



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ
САДУ



Ненад Зеленовић

ПРИМЕНА ИНДУСТРИЈСКОГ СТАНДАРДА ISA-95

МАСТЕР РАД
- Мастер академске студије –

Нови Сад, 2021.



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА


Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска публикација		
Тип записа, ТЗ:	Текстуални штампани документ/ ЦД		
Врста рада, ВР:	Мастер рад		
Аутор, АУ:	Ненад Зеленовић		
Ментор, МН:	др Бранислав Атлагић		
Наслов рада, НР:	Примена индустријског стандарда ISA-95		
Језик публикације, ЈП:	Српски (латиница)		
Језик извода, ЈИ:	Српски/енглески		
Земља публикавања, ЗП:	Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2021.		
Издавач, ИЗ:	Ауторски репринт		
Место и адреса, МА:	Факултет техничких наука (ФТН), Д. Обрадовића 6, 21000 Нови Сад		
Физички опис рада, ФО: <small>(поглавља/страна/ цитата/табела/слика/графика/прилога)</small>	7/37/18/6/32/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Примењене рачунарске науке и информатика		
Предметна одредница/Кључне речи, ПО:	ISA-95 стандард, компоненте, комуникација		
УДК			
Чува се, ЧУ:	Библиотека ФТН, Д. Обрадовића 6, 21000 Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У овом раду представљена је апликација која се бави симулацијом командовања опреме у пољу помоћу симулатора или преко клијентске апликације, примењивајући принципе и смернице ISA-95 стандарда		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Члан:	Др Драган Кукољ, ред.проф	
	Члан:	Др Милана Бојанић, доцент	Потпис ментора
	Члан, ментор:	Др. Бранислав Атлагић, доцент	



UNIVERSITY OF NOVI SAD • FACULTY OF TECHNICAL SCIENCES
21000 NOVI SAD, Dositej Obradović Square 6

KEY WORDS DOCUMENTATION

Accession number, ANO :			
Identification number, INO :			
Document type, DT :	Monographic publication		
Type of record, TR :	Textual material, printed/CD		
Contents code, CC :	Macrep thesis		
Author, AU :	Nenad Zelenović		
Mentor, MN :	Branislav Atlagić, Ph. D.		
Title, TI :	Application of industry standard ISA-95		
Language of text, LT :	Serbian (latin script)		
Language of abstract, LA :	Serbian/English		
Country of publication, CP :	Serbia		
Locality of publication, LP :	Vojvodina		
Publication year, PY :	2021.		
Publisher, PB :	Author's reprint		
Publication place, PP :	Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad		
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	7/37/18/6/32/0/0		
Scientific field, SF :	Electrical and computer engineering		
Scientific discipline, SD :	Applied computer science and informatics		
Subject/Keywords, S/KW :	ISA-95 standard, components, communication		
UC			
Holding data, HD :	Library of Faculty of Technical Sciences, D. Obradovića 6, 21000 Novi Sad		
Note, N :			
Abstract, AB :	This paper presents an application that deals with the simulation of the sets of command for the equipment in the field using a simulator or through a client application and applying principles and guidelines of the ISA-95 standard.		
Accepted by the Scientific Board on, ASB :			
Defended on, DE :			
Defended Board, DB :	Member:	Ph. D. Dragan Kukolj	
	Member:	Ph. D. Milana Bojanić	Menthor's sign
	Member, Mentor:	Ph. D. Branislav Atlagić	

	УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА 21000 НОВИ САД, Трг Доситеја Обрадовића 6	Датум:
	ЗАДАТАК ЗА ИЗРАДУ МАСТЕР РАДА	Лист/Листова:

(Податке уноси предметни наставник - ментор)

Студијски програм:	Примењено софтверско инжењерство
Руководилац студијског програма:	Др Александар Селаков, доцент

Студент:	Ненад Зеленовић	Број индекса:	E533/2019
Област:	Електротехника и рачунарство		
Ментор:	др Бранислав Атлагић		

НА ОСНОВУ ПОДНЕТЕ ПРИЈАВЕ, ПРИЛОЖЕНЕ ДОКУМЕНТАЦИЈЕ И ОДРЕДБИ СТАТУТА ФАКУЛТЕТА ИЗДАЈЕ СЕ ЗАДАТАК ЗА МАСТЕР РАД, СА СЛЕДЕЋИМ ЕЛЕМЕНТИМА:

- проблем – тема рада;
- начин решавања проблема и начин практичне провере резултата рада, ако је таква провера неопходна;
- литература

НАСЛОВ МАСТЕР РАДА:

ПРИМЕНА ИНДУСТРИЈСКОГ СТАНДАРДА ISA-95

ТЕКСТ ЗАДАТКА:

<p>Анализирати примену и начин функционисања индустријског стандарда ISA-95. Имплементирати софтверско решење за симулацију командовања опреме у пољу помоћу симулатора или преко клијентске апликације, примењивајући принципе и смернице ISA-95 стандарда.</p>
--

Руководилац студијског програма:	Ментор рада:

Примерак за: <input type="checkbox"/> - Студента; <input type="checkbox"/> - Ментора
--

LISTA KORIŠĆENIH SKRAĆENICA

Skraćenica

Pun naziv

ISA	<i>Instrumentation, Systems and Automation Society</i>
ERP	<i>Enterprise Resource Planning</i>
SCADA	<i>Supervisory Control And Data Acquisition</i>
PLC	<i>Programmable Logic Controller</i>
HMI	<i>Human-Machine Interface</i>
MES	<i>Manufacturing Execution System</i>
ORM	<i>Object-relational mapping</i>
WPF	<i>Windows Presentation Foundation</i>
WCF	<i>Windows Communication Foundation</i>
SQL	<i>Structured Query Language</i>
HP	<i>Hewlett-Packard</i>
CLI	<i>Common Language Infrastructure</i>
CLR	<i>Common Language Runtime</i>
CIL	<i>Common Intermediate Language</i>
API	<i>Application Programming Interface</i>
XML	<i>Extensible Markup Language</i>
HTML	<i>Hypertext Markup Language</i>
CSS	<i>Cascading Style Sheets</i>
XAML	<i>Extensible Application Markup Language</i>
UI	<i>User Interface</i>
MVVM	<i>Model-View-ViewModel</i>
UML	<i>Unified Modeling Language</i>
TCP	<i>Transmission Control Protocol</i>
UDP	<i>User Datagram Protocol</i>
RTU	<i>Remote Terminal Unit</i>

SADRŽAJ

1.	UVOD	8
2.	TEORIJSKE OSNOVE	10
2.1	Istorija ISA-95 standarda	10
2.2	Svrha ISA-95 standarda	10
2.3	Nivoi ISA-95 standarda	11
2.4	Razlike između sistema proizvodnih operacija i poslovne logistike	12
2.4.1	Fokus svrhe	13
2.4.2	Sredstva za prikupljanje podataka	13
2.4.3	Šta pokreće akcije.....	13
2.5	Delovi ISA-95 standarda	14
2.5.1	Deo 1 - Modeli i terminologija	15
2.5.2	Deo 2 - Atributi objektnog modela	18
2.5.3	Deo 3 - Modeli aktivnosti upravljanja proizvodnim operacijama	19
2.5.4	Deo 4 - Objektni modeli i atributi sistema proizvodnih operacija	20
2.5.5	Deo 5 - Proizvodne i poslovne transakcije	20
2.6	Prednosti korišćenja ISA-95 standarda	22
3.	TEHNOLOGIJA I ALATI	23
3.1	.NET Framework	23
3.2	Microsoft Visual Studio	23
3.3	C#	23
3.4	Windows Presentation Form (WPF)	23
3.5	Model-View-ViewModel (MVVM)	24
3.6	Windows Communication Foundation (WCF)	24
3.7	Microsoft SQL Server	24
3.8	Entity Framework.....	25
3.9	Easy Modbus Server Simulator	25
3.10	Dizajn paterni.....	26
3.10.1	Command.....	26
3.10.2	Singleton	27
4.	ARHITEKTURA SISTEMA.....	29
5.	IMPLEMENTACIJA APLIKACIJE.....	31

5.1	SCADA.....	31
5.1.1	Inicijalno pokretanje.....	31
5.1.2	Akvizicija podataka sa simulatora	32
5.1.3	Komandovanje	32
5.2	Simulator	33
5.3	MES	34
5.3.1	Inicijalno pokretanje.....	34
5.3.2	Obrade akvizicije	35
5.4	Klijentska aplikacija	36
5.4.1	Inicijalno pokretanje.....	37
5.4.2	Komandovanje	38
5.4.3	Alarmi	38
5.4.4	Izveštaji	40
6.	ZAKLJUČAK.....	43
7.	LITERATURA	44
	Podaci o kandidatu	45

1. UVOD

Za vođenje uspešne kompanije potrebno je obezbediti odgovarajući sistem kojim je moguće deliti informacije kroz slojeve i delove kompanije, a da sistem bude pouzdan i sinhronizovan. Takođe te informacije moraju biti razumljive i čitljive kroz sve delove kompanije, od pogona i kontrolnih soba do kancelarija gde se razvijaju biznis planovi. Vrlo često, zaposleni iz proizvodnih i poslovnih slojeva u međusobnoj komunikaciji koristili različite nazive za iste pojmove. To je dovodilo do nesporazuma, nepotrebnog odlaganja i skupih grešaka u poslovanju kompanija.

U poslednjih par decenija, industrijske kompanije su investirale u sisteme poslovne logistike (ERP - *Enterprise Resource Planning*). Takođe, veliki deo novca i vremena je potrošen na investiranje u automatizaciju industrijskih pogona pomoću upravljačkih sistema kao što je SCADA (*Supervisory Control And Data Acquisition*). Zbog toga se ukazala potreba približavanja i uvezivanja ova dva sistema u jednu celinu. ERP sistemi su najkorisniji kada se hrane trenutnim i validnim informacijama, onim koje se po pravilu nalaze u kontrolnim sistemima [1]. Mnoge kompanije su još uvek u fazi gde zaposleni ručno razmenjuju i obrađuju potrebne informacije između ERP i kontrolnih sistema. Vremenom se polako počela shvatati važnost automatizacije proizvodnih sistema praćena automatizacijom tokova informacija, što dovodi do smanjenja grešaka i značajne uštede vremena. Uvideo se značaj dramatičnog skraćenja vremena između naručivanja proizvoda od strane klijenata i isporuke proizvoda, kao i redukcije troškova proizvodnje, kao rezultat kvalitetnijeg planiranja poslovnih aktivnosti. Uslov za to je dostupnost pravih informacije u pravo vreme i na pravom mestu.

U projekte koji su usmereni na integraciji između ERP i upravljačkih sistema, obično je uključen veliki broj odseka, kao što su proizvodnja, održavanje, laboratorija i kancelarija. Navedeno ukazuje da su preduzeća suočena sa velikom prazninom između poslovnih i kontrolnih sistema. Kada se pokuša ta praznina smanjiti, pojavljuju se problemi u komunikaciji između ljudi i sistema.

Stoga je bilo potrebno da se razvije standard koji će uvesti odgovarajuće principe, terminologiju i modele za uspešno poslovanje kompanije, kao i mogućnosti integrisanja slojeva u jednu sinhronizovanu celinu. Kao rešenje tog problema uvodi se ISA-95 standard.



Slika 1.1 – ISA-95 Standard

Predmet rada je istraživanje koncepta koji je izložen u standardu ISA-95, kao i implementacija jedne sprege ovog tipa na primeru školskog akviziciono upravljačkog sistema. Iako daleko od kompleksnosti realnih industrijskih sistema, ovaj rad treba primerom da ukaže na ključne komponente koje su neophodne pri sprežanju poslovnih i proizvodnih segmenata neke kompanije. Naravno, kako je poslovni segment podržan ERP softverom, a proizvodni svojim SCADA sistemom, u radu su u rudimentarnoj formi realizovane obe komponente, kao i sprega između njih u duhu standarda ISA-95.

