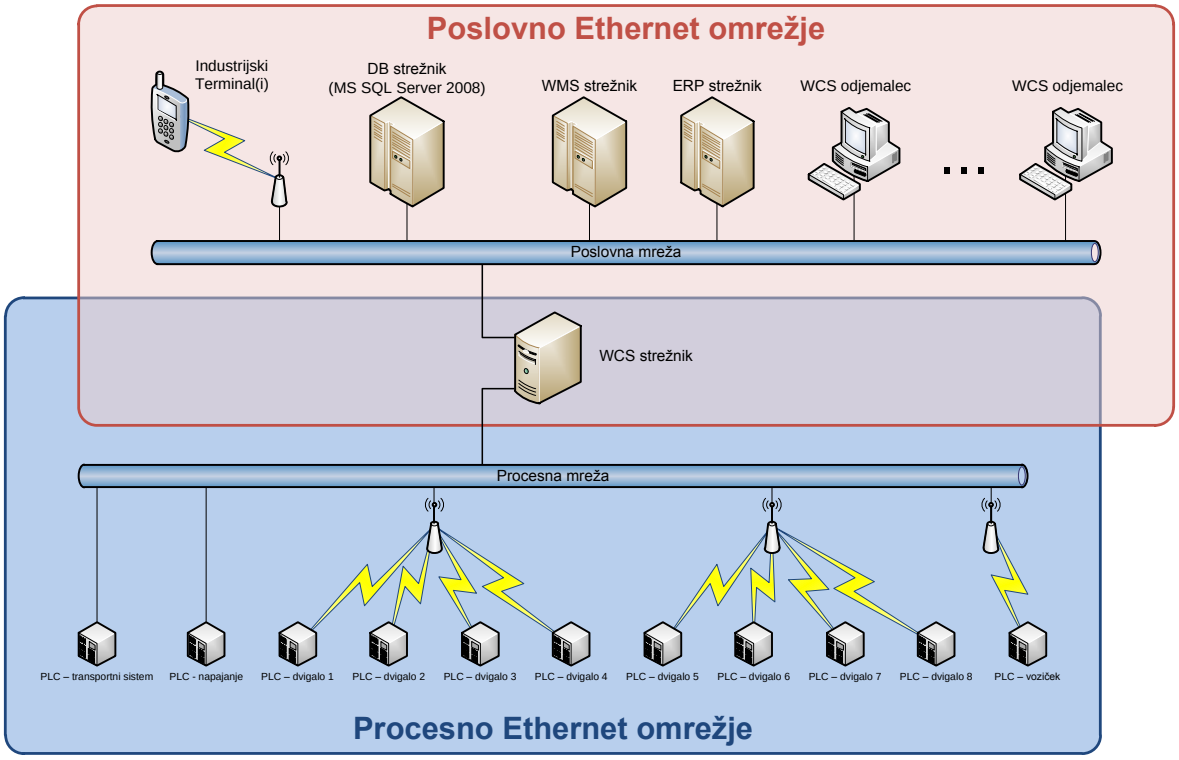
- surovine,
- embalaža,
- izdelki,
- dobava in odprema ter
- proizvodnja.



Slika 10: Tabela skladiščnih mest enega regala

Vsako skladiščno mesto je enolično definirano z oznako modula, številko regala ter koordinatami x, y in z (Slika 10). Enolična oznaka poljubnega skladiščnega mesta konkretnega skladišča je torej lahko zapisana kot **VRS.RR.XX.YY.Z**, pri čemer lahko spremenljivka RR v konkretnem primeru zavzame vrednost od 1 do 16, XX od 1 do 64, YY od 1 do 10 in Z 1. Poleg tega tabela skladiščnih mest definira dve različni višini – enojna in dvojna. Skladiščna mesta enojne višine imajo enojno nosilnost, skladiščna mesta dvojne višine pa dvojno nosilnost. To so za WCS pomembni atributi, saj mora skladiščno mesto po velikosti in nosilnosti ustrezati velikosti in teži skladiščne enote.

Strežnik sistema WCS je tipično povezan v dve Ethernet omrežji – poslovno in procesno (Slika 11). Preko procesnega Ethernet omrežja WCS komunicira s PLC nivojem, preko poslovnega pa z ostalimi nivoji, podatkovno bazo ter odjemalci.



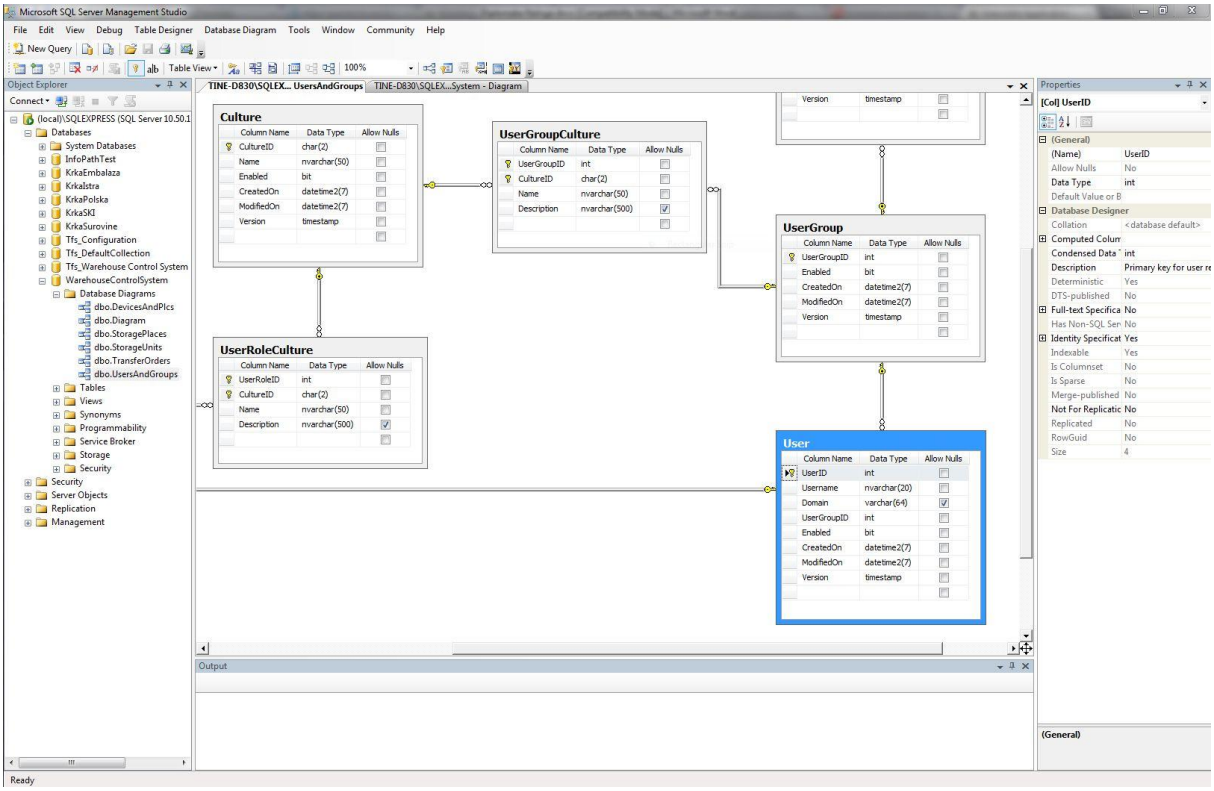
Slika 11: Shema mrežnih povezav

3 Implementacija

Posamezne elemente sistema sem izdeloval v vrstnem redu od zunaj navznoter. To pomeni, da sem najprej izdelal podatkovni nivo, nato predstavitveni nivo ter nazadnje aplikativni nivo s pripadajočimi vmesniki do zunanjih sistemov.