Samo izlaganje je organizovano u poglavlju 4 – Arhitektura sistema. Posle uvoda, u okviru teorijskih osnova je izložen sadržaj, primena, ciljevi, kao i prednosti korišćenja standarda ISA-95. Potom je u poglavlju 3 – Tehnologija i alati objašnjeno koje su sve tehnologije i alati korišćeni za implementaciju aplikacije. U poglavlju 5 – Implementacija aplikacije, je detaljno objašnjeno na koji način je aplikacija realizovana. Objašnjeno je koji su delovi tj. komponente ISA-95 standarda integrisani u konačno rešenje aplikacije i način na koji oni međusobno komuniciraju. Poglavlje 6 je ostavljeno za zaključak rada u kome pokušano ukratko objasniti predmet istraživanja rada, prednosti i mane ovakvog sistema. Takođe pokušaćemo da izvedemo i potencijalna mesta za unapređenje sistema. Zatim sledi poglavlje 7, koje je rezervisano za literaturu koja je korišćena u ovom radu.

2. TEORIJSKE OSNOVE

2.1 Istorija ISA-95 standarda

ISA-95 predstavlja internacionalni standard koji služi za integrisanje biznis i kontrolnih sistema u cilju smanjenja rizika, troškova i stvaranja grešaka koje idu paralelno sa implementacijom interfejsa između takvih sistema. ISA je globalna neprofitna organizacija. Prvobitno ISA je označavala *Instrument Society of America*, ali je ovaj naziv kasnije preimenovan u *Instrumentation, Systems and Automation Society*. Ovaj naziv je promenjen 2000 godine. ISA definiše svoje ključne aktivnosti kao što su: standardizacija, sertifikacija, obrazovanje i obuka, publikacije i izložbe iz oblasti industrijske automatizacije [2].

Iako je standard kreiran 1995 godine, njegovi principi i namena su do današnjeg dana ostali validni i relevanti. Većina razvojnog tima ISA-95 standarda je takođe razvijao stariju verziju standarda pod imenom ISA-88 koji se koristio za kontrolu skupa instrukcija (*batch control*), procesa i signala. Tako da nije čudno što se modeli i terminologija ova dva standarda približno podudaraju. Oba standarda pružaju koncepte koji omogućavaju definisanje kreiranja određenih proizvoda.

2.2 Svrha ISA-95 standarda

ISA-95 nije sistem automatizacije, već metoda, način rada, razmišljanja i komuniciranja. Ova metoda je opisana u nekoliko dokumenata, gde se svaki dokument sastoji od stotinu stranica. Ova dokumenta sadrže modele i terminologiju koja može da se iskoristi za analizu pojedinačne proizvodnje određene kompanije. Svaki od modela fokusira se na specifične aspekte integracije.

Komuniciranje o sistemu može biti teško, jer različiti ljudi u isti razgovor često koriste različita imena opštih termina. ISA-95 definiše reči koje se odnose na sisteme poslovne logistike (ERP) i na kontrolne sisteme. ISA-95 stavlja ovu terminologiju u modele koji jasno pokazuju vezu između različitih pojmova. Ovaj princip možemo uporediti sa nacrtima za kuću. Reči *prozor*, *vrata*, *zid* i *krov* su svima nama poznati i koristimo ih za međusobno razgovaranje kada se spominje kuća. Svaka kuća je drugačija, ali i dalje možemo opisati svaku kuću sa istim simbolima i rečima za vrata, krovove, zidove i prozore. Isto se odnosi i na ISA-95. Ne postoje dve slične kompanije i ipak možemo koristiti ISA-95 modele i terminologiju za razgovor sa drugim kompanijama o aktivnostima, funkcijama, tokovima informacija unutar tih kompanija... Kao rezultat, postalo je lakše ne samo na nivou ljudske komunikacije, već i na tehničkom nivou, za integraciju različitih sistema.

Cilj ISA-95 standarda je da smanji troškove, rizike i greške povezane sa implementacijom interfejsa između ERP i kontrolnih sistema. Standard se može koristiti za pojednostavljenje implementacije novih softverskih proizvoda i da na kraju se stvori laka interoperabilnost između ERP i kontrolnih sistema. ISA-95 definiše veliki broj potencijalnih prednosti. Omogućava kreiranje raznih alata za lakšu integraciju ERP i kontrolnih sistema. Pruža se krajnjim korisnicima da lakše kreiraju svoje zahteve. Dodatne pogodnosti tiču se integracije uopšteno, kao što su smanjenje troškova proizvodnih procesa i optimizacija lanca snabdevanja (*Supply Chain*).

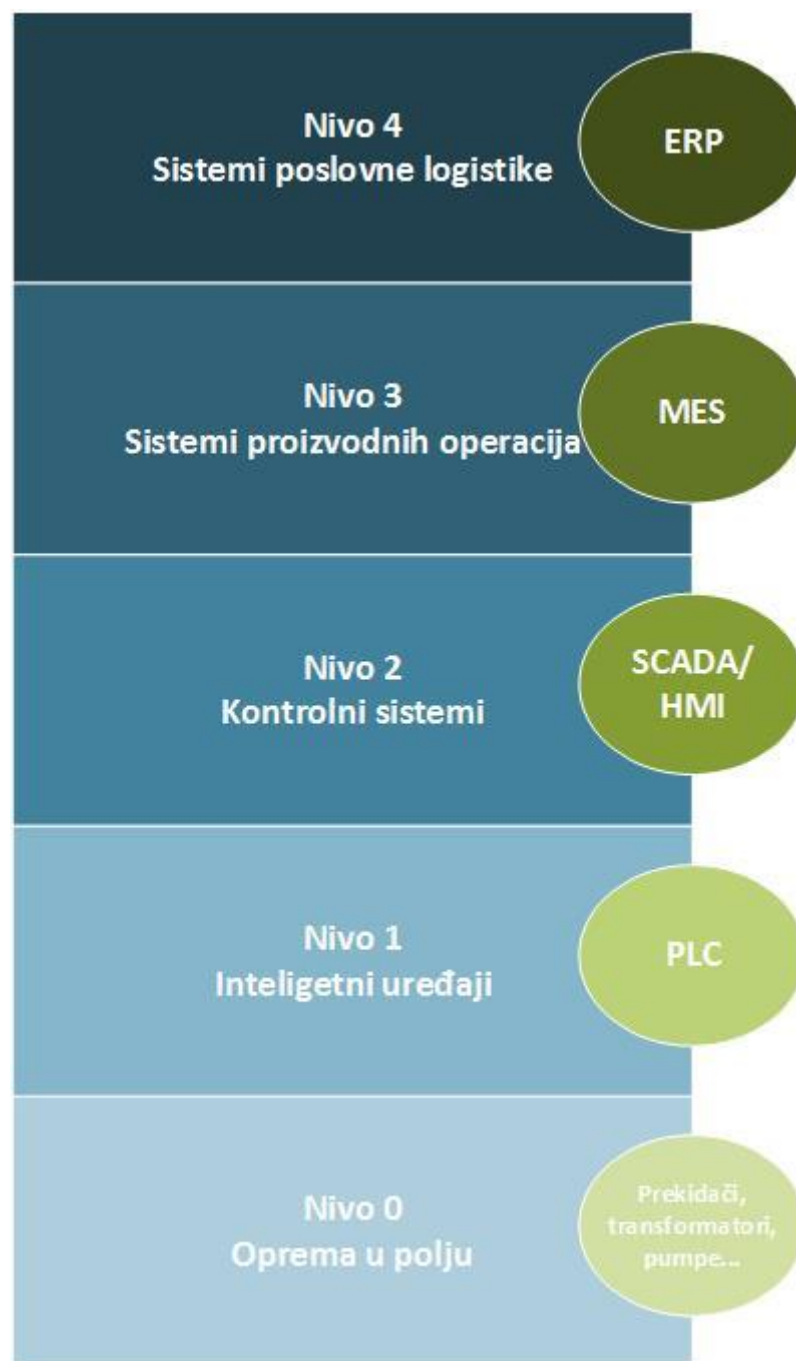
2.3 Nivoi ISA-95 standarda

ISA-95 standard deli postojenja, opremu i imovinu na 5 nivoa [3]:

- **Nivo 0 (nulti)** – predstavlja samu opremu u polju (prekidači, transformatori, pumpe...)
- **Nivo 1** – predstavlja opremu koja očitava stanja sa polja tj iz nultog nivoa, kao i manipulaciju nad njima. Iako ovaj nivo vrši manipulaciju nad opremom u polju, on ne može samoinicijativno da pokrene te manipulacije. Za to dobija komande od narednog nivoa. PLC (*Programmable Logic Controller*) predstavlja kontroler koji zadovoljava potrebe ovog nivoa.
- **Nivo 2** – omogućava monitoring, nadgledanje, manuelnu i automatsku kontrolu nad opremom u polju. Ovaj nivo sadrži kompletan uvid u infrastrukturu celog sistema, kao i uvid u svaki vid promene vrednosti opreme u polju. Iz ovog nivoa se zadaju setovi instrukcija za promene vrednosti opreme u polju. Te instrukcije se šalju nivou 1, koji ih izvršava. Sistemi tipični za ovaj nivo predstavljaju SCADA (*Supervisory Control And Data Acquisition*) i HMI (*Human-Machine Interface*).
- **Nivo 3** – ovaj nivo definiše aktivnosti radnog procesa za proizvodnju željenih krajnjih proizvoda, sadrži uvid u detaljan raspored proizvodnje, kao i kompletne istorijske podatke. U ovom sloju se konstantno vrše moguće optimizacije u procesu proizvodnje. Neki od uslova u procesu proizvodnje koji se moraju ispoštovati su da proizvod bude kvalitetan, da se proizvodi bezbedno, pouzdano i efikasno. Najznačajni sistem za ovaj nivo predstavlja MES (*Manufacturing Execution System*)
- **Nivo 4** – u ovom nivou se obavljaju funkcije kao što su operacioni menadžment, planiranje proizvodnje i logistika. Određuje se koji će se materijali koristiti, pregled stanja na zalihama, zaposlenje radnika kao i menadžment njihovih radnih smena. Utvrđuju se dugoročni, srednjoročni ili kratkoročni planovi koji će doneti profit. Sistem koji se koriste za ovakve svrhe je ERP (*Enterprise Resource Planning*)

Drugi nazivi za ove nivoe su:

- **Nivo 1** – Inteligentni uređaji
- **Nivo 2** – Kontrolni sistemi
- **Nivo 3** – Sistemi proizvodnih operacija
- **Nivo 4** – Sistemi poslovne logistike



Slika 2.1 - Nivoi ISA-95 standarda

Jedna od bitnih karakteristika nivoa jeste njihova komunikacija. Jedna nivo može da komunicira samo sa susednim nivoima tj sa nivom ispred i iza sebe uz korišćenje odgovarajućih interfejsa.

2.4 Razlike između sistema proizvodnih operacija i poslovne logistike

Sistem proizvodnih operacija i poslovne logistike dele neke uobičajne sličnosti u proizvodnom okruženju. Ova dva sistema su dovoljno slična da se njihove jedinstvene karakteristike lako mogu predvideti [4]. U ovom poglavlju objasnićemo osnovne razlike između ova dva složena i najveća nivoa ISA-95 standarda. Glavnu razliku između ova dva nivoa možemo videti na tabeli 2.1.

Sistem proizvodnih operacija (MES)	Sistem poslovne logistike (ERP)
<ul style="list-style-type: none"> • Gledanje u realnom vremenu • Fizičko kretanje proizvoda i odgovornost • Pogled iz radnih centara • Kako se proizvode proizvodi i gde se nalaze? 	<ul style="list-style-type: none"> • Dugoročno gledanje • Troškovi proizvodnje i ukupna zarada • Pogled iz sala za sastanke • Koliko vrede moji proizvodi?

Tabela 2.1 - Razlika između sistema proizvodnih operacija i poslovne logistike

2.4.1 Fokus svrhe

ERP funkcioniše kao sredstvo za razmenu informacija unutar organizacije. Sistem povezuje svaki deo posla i omogućava nesmetan protok informacija. Delujući kao sveobuhvatni put podataka čini ERP dragocenim alatom za upravljanjem, jer donosiocima odluka daje mogućnost da duboko zarone u svaki deo poslovanja i na smislene načine povežu ranije udaljene podake.

Razlog za dodavanje MES-a je, s druge strane, pružanje sredstava za preciznu kontrolu proizvodnog procesa. MES preduzima korake ili generiše izveštaje na osnovu onoga što se trenutno događa kako bi nagledale promenljive koje utiču na efikasnost proizvodnje. MES sinhronizuje brojne aspekte izrade kako bi postigao najbolje moguće rešenje za stvaranje profitabilnog procesa.

2.4.2 Sredstva za prikupljanje podataka

Ljudi obično većinu informacija prosleđuju u ERP. Ovo ručno prikupljanje podataka dobro funkcioniše, jer se sistem prvenstveno bavi prikupljanjem, organizovanjem i razmenjivanjem informacija širom organizacije radi planiranja i vođenja. Savremeni ERP sistemi imaju jednu bazu podataka za celu organizaciju, tako da se suvišni unosi smanjuju ili uklanjaju. Jedno skladište podataka takođe čini nesmetan proces razmene informacija u različitim funkcionalnim oblastima. Fokusira se na rad u vremenskim okvirima kao što su godine, meseci, nedelje i dani.

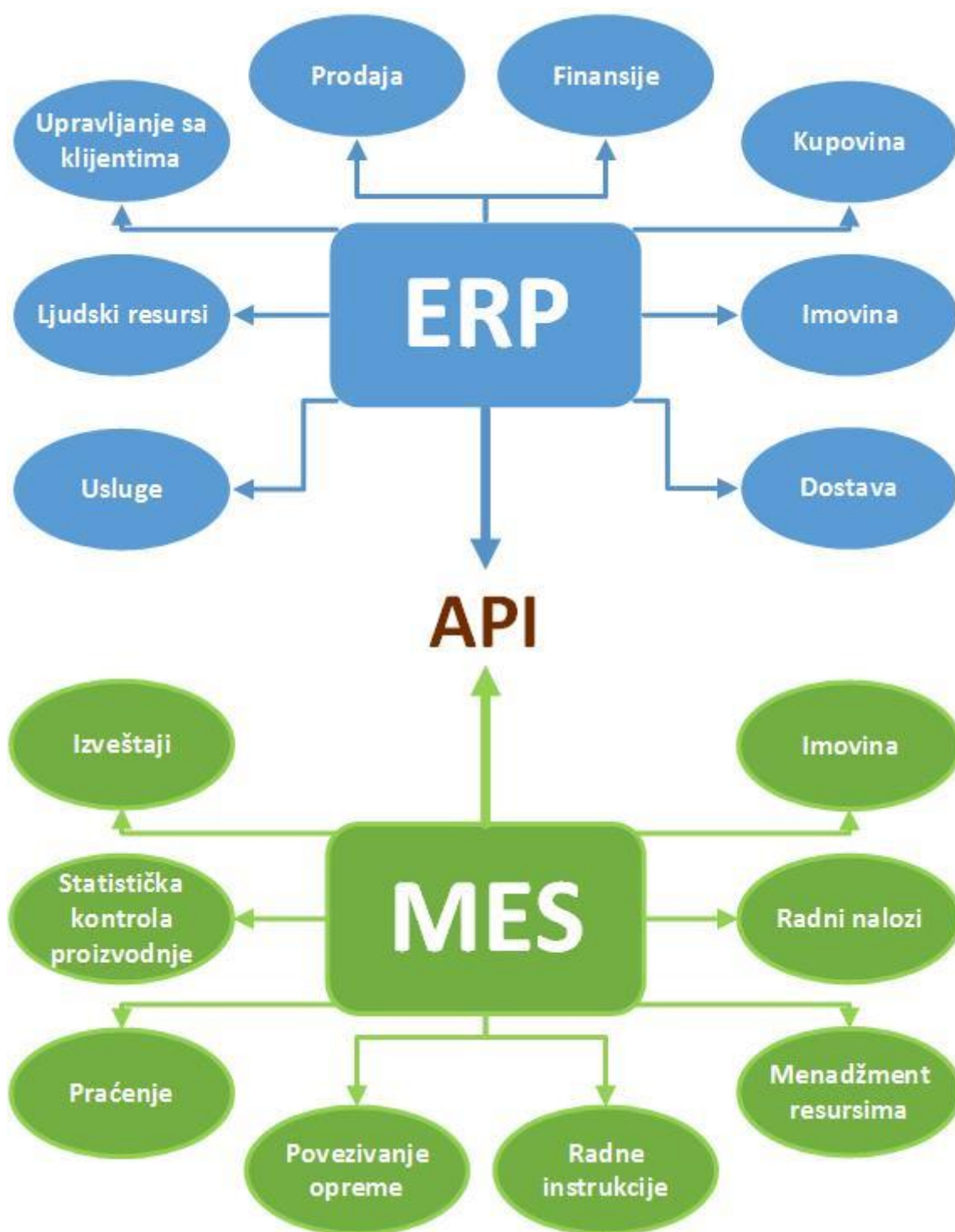
MES se koristi za pokretanje proizvodne operacije, upravljanjem i izveštavanjem o aktivnostima postrojenja u slučaju događaja, u realnom vremenu. Održava se evidencija istorije podataka i ističu se izuzetci. Tipične funkcije su elektronski zapisi skupa instrukcija, merenje i doziranje, upravljanje opremom, podešavanje i čišćenje. Ove informacije pružaju tačne i pravovremene informacije ERP-u, omogućavajući kompaniji da reaguje dovoljno brzo kako bi mogla pratiti korake koji se brzo menjaju. Smanjivanje latencije na ovaj način povećava šanse za ostvarivanje većeg novčanog prinosa. MES radi u vremenskim okvirima kao što su dani, smene, sati i minute.

2.4.3 Šta pokreće akcije

ERP izvršava instrukcije na osnovu finansijskih transakcija. Kada kupci izvrše narudžbine, dobavljači šalju te narudžbine. ERP je multifunkcionalan, ali ga programeri grade prvenstveno oko ekonomske podstrukture.

S druge strane, MES je vođenjem dešavanjem događaja. Ovaj sistem je dizajniran za nagledanje događaja specifičnih za proizvodno okruženje. Nagledanje podataka u realnom vremenu omogućava MES-u da obavlja zadatke kao što su obezbeđivanje usaglašenosti za proizvodnim procesom, praćenje

potrošnje zaliha, zakazivanje održavanja mašina na osnovu performansi i preuređivanje postupaka radi efikasnijeg korišćenja raspoloživih resursa.



Slika 2.2 - Grafički prikaz ERP i MES nivoa

2.5 Delovi ISA-95 standarda

ISA-95 standard se sastoji od nekoliko delova, gde svaki deo opisuje neke funkcije koje se obavljaju unutar određenih nivoa ISA-95 standarda ili opisuje način komuniciranja između tih nivoa. U daljem tekstu pokušaćemo da nabrojimo delove standarda i da ih objasnimo.

2.5.1 Deo 1 - Modeli i terminologija

Prvi deo predstavlja različite modele i terminologiju koja se koristi za pripremu i izvršavanje projekata, kao i za komuniciranje između ERP i MES sistema. Prema ISA-i, ovde je cilj povećati doslednost terminologije interfejsa i smanjiti rizik, troškove i greške povezane sa primenom ovih interfejsa.

Najvažniji modeli koji su definisani unutar prvog dela su:

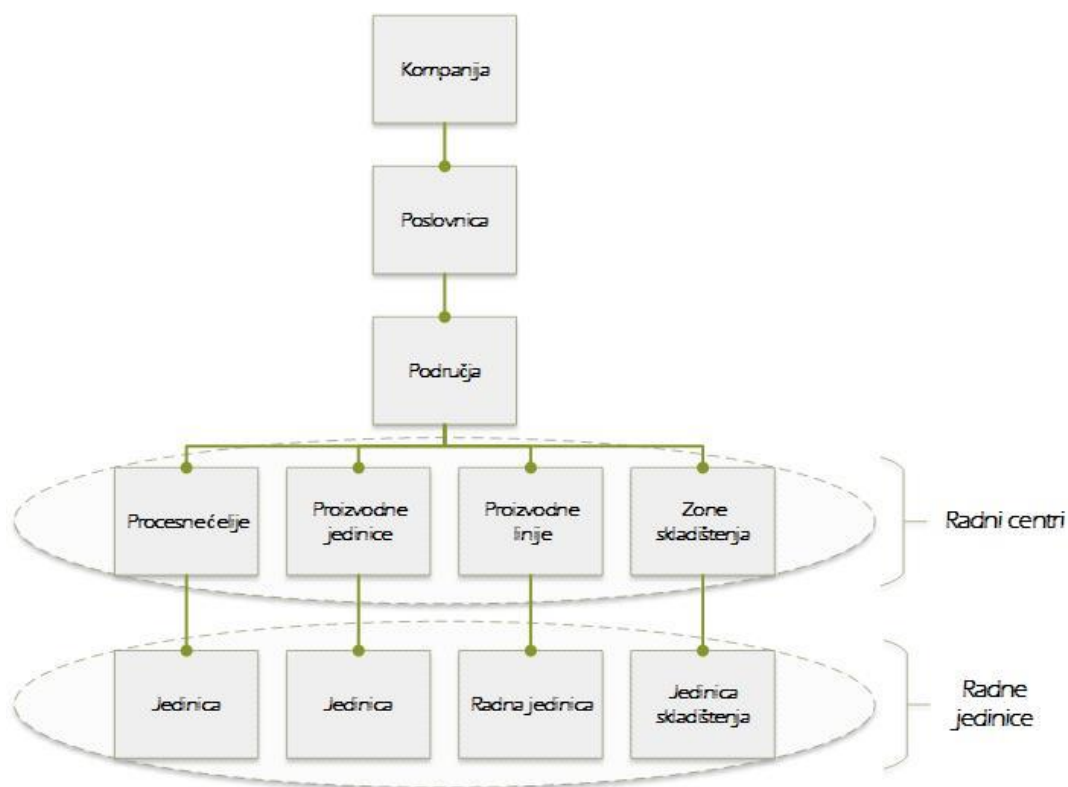
- Funkcionalno hijerarhijski model
- Model hijerarhije opreme
- Funkcionalni model upravljanja preduzećem
- Objektni model
- Model razmene informacija

2.5.1.1 Funkcionalno hijerarhijski model

Funkcionalno hijerarhijski model objašnjava kako se sistem deli na nekoliko nivoa unutar neke kompanije, i gde svaki nivo obavlja zadatu, predefinisanu funkciju. Najniži nivoi upravljaju sa samom opremom u polju, dok na najvišim nivoima se sklapaju biznis planovi za unapređenje poslovanja kompanije. Više o ovom modelu možemo pronaći u poglavlju 2.3.

2.5.1.2 Model hijerarhije opreme

Fizička imovina preduzeća koja se bavi proizvodnjom je obično organizovana na hijerarhijski način kao što je opisano na slici 2.3. Ovaj model definiše odgovornosti za različite definisane nivoe funkcija u modelu. Model hijerarhije opreme dodatno definiše neke od objekata koji se koriste u razmeni informacija između funkcija.