3.1 Uporabljena orodja in tehnologije

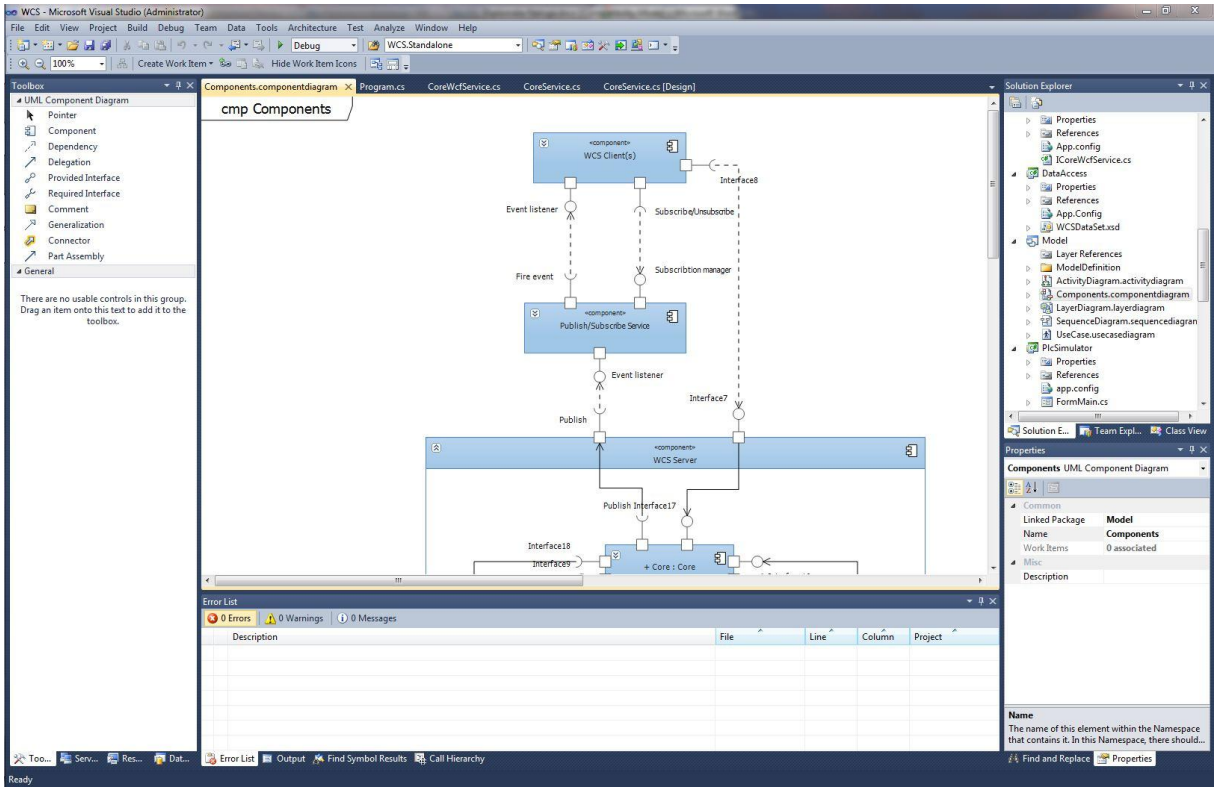
Pri izdelavi diplomskega dela sem uporabil izključno orodja in tehnologije podjetja Microsoft. Za zasnovo in izdelavo podatkovne baze sem uporabil orodje *SQL Server 2008* s pripadajočim IDE *SQL Server Management Studio*. Omenjeno orodje omogoča enostavno grafično izdelavo tabel (entitet), stolpcev (atributov) in relacij, kot prikazuje Slika 12.



Slika 12: SQL Server Management Studio

Pri arhitekturni zasnovi sistema, kakor tudi implementaciji, sem si pomagal z IDE *Visual Studio 2010 Ultimate* (Slika 13), ki omogoča celovit razvoj programske opreme vključno z izdelavo diagramov UML. Moja celotna rešitev temelji na programskem ogrodju *.NET Framework 4.0*, posebno pozornost pa sem namenil ogrodju WCF (ang. Windows

Communication Foundation). Omenjena tehnologija sicer ni novost .NET 4.0, saj je del ogrodja .NET od vključno verzije 3.0 naprej.



Slika 13: Visual Studio 2010 Ultimate

WCF je API za razvoj povezanih, storitveno usmerjenih aplikacij. Oblikovan je v skladu z načeli storitveno usmerjene arhitekture (ang. Service-Oriented Architecture) za podporo porazdeljenemu izvajanju, kjer odjemalci koristijo storitve. Odjemalec lahko koristi različne storitve in storitev lahko služi različnim odjemalcem, storitve pa so med seboj ohlapno povezane.

Storitev je v osnovi sestavljena iz treh delov:

- tipa storitve – implementacije ponujane storitve,
- gostitelja – procesa, ki gosti storitev in
- ene ali več končnih točk (ang. endpoint) zadolženih za komunikacijo z odjemalci.

Tip storitve je lahko katerikoli CLR tip, ki je ustrezno opremljen z atributi (Slika 14).

```
[ServiceContract]
public class PersonWCF
{
   private Person person;

   [OperationContract]
   Person GetPerson()
   {
      return this.person;
   }

   [OperationContract]
   void SetPerson(Person person)
   {
      this.person = person;
   }
}

[DataContract]
public class Person
{
   private string firstName;
   private string lastName;

   [DataMember]
   public string FirstName
   {
      get { return firstName; }
      set { firstName = value; }
   }

   [DataMember]
   public string LastName
   {
      get { return lastName; }
      set { lastName = value; }
   }
}
```

Slika 14: Primer deklaracije tipa WCF

Vsi tipi storitve WCF morajo biti označeni z atributom `ServiceContract`, pripadajoče metode pa z atributom `OperationContract`. Morebitni tipi po meri, ki so predmet podatkovnega prenosa storitve WCF morajo biti označeni z atributom `DataContract`, pripadajoče lastnosti pa z `DataMember`.

Gostitelj je lahko katerikoli proces okolja Windows, lahko pa gostovanje prepustimo tudi infrastrukturi spletnega strežnika *IIS* oziroma v okoljih Windows Vista/7 storitvi Windows Activation Service.

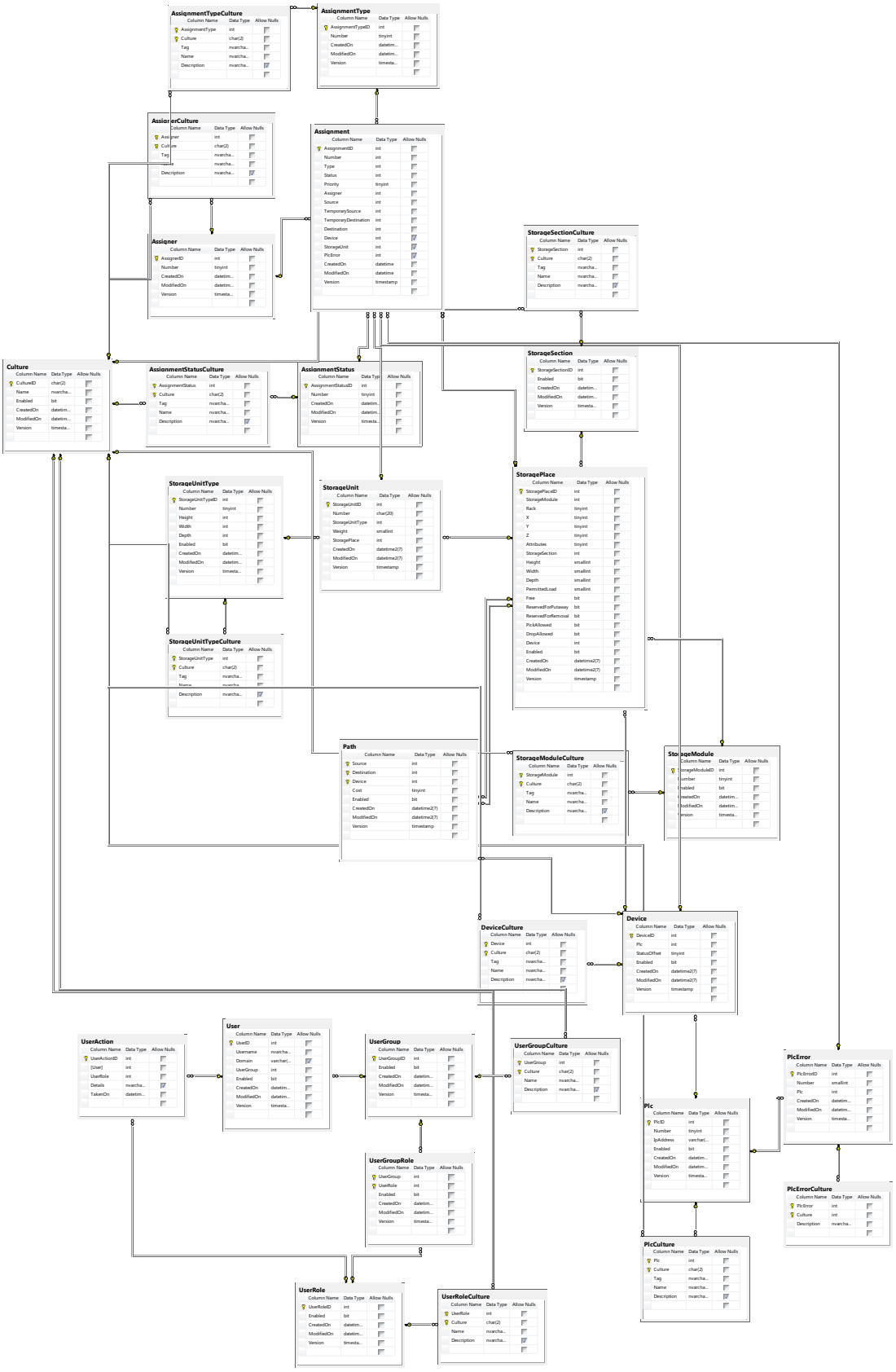
Odjemalec storitve WCF je lahko Windows aplikacija, spletna stran, ali kar druga storitev.