Slika 2.3 - Model hijerarhije opreme

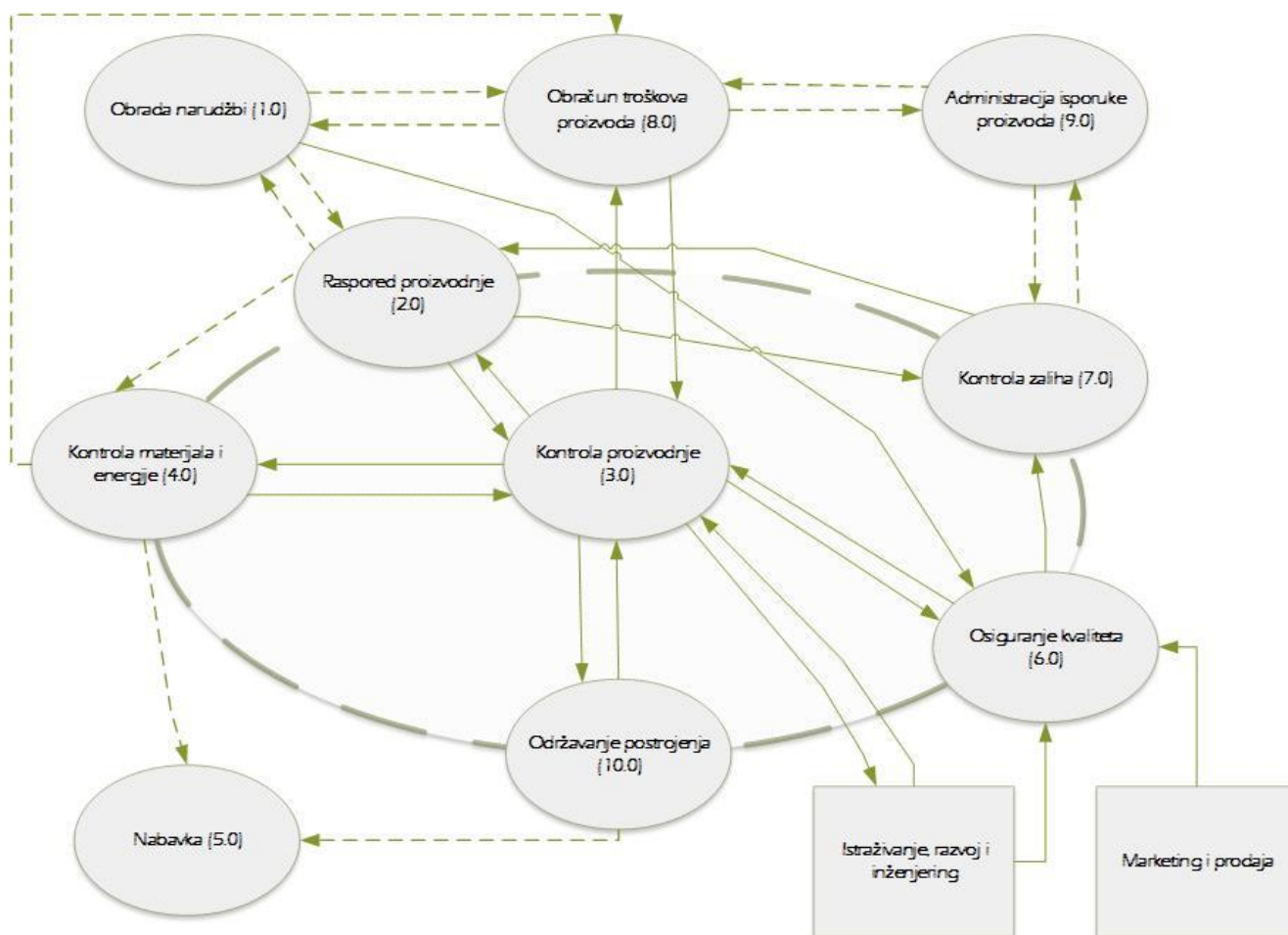
Model hijerarhije opreme jasno definiše kako se fizička imovina preduzeća može podeliti na različite nivoe. Na nivou preduzeća, imamo naziv kompanije, kao što je npr. *Coca Cola*. Preduzeća mogu sadržati jednu ili više poslovnica, kao što su npr. poslovnica u Beogradu, Pragu ili u Berlinu... Za imenovanje poslovnica se obično koristi ime grada. Zatim jedna poslovnica može sadržati jedno ili više područja, što se može protumačiti kao proizvodnja odeljenja. Primer bi bio da unutar Coca Cola poslovnice u Beogradu postoji područje gde vrši proizvodnja i gde se vrši pakovanje proizvoda. Kompanije često grupišu područja unutar specifične grupe proizvoda, da bi se smanjili troškovi proizvodnje i transporta. Svako područje može sadržati kombinaciju jednog ili više radnih centara. Ovo je opšti termin za procesne ćelije, proizvodne jedinice, proizvodne linije i zone sladištenja.

2.5.1.3 Funkcionalni model upravljanja preduzećem

Ovaj model objašnjava kako proizvodne kompanije obavljaju jako puno različitih funkcija. Standard opisuje najsitnije zadatke unutar svake funkcije. Svaka kompanija će imati drugo ime za ove funkcije. Neke kompanije su podelile zadatke za jednu funkciju preko dva ili više odeljenja, dok druga preduzeća dodele odgovornost jednom odeljenju da obavi dve ili više funkcija. Funkcija koja je možda veoma važna u jednoj kompaniji, u drugoj zaslužuje veoma malo pažnje. Na primer, kontrola materijala za proizvodnju automobila je vrlo kompleksna i osetljiva funkcija, dok za kompaniju koja se bavi distribucijom vode je prilično jednostavna.

Nije nam namenjeno da koristimo ovaj model za reorganizaciju kompanije. Model je vrsta mernog štapa, duž kojeg možete položiti karakteristike određene kompanije da biste stekli uvid u raspon odeljenja, kao i u podelu odgovornosti. Pomoću modela možemo odrediti i šta spada, a šta ne spada u okvir odgovornosti neke kompanije.

Ovaj model sadrži dvanaest funkcija, svaka sa serijskim brojem (Slika 2.4). Većina funkcija su ovalnog oblika. Funkcije *Istraživanje*, *razvoj i inženjering*, kao i *Marketing i prodaja* su prikazani u pravougaonog obliku što ukazuje da su to spoljni entiteti. To su komponente izvan granice modela koje podatke šalju i primaju od funkcija. Široka isprekidana linija u modelu prikazuje granicu između ERP nivoa i MES nivoa. Sve ono što se nalazi izvan isprekidanih linija pripada ERP nivou, dok unutar granica pripada MES nivou. *Obrada narudžbi*, *Nabavka*, *Istraživanje*, *razvoj i inženjering*, *Marketing i prodaja*, *Administracija isporuke proizvoda* i *Obračun troškova proizvoda* su dakle sastavni deo ERP nivoa. Funkcija koja pripada isključivo MES nivou je *Kontrola proizvodnje*, dok su funkcije *Raspored proizvodnje*, *Kontrola materijala i energije*, *Održavanje postrojenja*, *Osiguranje kvaliteta* i *Kontrola zaliha* pripadaju i ERP i MES nivou, pa ih oba nivoa izvršavaju.



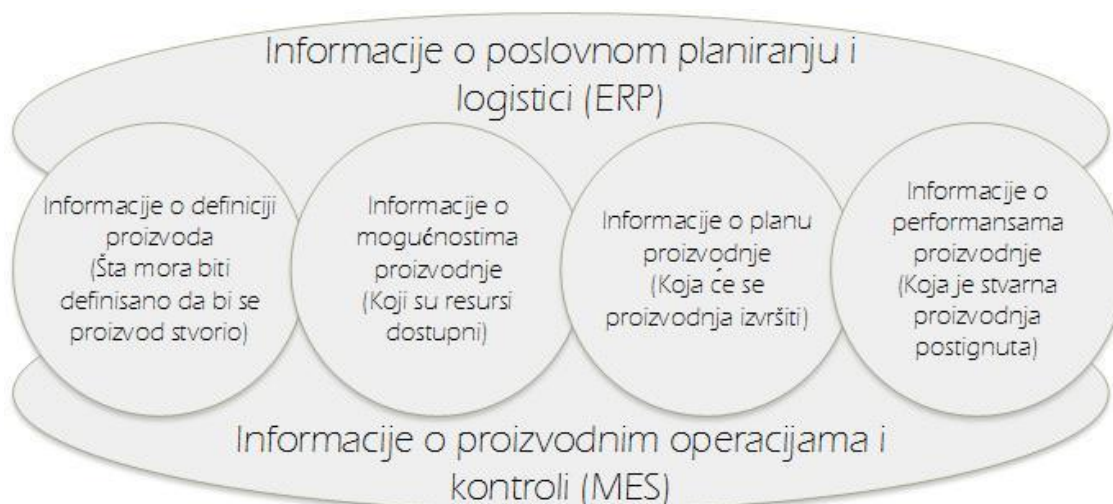
2.5.1.4 Objektni model

Ako uporedite informacije u različitim tokovima informacija, nailazite na mnogo preklapanja. Na primer, i recepti i rasporedi proizvodnje sadrže informacije o materijalima. Da bi se standardizovala velika količina informacija koje treba razmeniti, neophodno je prvo sprovesti proces normalizacije. Pod tim se podrazumeva, analiza tokova informacija radi utvrđivanja osnovnih podataka, tako da možemo da razvijemo standardne modele za razmenu informacija, u kojima se podaci pojavljuju samo jednom. ISA-95 standard definiše sledeće osnovne podatke: opremu, materijal i osoblje. Ovo su osnovni gradivni elementi sa kojima konstruišemo tokove informacija. Pored osnovnih gradivnih elemenata osoblja, opreme i materijala, ISA-95 standard uvodi pojam procesni segmenti.

Procesni segment je logička grupa opreme, osoblja i materijala potrebna za obavljanje određenog dela procesa. Primeri procesnog segmenta bi bilo *mešanje*, *testerisanje* i *farbanje*. Za *mešanje*, potrebno nam je radnik (Osoblje), mešalica (Oprema) i sirovi sastojci (Materijal). Za *testerisanje* potrebno nam je testera (Oprema), drvo (Materijal) i zaposleni (Osoblje). A za *farbanje*, potrebna vam je farba (Materijal), slikar (Osoblje) i četka (Oprema). Po definiciji segmenti ne moraju da uključuju opremu, materijal i osoblje. Za procesni segment kao što je *inspekcija*, na primer, potrebno nam je samo inspektor.

2.5.1.5 Model razmene informacija

Neke informacije moraju da se dele između MES i ERP sistema, kao što je prikazano na slici 2.5.



Slika 2.5 - Model razmene informacija

Da bi se standardizovali tokovi informacija, ISA-95 standard grupiše sve informacije koje treba razmeniti u četiri kategorije: informacije o mogućnostima proizvodnje, informacije o definiciji proizvoda, informacije o planu proizvodnje i informacije o performansama proizvodnje. Objektni modeli kao što su oprema, osoblje, materijal i procesni segmenit formiraju osnovne građevinske blokove za svaku kategoriju. Informacije o mogućnostima proizvodnje su informacije o raspoloživosti proizvodnih resursa, kao što su mašine, alati, operateri, privremeni radnici, sastojci i energija. Informacije o definiciji proizvoda predstavlja prikupljanje informacija koje opisuju kako se pravi proizvod, kao što su recepti i uputstva za montažu. Informacije o planu proizvodnje su informacije o tome šta treba da se proizvede i kada da se proizvede, kao što je plan proizvodnje ili prognoza. A informacije o performansama proizvodnje su informacije o tome šta i koliko je proizvedeno i koje osobe i resursi su korišćeni, kao što je izveštaj.

2.5.2 Deo 2 - Atributi objektnog modela

Ako želite da primenite ISA-95 standard, da biste kreirali standardizovane interfejsne između ERP i MES sistema, ne možete bez drugog dela. Drugi deo se proširuje na model informacija iz prvog dela. Daje detaljan opis informacija, u obliku atributa. Za svaki objekat u prvom delu, drugi deo ISA-95 standarda obezbeđuje tabelu sa standardnim atributima objekta.

Na primer, objekat oprema sastoji se od standardnog ID-a i opisa, a objekat klasa opreme takođe sadrži standardni ID i opis. Uzmimo, na primer, hidrauličnu pumpu P105 (tabele 2.2 i 2.3). P105 je u ovom slučaju ID opreme, dok je hidraulična pumpa sa oznakom P105, opis za pumpu P105. Pumpa P105 pripada klasi opreme hidrauličnih pumpi. Hidraulične pumpe su opis klase, a klasa ima jedinstven ID - 5.

Ime atributa	Vrednost atributa
ID	P105
Opis	Hidraulična pumpa sa oznakom P105

Tabela 2.2 - Atributi objekta oprema za pumpu P105

Ime atributa	Vrednost atributa
ID	Jedinstven ID - 5
Opis	Hidraulična pumpa

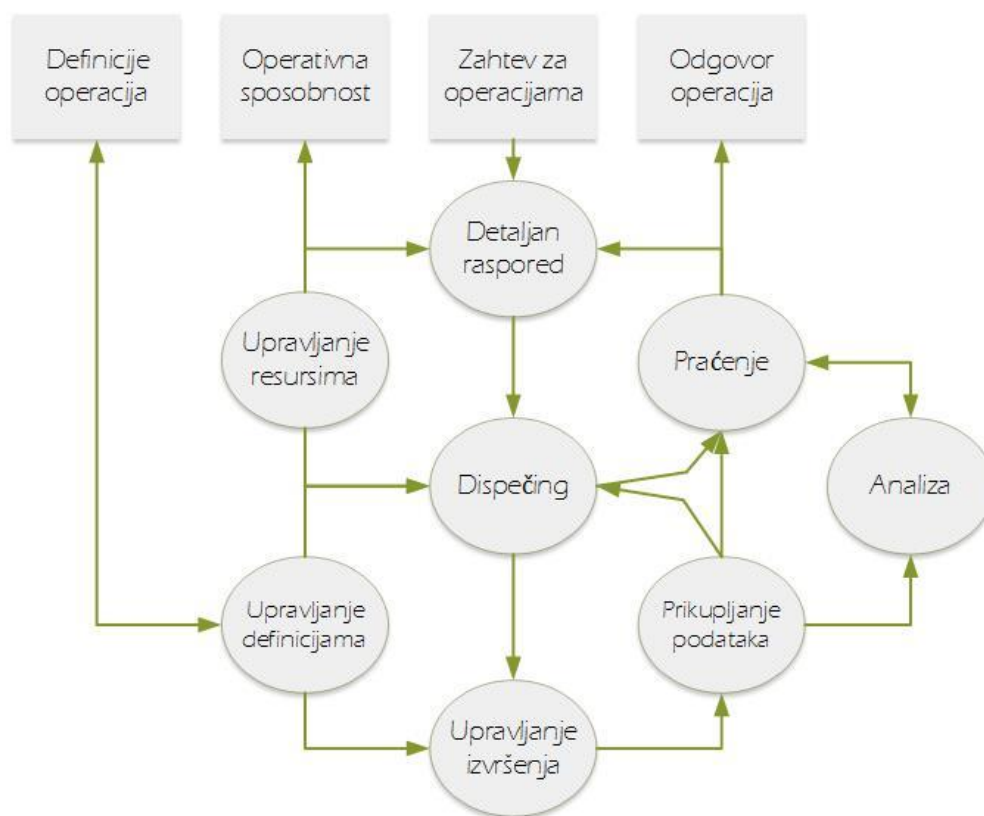
Tabela 2.3 - Atributi objekta klasa opreme za hidraulične pumpe

Na nivou opreme, osoblja i materijala, ISA-95 standard definiše samo nekoliko veoma osnovnih atributa, tačnije ID i opis. Naravno, resursi imaju mnogo više svojstava. Na primer, karakteristike pumpe uključuju kapacitet, maksimalan pritisak i tip. Atributi materijala mogu da uključuju boju, viskoznost i pH vrednost. I na kraju, primeri osoblja su broj telefona, nivo, sertifikacija i dostupnost za prekovremeni rad. Važno je da svaka industrija, zadrži slobodu da sama odluči koji atributi su relevantni pored standardnog ID-a i opisa. Mnogi atributi su specifični za određenu industriju. Na primer, pH vrednost materijala je za veliki broj industrija nebitna. Zbog toga se komitet odlučio na fleksibilan metod za dodavanje dodatnih atributa u obliku svojstva. Korisnici mogu slobodno da definišu onoliko svojstava za materijal, osoblje i opremu koliko im je potrebno.

2.5.3 Deo 3 - Modeli aktivnosti upravljanja proizvodnim operacijama

Treći deo standarda ISA-95 objavljen je na leto 2005. Ovaj deo se u potpunosti fokusira na MES sloj (Nivo 3) i predstavlja modele i terminologiju koje možete koristiti za analizu i opisivanje aktivnosti unutar MES sloja. Naglašavaju se dobre prakse za proizvodne operacije i mogu se koristiti za poboljšanje postojećih proizvodnih sistema. Mogu se primeniti u teško automatizovanim kompanijama i u kompanijama koje u velikoj meri rade ručno. Treći deo navodi nekoliko potencijalnih prednosti koji se odnose na modele i terminologiju. Na primer, može pomoći prodavcima softvera da razviju odgovarajuće proizvode. A za krajnje korisnike, olakšava opisivanje njihovih želja i zahteva na standardizovan i konzistentan način.

Na slici 2.4 mogli smo da vidimo funkcionalni model razmena informacija i na njemu nekoliko funkcija koje obavljaju ERP i MES nivoi. Unutar tog modela funkcije koje obavlja MES nivo mogu se podeliti na četiri grupe aktivnosti: *aktivnosti proizvodnje* (kontrola i raspored proizvodnje), *aktivnosti zaliha* (kontrola materijala, energije i zaliha), *aktivnosti održavanja* (održavanje postrojenja) i *aktivnosti kvaliteta* (osiguranje kvaliteta). Komitet ISA-95 standarda razvio je jedan generički model kao osnovu za opisivanje aktivnosti u oblastima proizvodnje, održavanja, testiranja kvaliteta i zaliha (Slika 2.6).



Slika 2.6 - Modeli aktivnosti upravljanja proizvodnim operacijama

Ovaj model ISA-95 standarda pruža jasnu i logičnu podjelu svih aktivnosti unutar proizvodnih odeljenja, skladišta, laboratorija i odeljenja za održavanje (automatizovano ili na drugi način). Na primer, neko će morati da vodi evidenciju o dostupnosti osoblja, mašina i materijala (**Upravljanje proizvodnim resursima**). Takođe će biti neophodno održavati recepte i uputstva koja operateri koriste (**Upravljanje definicijom proizvoda**). Detaljan raspored (**Detaljan raspored proizvodnje**) potreban je za optimalno kombinovanje proizvodnih naloga, uzimajući u obzir ograničene kapacitete proizvodnih linija, promene tokom vremena i čišćenje. Dalje, neko će morati da otpremi porudžbine i dodeli zadatke timovima (**Dispečing**). Funkcija **Upravljanja izvršenjem proizvodnje** osigurava da proizvodno osoblje zaista izvršava te zadatke, u skladu sa važećim standardima kvaliteta. Na kraju, moraćete da prikupite različite podatke tokom proizvodnog procesa (**Prikupljanje podataka o proizvodnji**) i preformulišemo ih u informacije (**Praćenje proizvodnje**) za upotrebu u analizi i optimizaciji aktivnosti proizvodnog odeljenja (**Analiza proizvodnje**).

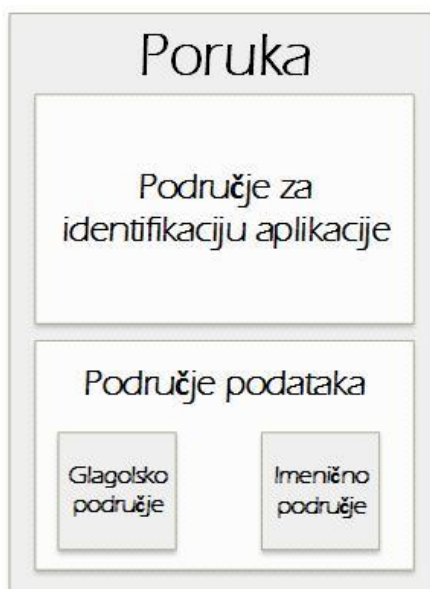
2.5.4 Deo 4 - Objektni modeli i atributi sistema proizvodnih operacija

Ovaj deo pruža detaljne modele i protok informacija između kategorija i aktivnosti unutar sistema proizvodnih operacija. Cilj mu je da standardizuje protok informacija unutar sistema proizvodnih operacija (MES). Prema ISA-i, obim četvrtog dela ograničen je na definisanje objektnih modela i atributa koji se razmenjuju između aktivnosti definisanih u trećem delu. Četvrti deo se još uvek u procesu razvijanja.

2.5.5 Deo 5 - Proizvodne i poslovne transakcije

Deo 5 detaljno razrađuje razmenu informacija koje opisuju prvi i drugi deo. Delovi 1 i 2 jasno pokazuju koje informacije ERP i MES sistemi moraju međusobno razmenjivati. Delovi 1 i 2 su standardizovali strukturu **poruke**, ali ne idu dalje u ono što sistem koji prima podatke treba da uradi sa podacima. Da li bi trebalo da kreira, menja ili briše podatke? Komitet ISA-95 standarda bavio se ovim pitanjem i kao rešenje razvio peti deo standarda. Peti deo precizira kako sistem slanja i prijema treba

da komunicira sa informacijama iz delova 1 i 2. U tom cilju, peti deo definiše nekoliko **transakcija**. Transakcije ISA-95 standarda su, razgovori između ERP i MES sistema o tipičnim ISA-95 subjektima. Transakcije se grade iz poruka. Ako transakciju posmatramo kao razgovor, tada se ISA-95 **poruka** može uporediti sa rečenicom koju izgovara jedan od sistema. Prijemni sistem „čuje“ tu rečenicu i odgovara drugom rečenicom. U školi smo naučili da rečenica mora uvek sadržati glagol i imenicu. ISA-95 poruke takođe imaju propisani sadržaj, čiji su deo glagoli i imenice. Poruke ISA-95 podrazumevano sadrže **područje za identifikaciju aplikacija** i **područje podataka** (Slika 2.7).



Slika 2.7 - Tipični skup razmenjenih podataka

Područje za identifikaciju aplikacije sadrži podatke koji su potrebni sistemima da bi znali odakle dolazi poruka, gde odgovor treba poslati, da li i drugi sistem takođe želi potvrdu, i tako dalje. **Područje podataka** sadrži glagol i imenicu. Kombinacija glagola i imenice daje jedinstvenu komandu, poput *GET equipment ID 201*, u kojoj je *GET* glagol, a *equipment ID 201* imenica. Imenice su podaci (objekti) koji su već navedeni u objektnim modelima u delovima 1 i 2 standarda, dok su glagoli novina. Koji će se glagol primeniti, zavisi od toga da li sistemi međusobno razmenjuju informacije u skladu sa Pull modelom, Push modelom ili Publish modelom.

U **Pull modelu**, jedan sistem zahteva informacije od drugog sistema. Sistem koji zahteva informacije koristi glagol *GET*, kao u primeru *GET equipment ID 201*. Sistem koji mora da dostavi podatke odgovara glagolom *SHOW*, kao u primeru *SHOW equipment ID 201*. U **Push modelu**, jedan sistem šalje podatke samoinicijativno drugom sistemu. Koristi glagole *PROCESS*, *CHANGE* ili *CANCEL*. Na primer, ERP sistem može poslati poruku *PROCESS Schedule 02-04-2021* MES sistemu. Sistem MES-a tada može odgovoriti *ACKNOWLEDGE – ACCEPTED Schedule 02-04-2021*. Treći način na koji sistemi mogu međusobno komunicirati je preko **Publish modela**. U ovom modelu sistem šalje podatke jednom ili više sistema. Podaci su takoreći „objavljeni“ (uporedite sa televizijom i radiom), a sistem koji šalje poruku u mnogim situacijama ne zna da li će primajući sistemi učiniti bilo šta sa porukom. Sistem koji šalje poruku objavljuje da je kreirao, promenio ili izbrisao podatke pomoću glagola *SYNC ADD*, *SYNC DELETE*, *SYNC CHANGE*. Na primer *SYNC DELETE equipment ID 201*.

Peti deo ISA-95 standarda opisuje kako za svaku kombinaciju glagola i imenica dobijamo koju radnju poruka treba da generiše.

2.6 Prednosti korišćenja ISA-95 standarda

ISA-95 standard se može koristiti kao metoda za definisanje interfejsa između sistema poslovne logistike i kontrolnih sistema. Primenom ovog standarda troškovi se mogu drastično smanjiti. Integracija rešenja različitih dobavljača postaće manje složena kada svi budu koristili ovaj standard.

Pomaže pri unapređenju komunikacije između različitih kompanija. Svaka proizvodna kompanija koristi sopstvenu terminologiju za opisivanje funkcija, aktivnosti i odeljenja u kompaniji. Kada morate da radite sa spoljnim konsultantima, komunikacija će biti teška. Velika je šansa da ćete te pričati o različitim stvarima kada koristite iste izraze ili obrnuto. Problem se uvećava svaki put kada se započne novi projekat ili svaki put kada se razgovara sa različitim klijentima. Dakle, kada se razgovara o interfejsima, bilo bi dobro da se diskusija zasniva na standardizovanoj terminologiji tako da obe strane ostvare zajednički jezik.