Sporočila, ki si jih izmenjujejo storitve in odjemalci so SOAP (ang. Simple Object Access Protocol) sporočila. Sporočila niso odvisna od transportnega protokola. Odjemalci WCS lahko koristijo storitve, ki niso WCS in obratno [4,5].

3.2 Podatkovni nivo

Podatkovna baza je temelj n-nivojskega sistema. Morebitne spremembe strukture podatkovne baze tipično pomenijo velike spremembe ostalih nivojev, zaradi tega sem temu elementu namenil posebno pozornost in posledično tudi sorazmerno največ časa. H kompleksnosti je še dodatno pripomogla odločitev, da zadostim potrebam večjezičnosti.

Jedro podatkovne baze tvorijo tabele skladiščno mesto (StoragePlace), skladiščna enota (StorageUnit) in aktivna naloga (ActiveAssignment). Omenjene tabele sem v postopku izdelave strukture podatkovne baze ustrezno normaliziral. Dodal sem tudi tabele za potrebe upravljanja z uporabniki. Vsak uporabnik pripada eni uporabniški skupini, posamezni uporabniški skupini pa je dodeljena množica uporabniških vlog. Vsaki funkcionalnosti sistema, ki je dostopna prek uporabniškega vmesnika, je dodeljena ena uporabniška vloga. S tem je zagotovljena zelo detajlna konfiguracija pooblastil posameznim uporabnikom. Slika 15 prikazuje entitetno-relacijski model podatkovne baze.



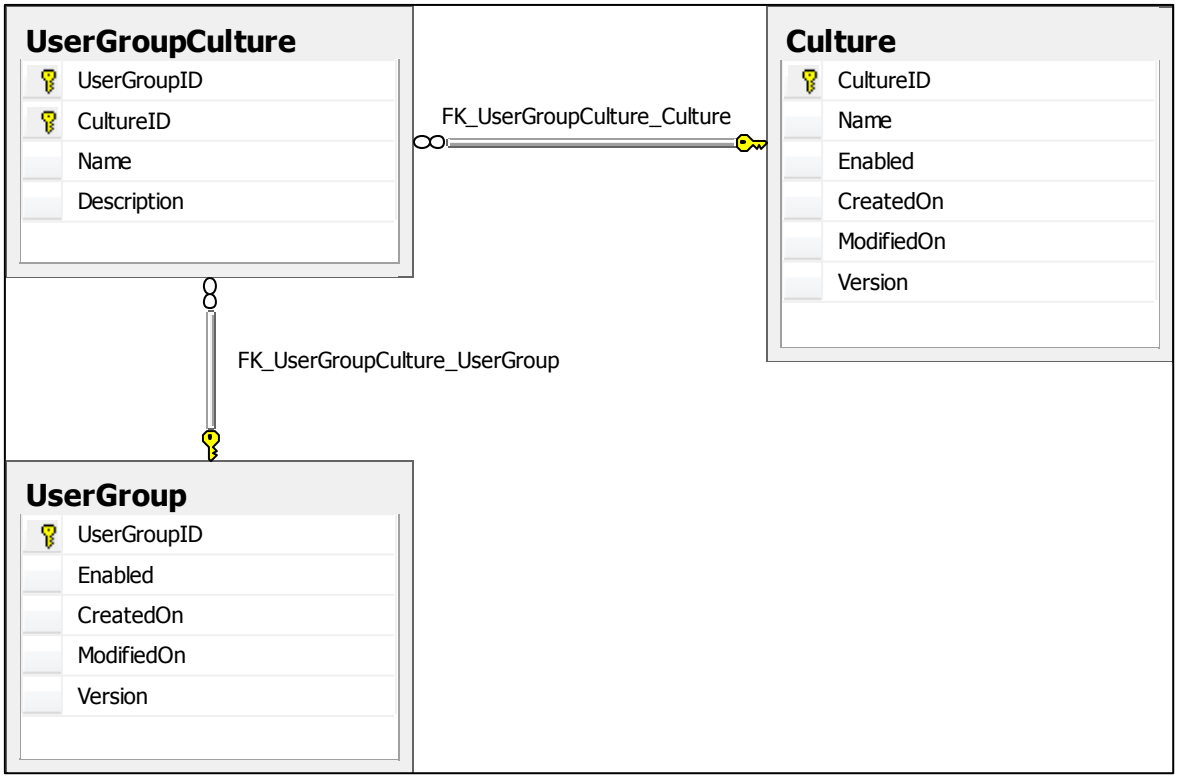
Slika 15: Entitetno-relacijski model

Primarni ključ vseh osnovnih tabel je identiteta, torej število, ki se samodejno povečuje za vrednost ena ob vsakem kreiranju novega zapisa in ni nosilec kakršnekoli informacije. Poleg tega je skupna lastnost vseh tabel tudi atribut *Version* tipa *rowversion*, ki je potreben za zagotavljanje integritete podatkov ob sočasnem dostopanju do podatkovne baze. Vsaka sprememba zapisa namreč povzroči spremembo atributa *Version* kar omogoča enostavno preverjanje ažurnih podatkov pred spreminjanjem.

Ker primarni ključ ni nosilec informacije, je enoličnost, integriteta in ustreznost podatkov zagotovljena z integritetnimi omejitvami in omejitvami po ključu. Tabela *StorageUnit* ima npr. omejitev po ključu pri katerem mora biti kombinacija atributov posameznih koordinat (*Module, Rack, X, Y in Z*) enolična.

3.2.1 Večjezičnost

Zaradi potrebe po večjezični podpori sem vpeljal entiteto *Culture*, ki vsebuje uporabljene jezike. Vse entitete z atributi, ki bi lahko nastopali v več jezikih sem predelal v dve entiteti. Starševska entiteta vsebuje atribute, katerih prevajanje ni potrebno, otroška entiteta pa vsebuje atribute, ki so lahko predmet prevajanja. Primarni ključ otroške entitete je sestavljen iz primarnih ključev starševske entitete in entitete *Culture*. Slika 16 prikazuje primer te rešitve.



Slika 16: Primer pomožne entitete za potrebe večjezičnosti

3.2.2 Dostop do podatkov na nivoju podatkovne baze

Z vidika zunanjega sistema je dostop do podatkov mogoč samo z uporabo baznih procedur (ang. stored procedure). Bazne procedure imajo vrsto prednosti pred ad-hoc poizvedbami SQL:

- največja možna hitrost izvajanja,
- koda SQL je samo na podatkovnem strežniku,
- varnost pred napadi z vrinjeno kodo SQL,
- onemogočeno siceršnje poizvedovanje in spreminjanje podatkov.

Zaradi tega sem za vsako posamezno tabelo izdelal bazne procedure za osnovne štiri operacije (izbiranje, vnašanje, spreminjanje in brisanje) ter v nekaterih primerih še dodatne, kot so izbiranje glede na glavni ključ, izbiranje glede na poljubni filter in podobno. Slika 17 prikazuje primer bazne procedure za izbiranje. V tem primeru gre za izbiranje vseh zapisov tabele.

```
CREATE PROCEDURE [dbo].[SelectUser]
AS

	SET NOCOUNT ON

	SELECT
		[UserID],
		[Username],
		[Domain],
		[UserGroupID],
		[Enabled],
		[CreatedOn],
		[ModifiedOn],
		[Version]
	FROM
		[dbo].[User]

	RETURN @@ROWCOUNT
```

Slika 17: Bazna procedura za izbiranje

Bazna procedura za vstavljanje (Slika 18) je malenkost bolj kompleksna. Procedura kreira nov zapis z danimi parametri. V kolikor vpisovanje ni uspešno, se procedura prekine, sicer pa osveži avtomatično generirane vrednosti (*ID*, *CreatedOn*, *ModifiedOn* in *Version*). Na ta način je zagotovljena ažurnost podatkov na strani procesa, ki je sprožil proceduro ter minimiziran podatkovni promet med podatkovnim strežnikom in odjemalcem.

```
CREATE PROCEDURE [dbo].[InsertUser]
(
	@UserID int OUTPUT,
	@Username nvarchar(20),
	@Domain varchar(64),
	@UserGroupID int,
	@Enabled bit,
	@CreatedOn datetime2(7) OUTPUT,
	@ModifiedOn datetime2(7) OUTPUT,
	@Version timestamp OUTPUT
)
AS

	SET NOCOUNT ON

	INSERT INTO [dbo].[User] (
		[Username],
		[Domain],
		[UserGroupID],
		[Enabled],
		[CreatedOn],
		[ModifiedOn])
	VALUES (
		@Username,
		@Domain,
		@UserGroupID,
		@Enabled,
		getdate(),
		getdate())

	IF @@ERROR <> 0 RETURN 0

	SET @UserID = SCOPE_IDENTITY()

	SELECT
		@CreatedOn = [CreatedOn],
		@ModifiedOn = [ModifiedOn],
		@Version = [Version]
	FROM
		[dbo].[User]
	WHERE
		[UserID] = @UserID

	RETURN @UserID
```

Slika 18: Bazna procedura za vstavljanje

Procedura za spreminjanje (Slika 19) je sorodna proceduri za vstavljanje. Procedura spremeni posamezen zapis glede na primarni ključ in verzijo zapisa, kar onemogoča spreminjanje zapisa, ki je bil morda spremenjen s strani drugega odjemalca od zadnje izbire. Poleg tega procedura spremeni samo tiste atribute, katerih argumenti so definirani (funkcija COALESCE). Morebitne izračunane vrednosti atributov so osvežene (v tem primeru *ModifiedOn* in *Version*).

```
CREATE PROCEDURE [dbo].[UpdateUser]
(
	@UserID int,
	@Version timestamp,
	@newUsername nvarchar(20),
	@newDomain varchar(64),
	@newUserGroupID int,
	@newEnabled bit,
	@newModifiedOn datetime2(7) OUTPUT,
	@newVersion timestamp OUTPUT
)
AS

	SET NOCOUNT ON

	UPDATE
		[dbo].[User]
	SET
		[Username] = COALESCE(@newUsername, [Username]),
		[Domain] = COALESCE(@newDomain, [Domain]),
		[UserGroupID] = COALESCE(@newUserGroupID, [UserGroupID]),
		[Enabled] = COALESCE(@newEnabled, [Enabled]),
		[ModifiedOn] = getdate()
	WHERE
		([UserID] = @UserID) AND
		([Version] = @Version)

	IF @@ERROR <> 0 RETURN 0

	SELECT
		@newModifiedOn = [ModifiedOn],
		@newVersion = [Version]
	FROM
		[dbo].[User]
	WHERE
		[UserID] = @UserID

	RETURN 1
```

Slika 19: Bazna procedura za spreminjanje

Procedura za brisanje (Slika 20) je najenostavnejša. Briše zapis glede na vrednost primarnega ključa in verzijo, kar preprečuje brisanje zapisov, ki so bili spremenjeni s strani drugega odjemalca od zadnje izbire.

```
CREATE PROCEDURE [dbo].[DeleteUser]
(
	@UserID int,
	@Version timestamp
)
AS

	SET NOCOUNT ON

	DELETE FROM
		[dbo].[User]
	WHERE
		([UserID] = @UserID) AND
		([Version] = @Version)

	IF @@ERROR <> 0
		RETURN 0
	ELSE
		RETURN @@ROWCOUNT
```

Slika 20: Bazna procedura za brisanje

Tabele, katerim so pridružene pomožne tabele za potrebe večjezične podpore imajo nekoliko bolj zapletene bazne procedure, saj sem želel, da navzven ti pari tabel delujejo kot ena tabela. Bazna procedura za izbiranje (Slika 21) v tem primeru vrača atribute treh tabel – nadrejene (v tem primeru *UserGroup*), podrejene (v tem primeru *UserGroupCulture*) ter tabele uporabljenih jezikov (*Culture*). Nad temi tabelami se izvede notranji stik in filtriranje glede na izbrani jezik.

```
CREATE PROCEDURE [dbo].[SelectUserGroup]
(
	@CultureID char(2)
)
AS

	SET NOCOUNT ON

	SELECT
		[UserGroup].[UserGroupID],
		[Culture].[CultureID],
		[UserGroupCulture].[Name],
		[UserGroupCulture].[Description],
		[UserGroup].[Enabled],
		[UserGroup].[CreatedOn],
		[UserGroup].[ModifiedOn],
		[UserGroup].[Version]
	FROM
		[dbo].[UserGroup]
	INNER JOIN
		[dbo].[UserGroupCulture]	ON	[UserGroup].[UserGroupID]	=
[UserGroupCulture].[UserGroupID]
	INNER JOIN
		[dbo].[Culture] ON [UserGroupCulture].[CultureID] = [Culture].[CultureID]
	WHERE
		([Culture].[CultureID] = @CultureID)

	RETURN @@ROWCOUNT
```

Slika 21: Izbiranje pri tabelah z večjezično podporo

Pri proceduri za vstavljanje (Slika 22) sem bil primoran uporabiti transakcije, saj je potrebno vpisovati tako v nadrejeno, kot tudi v podrejeno tabelo. Najprej se izvede vpis v nadrejeno tabelo, nato pa se uporabi ravnokar generirani ključ nadrejene tabele in se v podrejeno tabelo vpiše zapise glede na tabelo uporabljenih jezikov. Na ta način sta obe tabeli usklajeni in integriteta podatkov zagotovljena. Če transakcija ni uspešna, se spremembe razveljavijo.

```
CREATE PROCEDURE [dbo].[InsertUserGroup]
(
	@UserGroupID int OUTPUT,
	@Enabled bit,
	@Name nvarchar(50),
	@Description nvarchar(500),
	@CreatedOn datetime2(7) OUTPUT,
	@ModifiedOn datetime2(7) OUTPUT,
	@Version timestamp OUTPUT
)
AS

	SET NOCOUNT ON

	BEGIN TRANSACTION

	INSERT INTO [dbo].[UserGroup] (
		[Enabled],
		[CreatedOn],
		[ModifiedOn])
	VALUES (
		@Enabled,
		getdate(),
		getdate())

	IF @@ERROR <> 0
	BEGIN
		ROLLBACK TRANSACTION
		RETURN 0
	END

	SET @UserGroupID = SCOPE_IDENTITY()

	INSERT INTO [dbo].[UserGroupCulture]
		SELECT
			@UserGroupID,
			[dbo].[Culture].[CultureID],
			@Name,
			@Description
		FROM
			[dbo].[Culture]

	IF @@ERROR <> 0
	BEGIN
		ROLLBACK TRANSACTION
		RETURN 0
	END

	COMMIT TRANSACTION

	SELECT
		@CreatedOn = [CreatedOn],
		@ModifiedOn = [ModifiedOn],
		@Version = [Version]
	FROM
		[dbo].[UserGroup]
	WHERE
		[UserGroupID] = @UserGroupID

	RETURN @UserGroupID
```

Slika 22: Vstavljanje pri tabelah z večjezično podporo

Spreminjanje tabel (Slika 23) je prav tako urejeno z uporabo transakcije, brisanje (Slika 24) pa je zelo poenostavljeno, saj je potrebno brisati samo zapis v nadrejeni tabeli, relacija pa poskrbi za kaskadno brisanje ustreznih zapisov iz podrejene tabele.

```
CREATE PROCEDURE [dbo].[UpdateUserGroup]
(
	@UserGroupID int,
	@Version timestamp,
	@CultureID char(2),
	@newEnabled bit,
	@newName nvarchar(50),
	@newDescription nvarchar(500),
	@newModifiedOn datetime2(7) OUTPUT,
	@newVersion timestamp OUTPUT
)
AS

	SET NOCOUNT ON

	BEGIN TRANSACTION

	UPDATE
		[dbo].[UserGroupCulture]
	SET
		[Name] = COALESCE(@newName, [Name]),
		[Description] = COALESCE(@newDescription, [Description])
	WHERE
		([UserGroupID] = @UserGroupID) AND
		([CultureID] = @CultureID)

	IF @@ERROR <> 0
	BEGIN
		ROLLBACK TRANSACTION
		RETURN 0
	END

	UPDATE
		[dbo].[UserGroup]
	SET
		[Enabled] = COALESCE(@newEnabled, [Enabled]),
		[ModifiedOn] = getdate()
	WHERE
		([UserGroupID] = @UserGroupID) AND
		([Version] = @Version)

	IF @@ERROR <> 0
	BEGIN
		ROLLBACK TRANSACTION
		RETURN 0
	END

	COMMIT TRANSACTION

	SELECT
		@newModifiedOn = [ModifiedOn],
		@newVersion = [Version]
	FROM
		[dbo].[UserGroup]
	WHERE
		[UserGroupID] = @UserGroupID

	RETURN 1
```

Slika 23: Spreminjanje pri tabelah z večjezično podporo

```
CREATE PROCEDURE [dbo].[DeleteUserGroup]
(
	@UserGroupID int,
	@Version timestamp
)
AS

	SET NOCOUNT OFF

	DELETE FROM
		[dbo].[UserGroup]
	WHERE
		([UserGroupID] = @UserGroupID) AND
		([Version] = @Version)

	IF @@ERROR <> 0
		RETURN 0
	ELSE
		RETURN @@ROWCOUNT
```

Slika 24: Brisanje pri tabelah z večjezično podporo

Slika 25 prikazuje primer izbiranja zapisov tabele *StoragePlace* glede na poljubni filter.

```
CREATE PROCEDURE [dbo].[SelectStoragePlaceByFilter]
(
	@storageModule int,
	@rackFrom tinyint,
	@rackTo tinyint,
	@xFrom tinyint,
	@xTo tinyint,
	@yFrom tinyint,
	@yTo tinyint,
	@zFrom tinyint,
	@zTo tinyint,
	@attributes tinyint,
	@storageSection int,
	@heightFrom smallint,
	@heightTo smallint,
	@widthFrom smallint,
	@widthTo smallint,
	@depthFrom smallint,
	@depthTo smallint,
	@permittedLoadFrom smallint,
	@permittedLoadTo smallint,
	@free bit,
	@reservedForPutaway bit,
	@reservedForRemoval bit,
	@pickAllowed bit,
	@dropAllowed bit,
	@device int,
	@enabled bit,
	@createdOnFrom datetime2(7),
	@createdOnTo datetime2(7),
	@modifiedOnFrom datetime2(7),
	@modifiedOnTo datetime2(7)
)
AS

	SET NOCOUNT ON

	SELECT
		[StoragePlaceID],
		[StorageModuleID],
		[Rack],
		[X],
		[Y],
		[Z],
		[Attributes],
		[StorageSectionID],
		[Height],
```