Integracija između sistema poslovne logistike i kontrolnih sistema pre ISA-95 za vreme pravljenja nekog projekta je trajala između 1-2 godine sa procentom od 50% i manje uspešnosti. Nakon korišćenja ISA-95 standarda vremenski period se smanjio na 2-4 meseca sa procentom od 90% i više uspešnosti. Takođe definisanje specifikacija između ova dva sistema pre ISA-95 standarda je trajalo preko godinu dana, dok uz korišćenje ovog standarda taj vremenski period se prepolovio i iznosio je 6 meseci u nekim slučajevima i manje. Na osnovu ove statistike možemo da utvrdimo da je ISA-95 standard bio veliki uspeh.

ISA-95 standard nije ograničen na stvaranje interfejsa između sistema poslovne logistike i kontrolnih sistema, kako sugerise osnovna namena standarda. To je takođe dobra smernica za sastavljanje korisničkih zahteva, opisivanje funkcionalnih zahteva, razvijanje MES aplikacija i baza podataka, analiza i upoređivanje kapaciteta na različitim proizvodnim mestima i sticanje uvida u optimizaciju proizvodnih procesa. Modeli i terminologija ISA-95 standarda osigurava da svi ljudi koji koriste ovaj standard razmišljaju i razgovaraju na isti način o proizvodnim aktivnostima, sistemima i informacijama [5].

3. TEHNOLOGIJA I ALATI

U ovom poglavlju pričaćemo o svim tehnologija i alatima koji su korišćeni za implementaciju rada.

3.1 .NET Framework

Okruženje za razvoj sofvera, razvijano od strane Microsoft-a za Windows platforme. Uključuje veliku biblioteku klasa (*Framework Class Library*). Microsoft je sa razvojem .NET-a počeo ranih 1990-tih, pod nazivom *Next Generation Windows Services*. Početkom 2000-tih prva beta verzija .NET 1.0 je objavljena, a u avgustu 2000. u saradnji sa Intel-om i HP-om, Microsoft je počeo sa standardizacijom *CLI-ja*, koja će omogućiti izvršavanje različitih programskih jezika na različitim arhitekturama-platformama.

Programi se izvršavaju kroz softversko okruženje *CLR*, virtualnu mašinu koja sadrži: *memory managment, exception handling, garbage collector...* Omogućeno je korišćenje 25 programskih jezika od kojih su najpopularniji C#, C++ i VisualBasic. Jezici se, svaki preko svog kompajlera, kompajliraju u *CIL* među-jezik. Zatim, u zavisnosti od toga na kojoj se platformi izvršava, *CLR* kompajlira *CIL* u mašinski kod. Glavni razvojni alat je Visual Studio [6].

3.2 Microsoft Visual Studio

Predstavlja integrirano razvojno okruženje. Koristi se za razvoj računarskih programa za Windows, veb-stranica, aplikacija i usluga. Koristi Microsoft-ove platforme za razvoj raznih API-ja za Windows, Windows Forms, WPF. Program takođe sadrži alate poput dizajnera oblika koji se koristi za pravljenje aplikacija s grafičkim korisničkim interfejsom, veb-dizajnera, dizajnera klasa i dizajnera shema baza podataka. Visual Studio podržava različite programske jezike i dozvoljava uređivaču koda i debugger-u da podržava gotovo bilo koji programski jezik. Ugrađeni jezici su C, C++, VB.NET, C# i F#. Također podržava XML, HTML, JavaScript i CSS [7].

3.3 C#

Objektno orijentisan programski jezik koji je razvio Microsoft početkom 21-og veka. Reč je o jeziku opšte namene koji služi za pravljenje aplikacija u okviru .NET okruženja. Iako ne postoji dugo kao neki drugi programski jezici, C# je jedan od najpopularnijih jezika. C# se odlikuje velikim mogućnostima, jednostavnošću upotrebe i lakoćom usvajanja, zbog čega je danas jedan od najpopularnijih programskih jezika koji svoju primenu nalazi u velikim i malim kompanijama i u različitim oblastima [8].

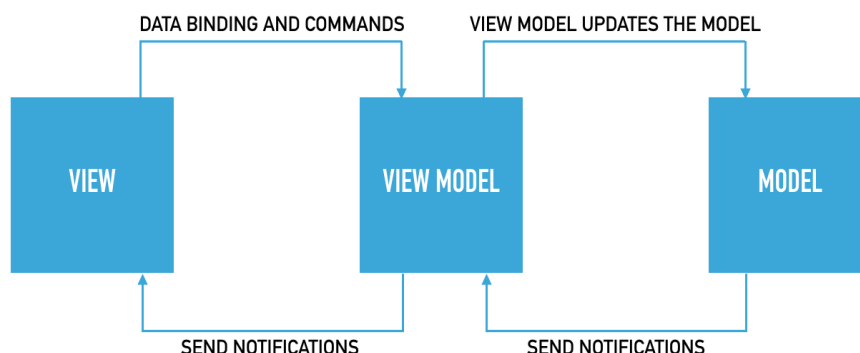
3.4 Windows Presentation Form (WPF)

Grafički podsistem za renderovanje korisničkog interfejsa u aplikacijama zasnovanim na Windows operativnim sistemima [9]. Razvijen je od strane Microsoft-a. WPF koristi XAML, izveden od XML-a da definiše i poveže različite UI elemente. WPF aplikacije mogu biti razvijene kao samostalni desktop programi ili kao ugrađeni objekti u website stranicama. Ima za cilj da objedini niz zajedničkih interfejs elemenata, kao što su 2D/3D renderovanja, fiskirana i adaptivna dokumenta, tipografiju, vektorsku grafiku...

3.5 Model-View-ViewModel (MVVM)

Patern koji razdvaja aplikaciju na više komponenti tako da svaka komponenta ima svoje specifične odgovornosti. MVVM arhitektura je preporučena od strane Google-a kao jedan od najboljih načina strukture koda Android aplikacija [10]. Pri korišćenju MVVM paterna kod aplikacije je razdvojen na tri dela:

- *View* - Ova sekcija sadrži klase (Aktivnosti i Fragmenti) koje su zadužene za prikaz interfejsa i prihvatanje akcija korisnika, nakon čega o tome obaveštava *ViewModel*
- *ViewModel* - Ova sekcija sadrži klase koje su zadužene za pristup podacima (Repository) i da obaveste *View* ukoliko dolazi do promena.
- *Model* - Ova sekcija sadrži klase zadužene za pristup raznim vrstama podataka (baza, webservice...) i da izvrši abstrakciju takvih izvora podataka kroz jedan API.



Slika 3.1 - MVVM patern

3.6 Windows Communication Fondation (WCF)

Servisno orijentisani model razmene poruka, koji omogućava programima da komuniciraju preko računarske mreže ili lokalno. WCF je alat koji u sebi uključuje set biblioteka razvijenih za distribuirano programiranje [11].

3.7 Microsoft SQL Server

Predstavlja relacijsku bazu podataka kojoj je primarni jezik za upite *Transact SQL* (T-SQL), što znači da osim osnovnih i klasičnih (SELECT tipa) SQL upita dozvoljava i složenije stvari poput *if* naredni ili *while* petlji. *Transact SQL* nastao je kao plod suradnje Microsoft-a i Sybase-a. SQL server je baza podataka koja je namenjena manjim i srednjim bazama.

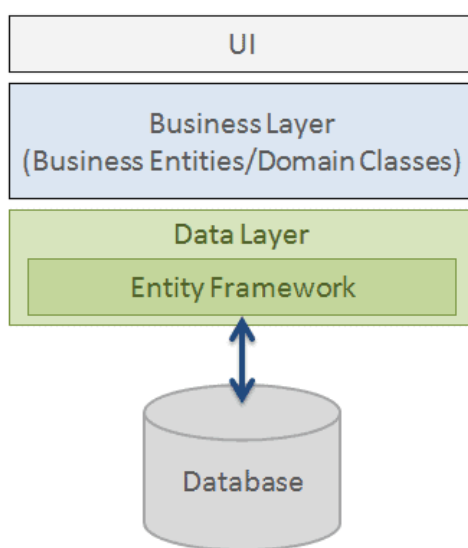
Prva verzija SQL Servera koji ima veze sa Microsoft-om izašla je 1989. godine pod imenom „SQL Server for OS/2 1.0“. Ta verzija bila je identična Sybase-ovom SQL serveru 3.0 koji je radio pod Unix sistemom. Microsoft SQL Server pod tim imenom se počeo prodavati 1992. godine, a puno ime mu je glasilo Microsoft SQL Server 4.2 koji se još uvek vrtio na OS/2 platformi. Prvi SQL Server za *Windows* operativne sisteme izašao je isto kada i sami Windows-i.

SQL Server je prva baza podataka na svetu koja je posedovala korisnički interfejs (*UI*). Sve baze su u tadašnje vreme radile pomoću „command-line“ sistema koji zna biti izrazito nezgodan [12].

3.8 Entity Framework

Entity Framework predstavlja *ORM Framework* otvorenog koda za .NET koje podržava Microsoft. Omogućava programerima da rade sa podacima koristeći objekte klase bez fokusiranja na osnovne tabele i kolone baze podataka u kojima se ti podaci čuvaju. Uz Entity Framework, programeri mogu raditi na višem nivou apstrakcije kada se bave podacima i mogu stvoriti i održavati aplikacije sa manje koda.

Na slici 3.2 možemo primetiti da se Entity Framework uklapa između poslovnih entiteta i baze podataka. Sprema podatke pohranjene u svojstvima poslovnih entiteta, a takođe preuzima podatke iz baze podataka i automatski ih pretvara u objekte poslovnih entiteta [13].



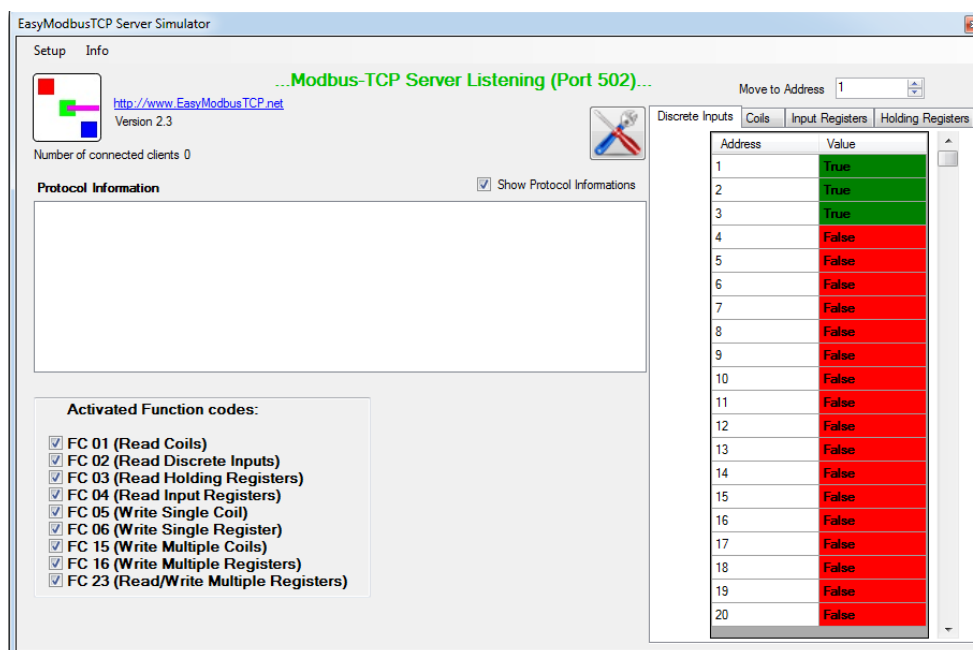
Slika 3.2 - Pozicija Entity Framework-a u arhitekturi sistema

3.9 Easy Modbus Server Simulator

Modbus je protokol za prenos podataka koji se koristi zajedno sa *PLC*-om. Modbus je postao standard za komunikacijski protokol i sada je glavno sredstvo za povezivanje industrijskih elektronskih uređaja. Razvijen je za industrijske aplikacije, relativno je jednostavan za primenu i održavanje u poređenju sa drugim standardima [14].