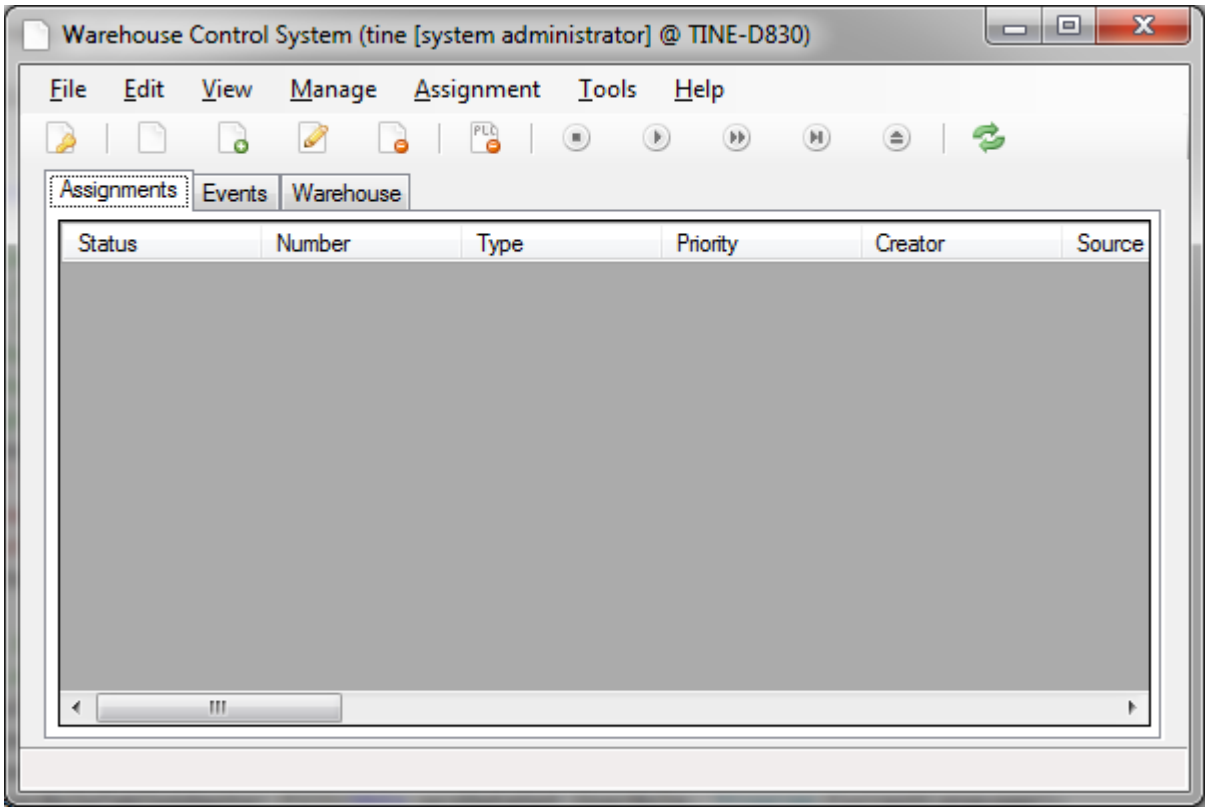
```
                [Width],
                [Depth],
                [PermittedLoad],
                [Free],
                [ReservedForPutaway],
                [ReservedForRemoval],
                [PickAllowed],
                [DropAllowed],
                [DeviceID],
                [Enabled],
                [CreatedOn],
                [ModifiedOn],
                [Version]
        FROM
                [dbo].[StoragePlace]
        WHERE
                ((@storageModule IS NULL) OR ([StorageModuleID] = @storageModule)) AND
                ([Rack] BETWEEN COALESCE (@rackFrom, 0) AND COALESCE (@rackTo, 255)) AND
                ([X] BETWEEN COALESCE (@xFrom, 0) AND COALESCE (@xTo, 255)) AND
                ([Y] BETWEEN COALESCE (@yFrom, 0) AND COALESCE (@yTo, 255)) AND
                ([Z] BETWEEN COALESCE (@zFrom, 0) AND COALESCE (@zTo, 255)) AND
                ((@attributes IS NULL) OR ([Attributes] = @attributes)) AND
                ((@storageSection IS NULL) OR ([StorageSectionID] = @storageSection)) AND
                ([Height] BETWEEN COALESCE (@heightFrom, 0) AND COALESCE (@heightTo, 32767))
AND
                ([Width] BETWEEN COALESCE (@widthFrom, 0) AND COALESCE (@widthTo, 32767)) AND
                ([Depth] BETWEEN COALESCE (@depthFrom, 0) AND COALESCE (@depthTo, 32767)) AND
                ([PermittedLoad] BETWEEN COALESCE (@permittedLoadFrom, 0) AND COALESCE
(@permittedLoadTo, 2147483647)) AND
                ((@free IS NULL) OR ([Free] = @free)) AND
                ((@reservedForPutaway IS NULL) OR ([ReservedForPutaway] =
@reservedForPutaway)) AND
                ((@reservedForRemoval IS NULL) OR ([ReservedForRemoval] =
@reservedForRemoval)) AND
                ((@pickAllowed IS NULL) OR ([PickAllowed] = @pickAllowed)) AND
                ((@dropAllowed IS NULL) OR ([DropAllowed] = @dropAllowed)) AND
                ((@device IS NULL) OR ([DeviceID] = @device)) AND
                ((@enabled IS NULL) OR ([Enabled] = @enabled)) AND
                ([CreatedOn] BETWEEN COALESCE (@createdOnFrom, '') AND COALESCE (@createdOnTo,
getdate())) AND
                ([ModifiedOn] BETWEEN COALESCE (@modifiedOnFrom, '') AND COALESCE
(@modifiedOnTo, getdate()))

        RETURN @@ROWCOUNT
```

Slika 25: Primer bazne procedure za izbiranje zapisov glede na poljuben filter

3.3 Predstavitveni nivo

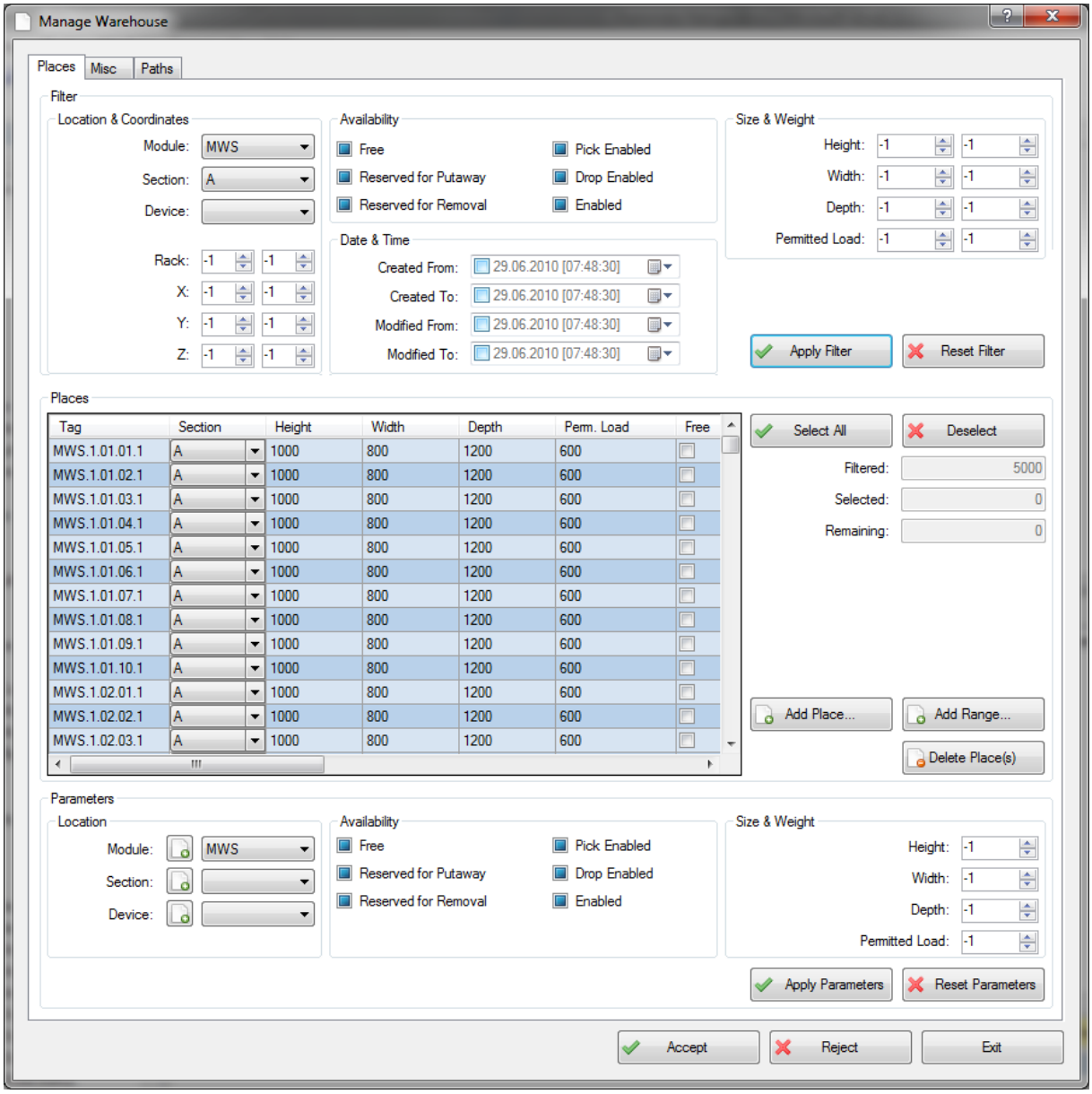
Predstavitveni nivo predstavlja edino vez uporabnika s sistemom. Osnovnim uporabnikom omogoča enostavno in učinkovito delo z nalogami, naprednim uporabnikom pa spreminjanje skladiščnih parametrov, spreminjanje topologije skladišča, upravljanje z uporabniki in uporabniškimi skupinami in podobno.
Odjemalca sem implementiral kot *WinForms* aplikacijo. Osnovno okno (Slika 26) sestavljajo standardni elementi – vrstica z meniji, orodna vrstica, osrednji del in statusna vrstica. Preko menijev lahko uporabniki dostopajo do vseh funkcij sistema, orodna vrstica omogoča hiter dostop do najbolj uporabljanih funkcij za delo z nalogami (nova naloga, brisanje naloge in podobno), osrednji del pa prikazuje seznam nalog, ki so v danem trenutku v obdelavi. Naslovna vrstica prikazuje ime uporabnika, ki je trenutno prijavljen, pripadajočo uporabniško skupino in ime računalnika na katerem se izvaja odjemalec.



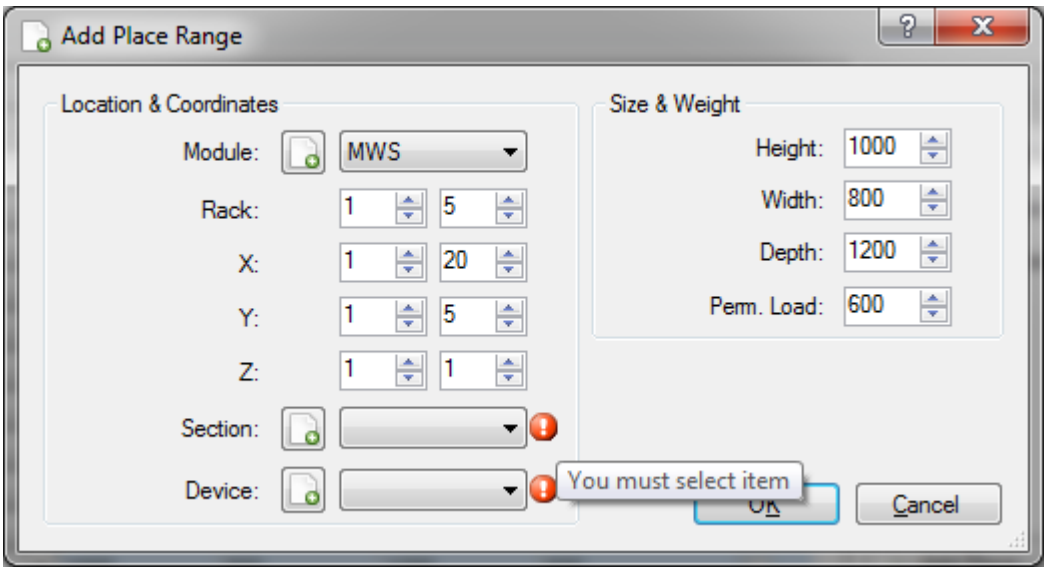
Slika 26: Osnovno okno odjemalca

Slika 27 predstavlja dialog za upravljanje skladiščnih mest in poti, ki je eden bolj kompleksnih, predstavlja pa tipično zaznovo, ki sem jo uporabil pri vseh dialogih. Uporabniku dialog omogoča filtriranje odstoječih zapisov, brisanje in spreminjanje, ter s pomočjo namenskih vnosnih mask (Slika 28) kreiranje novih zapisov. Pri spreminjanju in

kreiranju se na strani predstavitvenega nivoja izvaja osnovna validacija vnešenih podatkov in tudi validacija relacijskih omejitev. Namen kvalitetne validacije je minimizacija nepotrebne komunikacije med posameznimi nivoji.



Slika 27: Dialog za upravljanje skladiščnih mest in poti



Slika 28: Vnosna maska za kreiranje novih skladiščnih mest

3.4 Aplikativni nivo

Aplikativni nivo predstavlja jedro sistema. Implementiral sem ga kot storitev Windows. Združuje vmesnike do zunanjih sistemov in entitete. Dostop do podatkovne baze je izveden s pomočjo podatkovne strukture *DataSet*. Komunikacija s PLC nivojem je izvedena z uporabo TCP/IP vtičnic (ang. socket). Z vidika TCP/IP komunikacije je jedro strežnik in posamezni PLC-ji odjemalci. Za potrebe komunikacije s predstavitvenim nivojem (odjemalci) in sistemom WMS jedro gosti več storitev WCF:

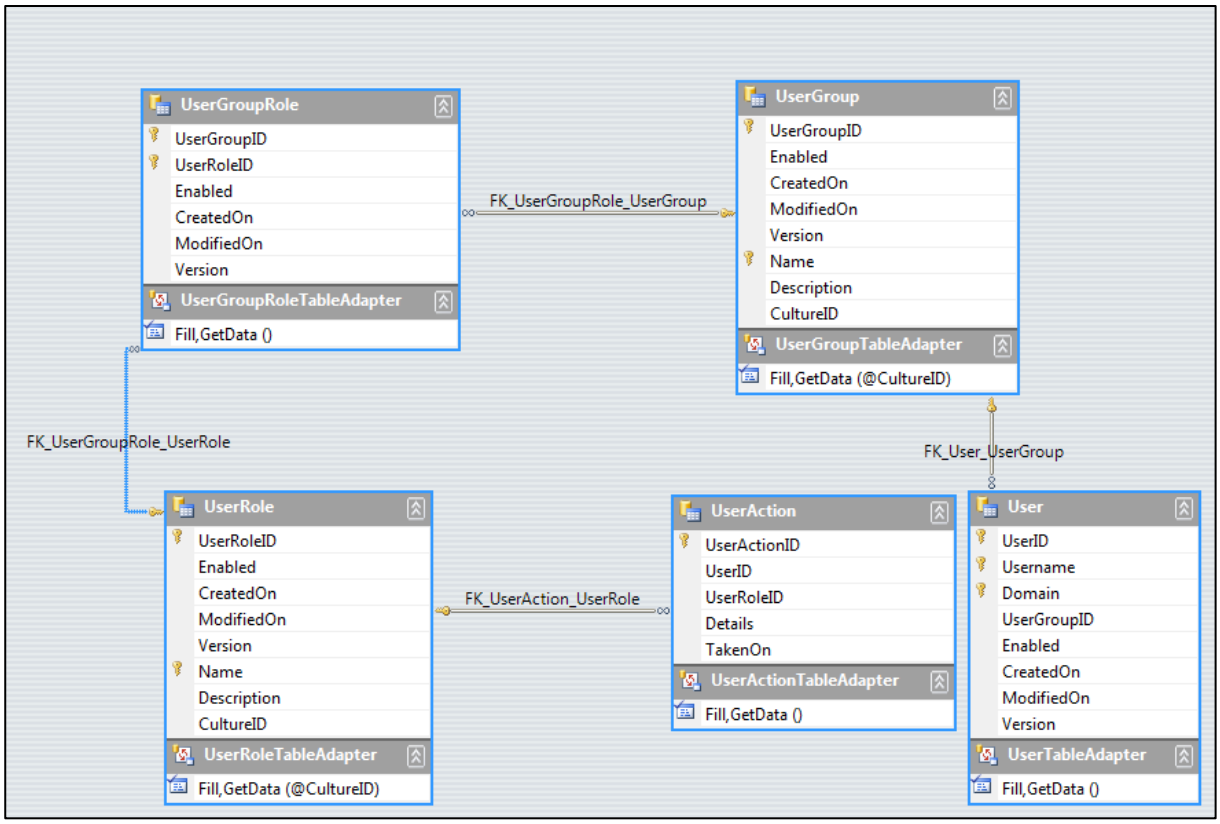
- storitev WCF za klice s strani sistema WMS,
- storitev WCF za klice s strani predstavitvenega nivoja in
- objavi/naroči (ang. publish/subscribe) WCF storitev za potrebe obveščanja odjemalcev o dogodkih.