Easy Modbus Server Simulator je dizajniran da pomogne programerima koji koriste Modbus protokol. Svi Modbus podaci su prikazani u Server Simulatoru i omogućava debugovanje grešaka klijentskih aplikacija. Podržava operacije čitanja i pisanja. Server Simulator podržava Modbus *TCP*, Modbus *UDP* i Modbus *RTU* u .NET verziji. Pomoćni kodovi funkcija mogu biti onemogućeni. Meni svojstva omogućava promenu protokola između Modbus *TCP*-a, *UDP*-a i *RTU*-a [15].

Vrednosti sa kojima Server Simulator radi su: *Coils*, *Discrete inputs*, *Input registers*, *Holding registers*. Prve dve vrednosti predstavljaju diskretne vrednosti, dok zadnje dve su analogne.



Slika 3.3 - Easy Modbus Server Simulator

3.10 Dizajn paterni

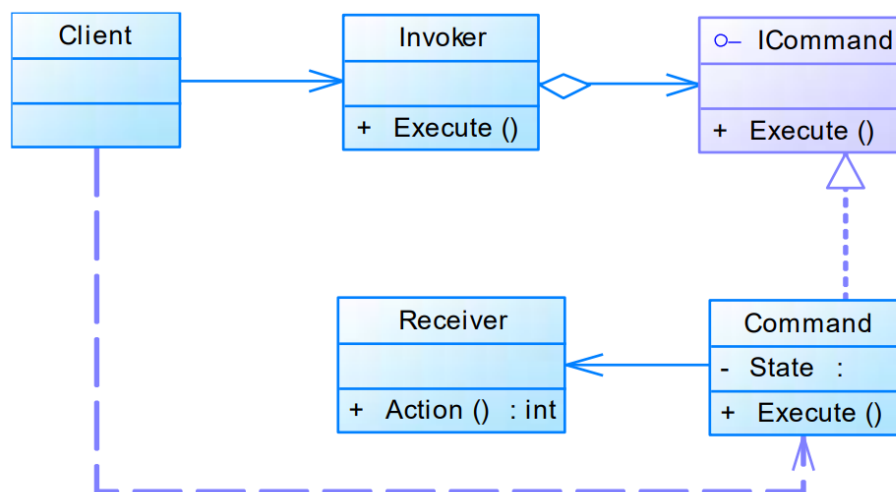
U softverskom inženjerstvu, *dizajn patern* ili *šablon* ili *obrasac* je opšte, ponovo upotrebljivo rešenje za česte probleme koji se sreću prilikom projektovanja softvera. *Dizajn patern* nije gotov dizajn koji se može direktno pretvoriti u izvorni kod. On služi samo kao opis ili šablon prilagođen da reši neki opširniji problem u posebnom kontekstu [16]. Postoje tri vrste *dizajn patern*a:

- *Strukturalni paterni* - bave se kompozicijom i obično predstavljaju različite načine za definisanje odnosa među objektima. Oni obezbeđuju da kada je neophodna promena u jednom delu sistema, ostatak sistema ne mora da se menja. Takođe pomažu da svaki deo sistema radi ono čemu je najbolje prilagođen. Neki od *strukturalnih patern*a su: Decorator, Facade, Flyweight, Adapter i Proxy.
- *Kreacioni paterni* - ovi paterni bave se kreacijom objekata, na način prilagođen određenoj primeni. Posebno su važni u situacijama u kojima bi uobičajen pristup kreiranju objekata doveo do povećanja kompleksnosti projekta. Neki od paterni koji spadaju u ovu grupu su: Constructor, Factory, Prototype, Singleton i Builder.
- *Bihevioralni paterni* - Ova grupa paterni tiče se poboljšanja komunikacije između različitih objekata u sistemu. Poznati primeri su: Strategy, Iterator, Mediator, Observer i Visitor.

Dizajn paterni koji su korišćeni u implementaciji rada biće nabrojavi i objašnjeni u tekstu ispod.

3.10.1 Command

Command patern kreira distancu između klijenata koji zahtevaju operacije i objekata koji ih izvršavaju. Patern je izrazito višestran. On može da podrži: slanje zahteva ka različitim objektima, smeštanje zahteva u redove, logovanje i odbijanje zahteva [17].



Slika 3.4 - Command patern

U okviru UML dijagrama možemo videti više učesnika čije su uloge sledeće:

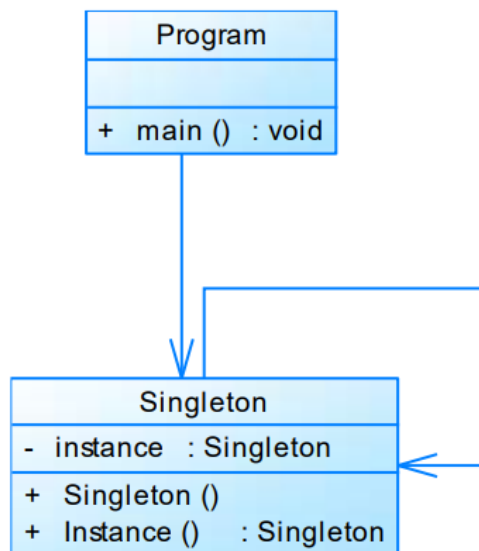
- *Client* - kreira i izvršava komande
- *ICommand* - interfejs koji navodi operacije Execute koje se mogu izvršiti
- *Invoker* - poziva klasu Command da izvrši određenu akciju
- *Command* - klasa koja implementira Execute operaciju tako što uključuje operacije iz klasa Receiver
- *Receiver* - klasa koja može da izvrši zahtevanu akciju
- *Action* - operaciju koju je potrebno izvršiti

Command patern na prvi pogled ima puno učesnika, ali se neki od njih odbacuju kada se koriste delegati.

Razlog uvođenja paternu: Ako želimo da implementiramo pravi *MVVM* patern, potrebno je odvojiti svu poslovnu logiku iz *View*-a i smestiti u *ViewModel*. Kada se doda neko dugme u *View*-u za to dugme je potrebno uraditi *binding* (vezivanje). Sa tim *binding*-om kažemo da će sva poslovna logika prenesti u *ViewModel* i tamo će se pozvati. Takođe mora postojati implementirana metoda na *ViewModel*-u, koja će biti pozvana preko klikanja na dugme. Ta metoda predstavlja komandu. Sam taj *binding* prestavlja akciju na koju se trigeruje komanda.

3.10.2 Singleton

Singleton patern ograničava instanciranje klase i osigurava da samo jedna instanca date klase postoji i pruža globalnu tačku pristupa ka toj instanci. Patern osigurava da je klasa instancirana samo jednom i da su svi zahtevi upućeni ka tom jednom i samo jednom objektu [18].



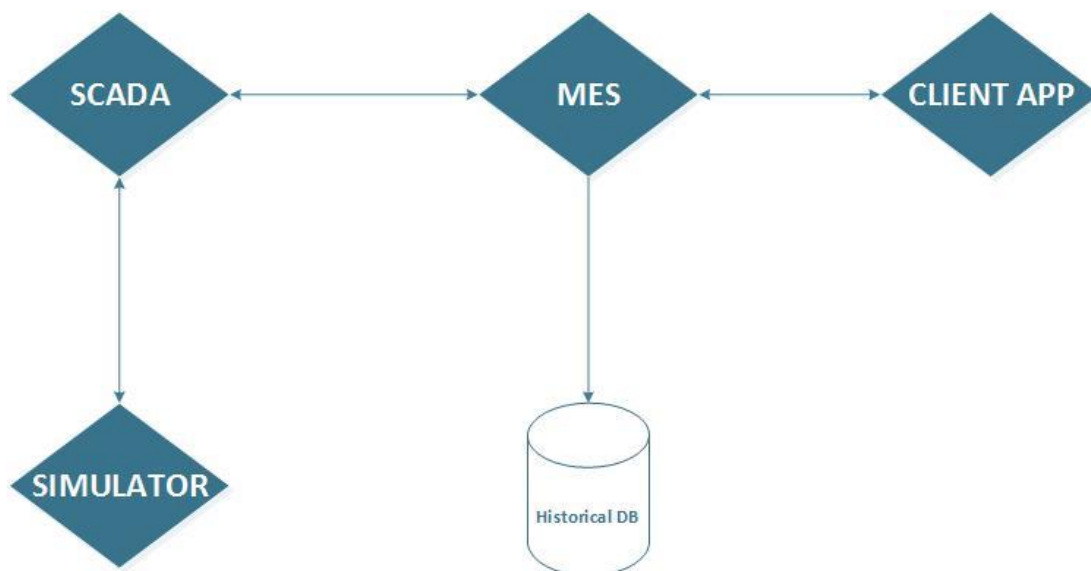
Slika 3.5 - Singleton patern

UML dijagram klasa koji predstavlja *singleton* patern je prikazan na Slika 3.5. Sa slike se može videti da kreiramo samo klasu *Singleton* koja obezbeđuje da će biti kreirana samo jedna njena instanca.

Razlog uvođenja paterna: U radu *Singleton* i *MVVM* su na neki način usko povezani. Kada pravimo neki *View* u i ako znamo da će objekti unutar tog *View* biti dinamični, onda se dešava problem da tokom dizajniranja nemamo uvid u izgled *View* zbog dinamičnih objekata. Rešenje toga jeste pravljenje jednog dizajn *ViewModel*-a u kojem možemo definisati statičke objekte i tako imati uvid tokom dizajniranja. Povezivanje tog dizajn *ViewModel*-a i *View* pored korišćenja *MVVM* koristi se i *Singleton* patern.

4. ARHITEKTURA SISTEMA

Aplikacija sistema je podeljena na nekoliko komponenti. Svaka komponenta ima neka svoja zaduženja i procese koje obavlja nezavisno od ostatka sistema i drugih komponenti. Arhitektura sistema i komunikacija između komponenti je prikazana na slici 4.1.



Slika 4.1 - Arhitektura sistema

Komponente sistema predstavljaju:

1. *Simulator* - Simulator predstavlja third-party aplikaciju koja služi za simuliranje vrednosti elemenata u polju. Ostatak teksta o Simulatoru možemo pronaći u poglavlju 3.9.
2. *SCADA* - je servis koji vrši periodičnu akviziciju stanja vrednosti na simulatoru koristeći MODBUS protokol. Kada primi trenutno stanje, šalje se razlika trenutnog i prethodnog stanja na MES servis na interpretaciju.
3. *MES* - servis koji prima podatke sa SCADA-e i interpretira njihovo značenje. Dužan je da proveri da li se neki od uređaja nalazi u alarmnom stanju i da zapisuje podatke u SQL bazu podataka (na slici Historical DB). Za to vreme se pristigli podaci proslede na klijentskoj aplikaciji. Ako se utvrdi da je neki uređaj u alarmnom stanju, dodatne informacije se šalju klijentskoj aplikaciji. Na zahtev klijenta generiše izveštaje i šalje ih nazad.
4. *Historical DB* - predstavlja SQL bazu podataka u kojoj se čuva stanje sistema. Takođe čuvaju se svi događaji koji su se desili u sistemu tj. sva alarmatna stanja. Na zahtev MES komponente, Historical DB vraća tražene vrednosti.
5. *Client App* - korisnik za interakciju sa sistemom ima na raspolaganju korisnički interfejs. Funkcionalnosti mu omogućavaju manipulaciju uređajima koji se nalaze u sistemu, prikaz izveštaja, kao i pregled alarmatnih stanja u sistemu.

U ovom radu fokus je bio na komandovanju opreme u polju preko klijentske aplikacije ili preko simulatora, poštujući ISA-95 standarde i koncepte. Na osnovu toga implementirani su svi nivoi ISA-95 standarda. U sistemu postoji simulator koji simulira rad prekidača u polju. SCADA vrši periodičnu akviziciju stanja vrednosti na simulatoru i te vrednosti šalje ka MES-u. MES obavlja biznis logiku sistema, čuva istorijske događaje u bazi podataka, generiše izveštaje... Pored toga postoji i klijentska aplikacija preko koje možemo da komandujemo sa prekidačima, dobijamo povratne vrednosti u slučaju komandovanja preko simulatora u vidu alarma, kao i mogućnost ispisivanje izveštaja.