Jedro na podlagi informacij s strani vmesnikov do zunanjih sistemov v realnem času obdeluje naloge. Vsaka naloga je sestavljena iz delnih nalog, ki se sekvenčno izvedejo. Signal za začetek izvajanja naloge pride bodisi s strani sistema WMS, bodisi s strani PLC nivoja. Za vsako delno nalogo je potrebno na podlagi trenutne razpoložljivosti naprav in definiranih skladiščnih poti poiskati nov ustrezen delni cilj. Skladiščne poti so v podatkovni bazi definirane z izvornim skladiščnim mestom, ciljnim skladiščnim mestom, napravo in ceno. Na podlagi teh parametrov algoritem poišče pot do naslednjega razpoložljivega delnega cilja, ki vodi do končnega cilja. Za iskanje poti sem uporabil algoritem Dijkstra [6].

3.4.1 Dostop do podatkovne baze

Za dostop do podatkovne baze sem uporabil podatkovno strukturo *DataSet* s pripadajočimi *DataTable* strukturami in *TableAdapter* strukture ogrodja ADO.NET.

Poglavitna prednost uporabe *DataSet*-a je relativna preprostost uporabe oziroma konfiguracije, pri čemer je rezultat preslikava strukture podatkovne baze v CLR podatkovne tipe vključujoč relacije in integritetne omejitve. Čarovnik za izdelavo DataSet-a na podlagi entitetno-relacijskega modela podatkovne baze izdela objektno-relacijsko preslikavo (Slika 29). Preslikavo je nato potrebno samo še ustrezno skonfigurirati – definirati klicanje baznih procedur, popraviti relacije in podobno.



Slika 29: Del objektno-relacijske preslikave

DataSet torej predstavlja entitete, *TableAdapter*-ji pa podnivo dostopa do podatkovne baze. Posamezen *TableAdapter* je posrednik med *DataSet*-om in baznimi procedurami.

3.4.2 Komunikacija s PLC nivojem

Komunikacija med PLC-ji in sistemom WCS je izvedena z izmenjavo telegramov internega protokola prek TCP/IP. Sistem WCS je strežnik in PLC-ji so odjemalci. WCS od PLC-jev prejema telegrame s statusi naprav, na podlagi katerih ažurira razpoložljivost naprav, ter telegrame o izvedenih delnih nalogah, na podlagi katerih lahko izračuna novo delno nalogo za ustrezen PLC. Vse metode strežnika so implementirane asinhrono. To pomeni, da klicatelju metode ni potrebno čakati na izvedbo metode, ampak lahko medtem nadaljuje z delom. Na ta način je zagotovljena karseda velika odzivnost sistema. Strežnik je konfiguriran tako, da sprejme samo povezave z naslovov IP, ki so definirani v PLC tabeli podatkovne baze. S tem je onemogočeno sprejemanje nepooblaščenih odjemalcev.

3.4.3 Komunikacija s predstavitvenim nivojem in sistemom WMS

Pri izdelavi komunikacijske infrastrukture med aplikativnim in predstavitvenim nivojem, kakor tudi sistemom WMS, sem uporabil ogrodje WCF. Aplikativni nivo tako gosti tri storitve WCF. Slika 30 prikazuje deklaracijo storitve za klice s strani sistema WMS. Storitev vsebuje samo metodo za kreiranje nove naloge. Argument metode je tipa WmsAssignment in vsebuje oznako izvornega skladiščnega mesta ter identifikacijsko oznako skladiščne enote.

```
[ServiceContract]
public interface IWmsService
{
    [OperationContract]
    void CreateAssignment(WmsAssignment wmsAssignment);
}
```

Slika 30: Deklaracija storitve WCF za klice s strani sistema WMS

Slika 31 prikazuje deklaracijo storitve za klice s strani predstavitvenega nivoja. Storitev vsebuje metode za rokovanje z nalogami in metodo za ažuriranje podatkovne baze.

```
[ServiceContract]
public interface IClientService
{
    [OperationContract]
    void StopAssignment();

    [OperationContract]
    void StartAssignment();

    [OperationContract]
    void CancelAssignment();

    [OperationContract]
    void FinishAssignment();

    //
    // ...
    //

    [OperationContract]
    void UpdateData(WCS.BusinessEntities.WCSDataSet data);
}
```

Slika 31: Deklaracija storitve WCF za klice s strani predstavitvenega nivoja

S storitvijo objavi/naroči [7] sem rešil problem obveščanja odjemalcev o dogodkih. Preko te namenske storitve se odjemalci naročijo na poljubne dogodke, aplikativni nivo pa dogodke objavlja. Slika 32 prikazuje deklaracijo storitve. Odjemalcem so namenjene metode za naročanje in odjavljanje, aplikativnemu nivoju pa metode za proženje dogodkov. S pomočjo te storitve je moč enostavno zagotavljati ažurnost podatkov tako na strani odjemalcev, kot tudi jedra. Ko uporabnik preko odjemalca spremeni podatke, se podatki na strani odjemalca še

ne shranijo, ampak se kliče WCF metoda `UpdateData` s spremembami podatkov. Jedro na podlagi sprememb ažurira podatkovno bazo in nato proži dogodek `OnDataChanged`. Ta dogodek vsebuje dejanske spremembe podatkov in na podlagi tega vsi odjemalci, ki so naročeni na dogodek ažurirajo svoje podatke.

```
[ServiceContract]
public interface IPubSubService
{
   [OperationContract]
   void Subscribe(string eventName);

   [OperationContract]
   void Unsubscribe(string eventName);

   [OperationContract]
   void SubscribeAll();

   [OperationContract]
   void UnsubscribeAll();

   [OperationContract]
   void OnDataChanged(WCS.BusinessEntities.WCSDataSet data);

   //
   // ...
   //
}
```

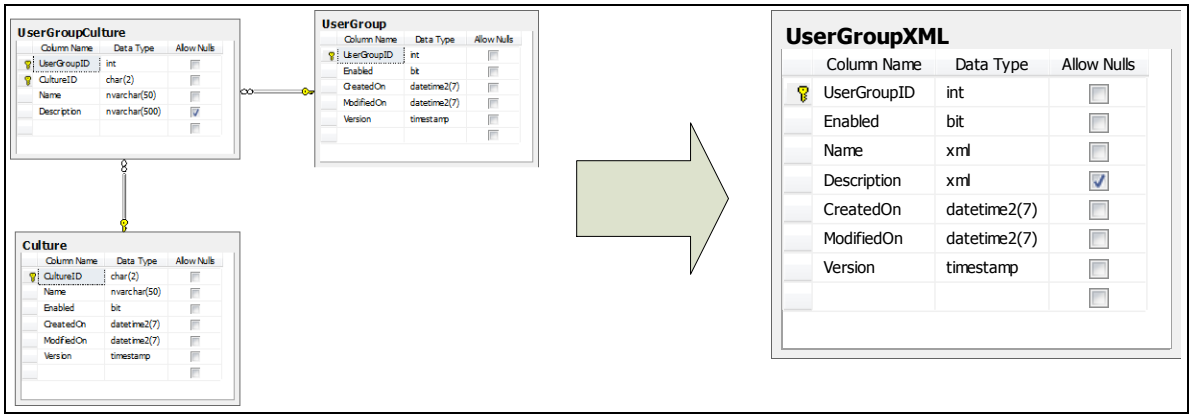
Slika 32: Deklaracija objavi/naroči storitve WCF

4 Sklepne ugotovitve

Kljub temu, da predstavljeni koncept uspešno zadostuje potrebam nadgradljivosti, prilagodljivosti in večjezični podpori sistema za procesno upravljanje avtomatiziranega skladišča, sem tekom razvoja odkril, da je prostora za izboljšave in nadaljni razvoj še veliko.

4.1 Izboljšava podatkovne baze

Vpeljava podpore večjezičnosti v strukturo podatkovne baze predstavlja nezanemarljiv vpliv na hitrost izvajanja poizvedovanj SQL. Ta problem je moč obiti z uporabo podatkovnega tipa XML za atribute, ki lahko nastopajo v večih jezikih. Omenjena rešitev bi zelo poenostavila strukturo baze in pohitrila povezana poizvedovanja, saj ne bi bilo več potrebe po uporabi transakcij in stikov. Pomožne tabele bi zamenjali ustrezni atributi XML (Slika 33).



Slika 33: Uporaba XML podatkovnega tipa pri strukturi podatkovne baze

Na ta način bi razbremenil strežnik podatkovne baze in breme prenesel na nivo odjemalcev v smislu razčlenjevanja (ang. parsing) podatkov XML. Slika 34 prikazuje primer zapisa XML.

```
<name>
  <text xml:lang="si-SI" value="Ime elementa" />
  <text xml:lang="en-EN" value="Element name" />
</name>
```

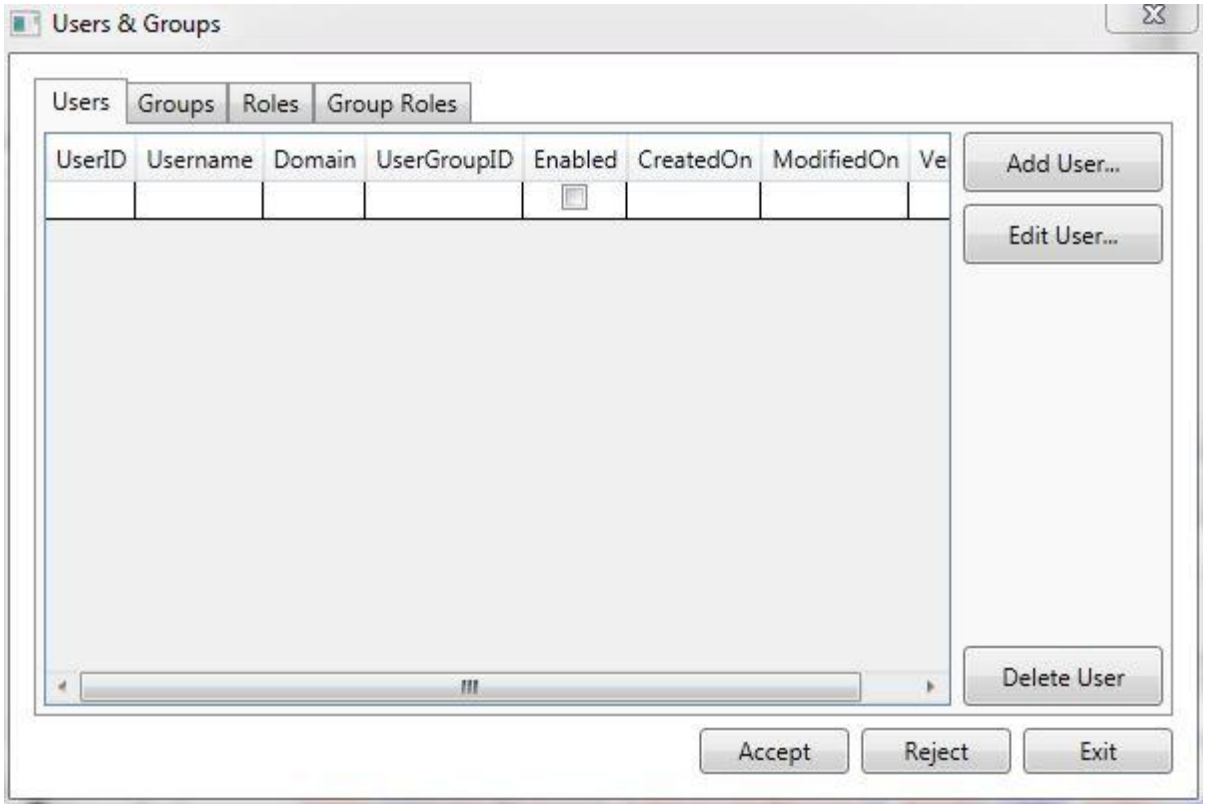
Slika 34: Primer vrednosti atributa *Name*

4.2 Uporaba ogrodja WPF pri implementaciji predstavitvenega nivoja

WPF je API za razvoj uporabniških vmesnikov v okolju Windows. Temelji na knjižnici DirectX s čimer je omogočeno strojno pospeševanje izvajanja ter vpeljava naprednih funkcij v prikazovanje UI, kot so prosojnost, prelivi in transformacije. WPF zagotavlja dosleden programski model za gradnjo aplikacij in jasno ločitev med uporabniškim vmesnikom in poslovno logiko.

Poleg tega WPF vpeljuje nov označevalni jezik XAML, ki omogoča alternativno definiranje elementov UI kakor tudi relacij do drugih elementov UI [8,9].

Izgled preprostega dialoga WPF prikazuje Slika 35, Slika 36 pa prikazuje XAML definicijo istega dialoga.



Slika 35: Izgled dialoga WPF

```
<Window x:Class="WpfClient.WindowUsersAndGroups"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="{StaticResource  txtUsersAndGroups}"  Height="400"  Width="600"  ShowInTaskbar="False"
ResizeMode="NoResize"               WindowStartupLocation="CenterOwner"               xmlns:my="clr-
namespace:WCS.BusinessEntities;assembly=BusinessEntities" Loaded="OnLoad">
   <Window.Resources>
      <my:WCSDataSet x:Key="wcsDataSet" />
   </Window.Resources>
   <Grid>
      <Button     Content="{StaticResource     txtExit}"     Margin="0,0,12,12"     Name="buttonExit"
HorizontalAlignment="Right" Width="75" Height="23" VerticalAlignment="Bottom" Click="OnExit" />
      <Button     Height="23"     HorizontalAlignment="Right"     Margin="0,0,93,12"     Name="button1"
```

```
VerticalAlignment="Bottom" Width="75" Content="{StaticResource txtReject}" />
      <Button     Height="23"     HorizontalAlignment="Right"     Margin="0,0,174,12"     Name="button2"
VerticalAlignment="Bottom" Width="75" Content="{StaticResource txtAccept}" />
      <TabControl Name="tabControl1" Margin="12,12,12,41">
         <TabItem Header="{StaticResource txtUsers}" Name="tabItemUsers">
            <Grid>
               <DataGrid     Name="dataGridUsers"     Margin="0,0,106,0"     HeadersVisibility="Column"
ItemsSource="{Binding Source={StaticResource wcsDataSet}, Path=User}" AutoGenerateColumns="False">
                  <DataGrid.Columns>
                     <DataGridTextColumn Binding="{Binding Path=UserID, Mode=OneWay}" Header="UserID"
IsReadOnly="True" />
                     <DataGridTextColumn Binding="{Binding Path=Username}" Header="Username" />
                     <DataGridTextColumn Binding="{Binding Path=Domain}" Header="Domain" />
                     <DataGridTextColumn Binding="{Binding Path=UserGroupID}" Header="UserGroupID" />
                     <DataGridCheckBoxColumn Binding="{Binding Path=Enabled}" Header="Enabled" />
                     <DataGridTextColumn      Binding="{Binding      Path=CreatedOn,      Mode=OneWay}"
Header="CreatedOn" IsReadOnly="True" />
                     <DataGridTextColumn      Binding="{Binding      Path=ModifiedOn,      Mode=OneWay}"
Header="ModifiedOn" IsReadOnly="True" />
                     <DataGridTextColumn      Binding="{Binding      Path=Version,      Mode=OneWay}"
CanUserSort="False" Header="Version" IsReadOnly="True" />
                  </DataGrid.Columns>
               </DataGrid>
               <Button Content="Add User..." Height="30" Name="buttonAddUser" VerticalAlignment="Top"
HorizontalAlignment="Right" Width="100" />
               <Button      Content="Edit      User..."      Height="30"      HorizontalAlignment="Right"
Margin="0,36,0,0" Name="button4" VerticalAlignment="Top" Width="100" />
               <Button  Content="Delete  User"  HorizontalAlignment="Right"  Name="button5"  Width="100"
Height="30" VerticalAlignment="Bottom" />
            </Grid>
         </TabItem>
         <TabItem Header="{StaticResource txtGroups}">

         </TabItem>
         <TabItem Header="{StaticResource txtRoles}">

         </TabItem>
         <TabItem Header="{StaticResource txtGroupRoles}">

         </TabItem>
      </TabControl>
   </Grid>
</Window>
```

Slika 36: XAML definicija dialoga

Ogrodje WPF in njegove multimedijske zmogljivosti bi lahko kasneje uporabil tudi za implementacijo nekaterih SCADA funkcionalnosti v uporabniški vmesnik odjemalca.

Seznam slik

Literatura

[1] Wikipedia - Enterprise Resource Planning. Dostopno na: http://en.wikipedia.org/wiki/Enterprise_resource_planning

[2] Wikipedia - Warehouse Management System. Dostopno na: http://en.wikipedia.org/wiki/Warehouse_management_system

[3] Wikipedia - Programmable Logic Controller. Dostopno na: http://en.wikipedia.org/wiki/Programmable_logic_controller

[4] Wikipedia - Windows Communication Foundation. Dostopno na: http://en.wikipedia.org/wiki/Windows_Communication_Foundation

[5] Juval Löwy, *Programming WCF Services*.: O'Reilly, 2007.

[6] Wikipedia - Dijkstra's algorithm. Dostopno na: http://en.wikipedia.org/wiki/Dijkstra's_algorithm

[7] WCF: Working with One-Way Calls, Callbacks, And Events. Dostopno na: http://msdn.microsoft.com/en-us/magazine/cc163537.aspx

[8] Wikipedia - Windows Presentation Foundation. Dostopno na: http://en.wikipedia.org/wiki/Windows_Presentation_Foundation

[9] Chris Sells and Ian Griffiths, *Programming WPF, Second Edition*.: O'Reilly, 2007.