U narednom poglavlju objasnićemo detaljan rad aplikacije i njenih komponenti.

5. IMPLEMENTACIJA APLIKACIJE

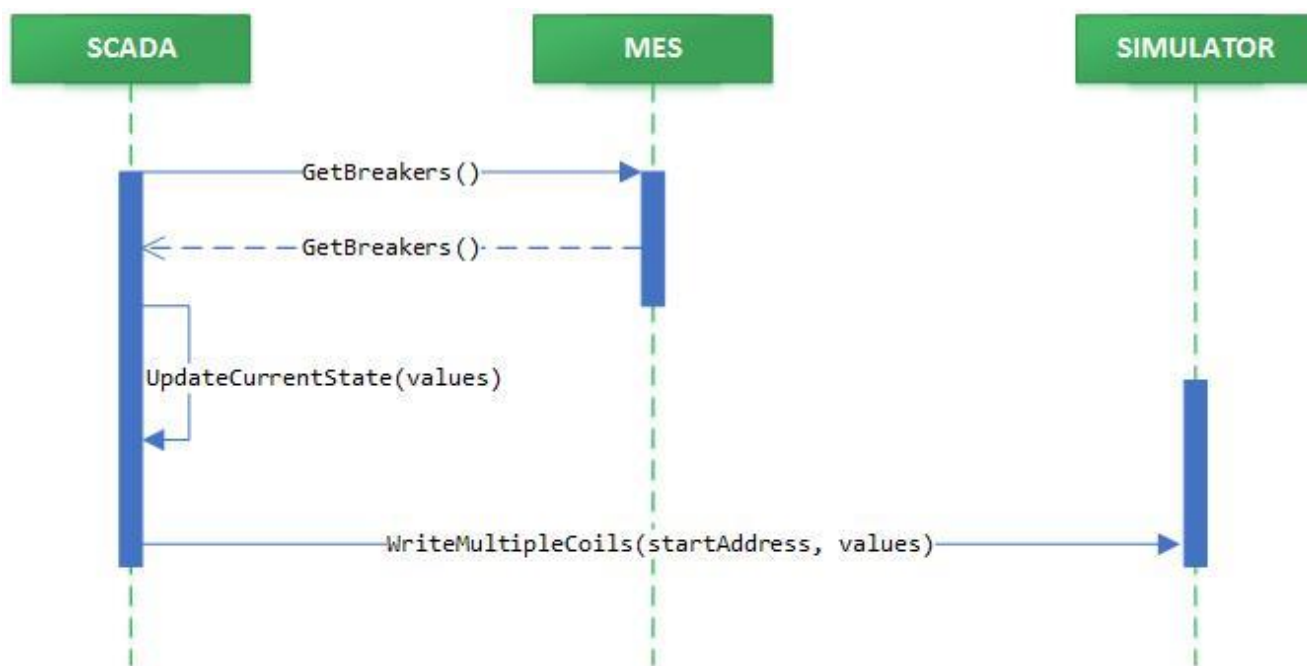
U ovom poglavlju biće objašnjena implementacija i rad sledećih komponenti aplikacije:

5.1 SCADA

5.1.1 Inicijalno pokretanje

Da bi se ostvarile visoke performanse *SCADA* komponente, potrebno je ovaj servis što više olakšati kad su u pitanju procesi koji se dešavaju u njemu. Želimo da *SCADA* servis ima što manje poslova koje treba da obavlja. Razlog ostvarivanja visokih performansi jeste to da želimo da *SCADA* što pre očitava nove vrednosti sa simulatora i te vrednosti prosledi dalje u sistem da bi se krajni korisnik mogao obavestiti o novonastalim promenama. Zbog ovakvih uslova, *SCADA* ne ostvaruje pristup istorijskim podacima niti vrši neku biznis logiku sistema.

Nakon prvog pokretanja *SCADA* servisa, potrebno je prethodno sačuvano stanje prekidača upisati u simulator. Pošto *SCADA* ne sadrži te vrednosti, ona mora da ih zatraži od servisa koji poseduje te vrednosti tj. od *MES*-a. Otvara se komunikacioni kanal prema *MES*-u i traži se trenutno stanje prekidača. Nakon prijema vrednosti, prvo se ažuriraju vrednosti u *SCADA* memoriji, a zatim se i ažurira stanje u simulatoru. Na slici 5.1 možemo da vidimo dijagram sekvenci ove opisane akcije. Kada se završi ažuriranje stanja, *SCADA* može da nastavi sa svojim primarnim zadatkom, a to je akvizicija podataka sa simulatora.



Slika 5.1 - Inicijalni start *SCADA* servisa

SCADA poseduje svoju lokalnu bazu podataka, ali u toj bazi postoji samo jedna tabela pod nazivom **CoilsAddress**. Značenje te tabele predstavljeno je na tabeli 5.1. Razlog postojanja ove baze jeste da bi se optimizovala akvizicija podataka sa simulatora. Iz baze možemo dobiti informaciju o minimalnoj i maksimalnoj vrednosti korišćene adrese. Na osnovu te informacije možemo da smanjimo krug prekidača za koje je potrebno vršiti akviziciju. Bez ove informacije morali bi vršiti akviziciju svih adresa sa simulatora, bez obzira da li se dešava promena ili ne.

Address (int)	Označava adresu na simulatoru
Id (int)	Označava jedinstveni identifikacioni broj prekidača u sistemu
Used (bool)	Označava da li je neki prekidač zauzeo mesto na datoj adresi u simulatoru

Tabela 5.1 - Prikaz redova tabele CoilsAddress

5.1.2 Akvizicija podataka sa simulatora

Pošto se stanje prekidača u polju tj. simulatoru može konstantno menjati, mora se periodično proveravati njihovo stanje. SCADA periodično, koristeći MODBUS protokol, proziva simulator i „pita“ kakvo je trenutno stanje u polju. Od simulatora dobije informaciju o novom stanju prekidačke opreme. Po prijemu novog stanja primljena je informacija o stanju svakog prekidača u polju. Međutim, tu se nalaze i prekidači kojima se nije promenilo stanje, a čak može da se desi i da se nijednom nije promenilo stanje. Slika 5.2 pokazuje kako se utvrđuje gde je došlo do promena. Ovom metodom se smanjuje opterećenje ostalih komponenti kojima se prosleđuje stanje.

	1	2	3	4	5	6	7	8
Trenutno stanje	0	1	1	0	0	0	0	1
	1	2	3	4	5	6	7	8
Novo stanje	0	1	0	0	1	1	0	0
	1	2	3	4	5	6	7	8
Razlika	0	1	0	0	1	1	0	0

Slika 5.2 - Ilustracija izračunavanja razlike stanja prekidača

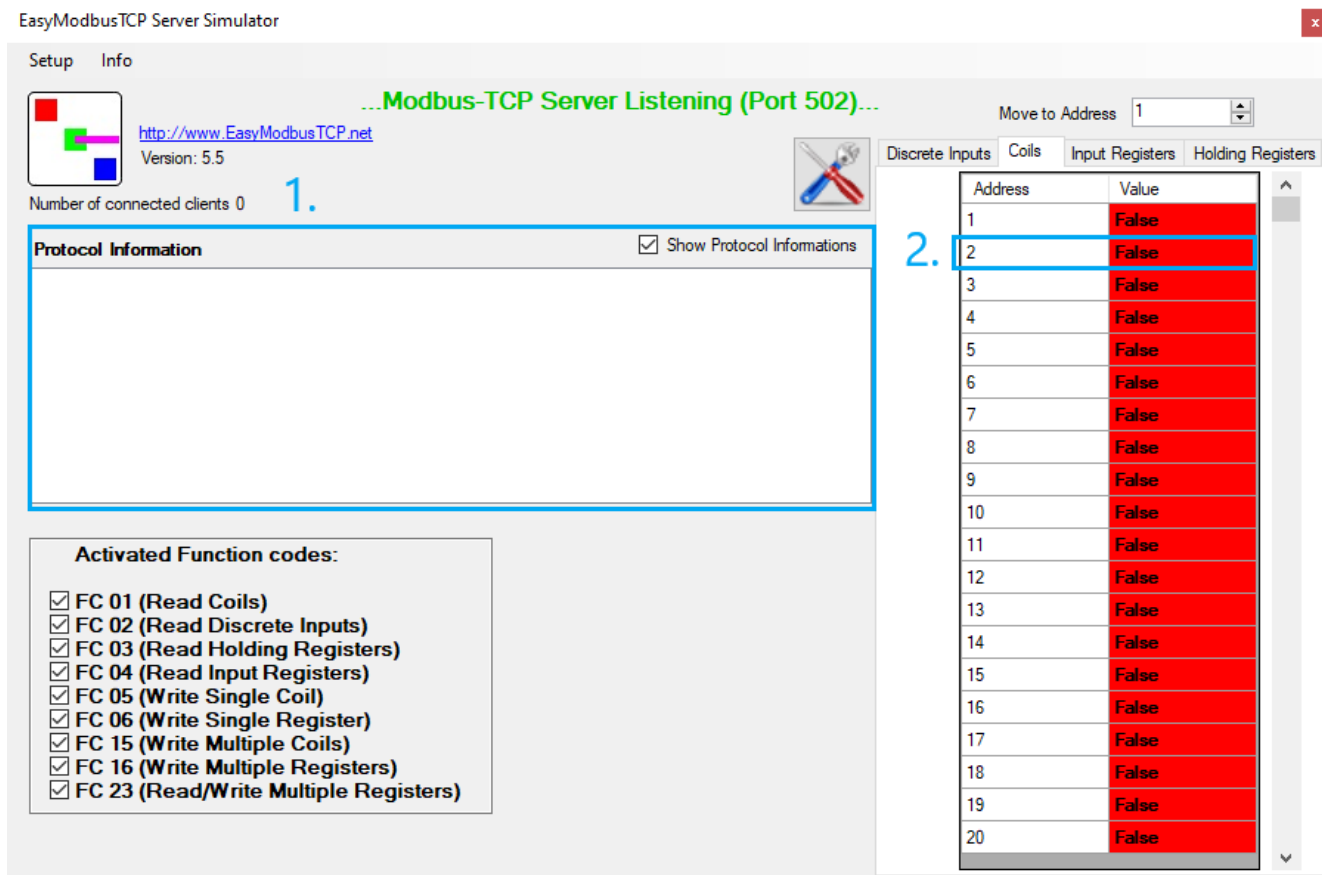
Po prijemu novog stanja, trenutno i novo stanje se proslede u funkciju koja utvrđuje razliku između ta dva stanja. Utvrđivanje se vrši tako što se izvrši logička operacija **ekskluzivno ili** između njih da bi se dobili indeksi izmenjenih vrednosti. Izlaz funkcije je novo stanje sa označenim promenjenim vrednostima. Nakon toga se novo stanje sačuva kao trenutno stanje i promenjene vrednosti pošalju na MES komponentu.

5.1.3 Komandovanje

Kada je reč o komandovanju SCADA radi samo ono što joj je zadano. Tu ne postoji nikakva logika. Od MES dobija instrukcije i te instrukcije prosleđuje ka simulatoru. Te instrukcije predstavljaju otvaranje/zatvaranje prekidača. Pošto u sistemu postoje prekidači, svaki prekidač u sebi sadrži jedinstveni identifikacioni broj. MES pošalje na SCADA sistem instrukciju otvori/zatvori prekidač i identifikacioni broj prekidača. Pošto SCADA u sebi sadrži listu **CoilsAddress** na osnovu toga možemo da odredimo koji prekidač na kojoj adresi u simulatoru treba da otvorimo/zatvorimo.

5.2 Simulator

Za simulaciju sistema koristimo Easy Modbus Server Simulator, sa kojim vršimo komandovanje opreme u polju. Sve izmene nastale u simulatoru, šalju se na SCADA servis preko Modbus TCP protokola. Simulator iza sebe nema nikakvu logiku, njegov posao je da “simulira” tj. samo vrši otvaranje/zatvaranje opreme u polju. On nema uvid o kakvoj je to opremi reč, kakve su specifikacije opreme, naziv, primena... Zbog ovakvog načina rada, simulator vrši komandovanje u veoma kratkom vremenskom roku, u milisekundama.

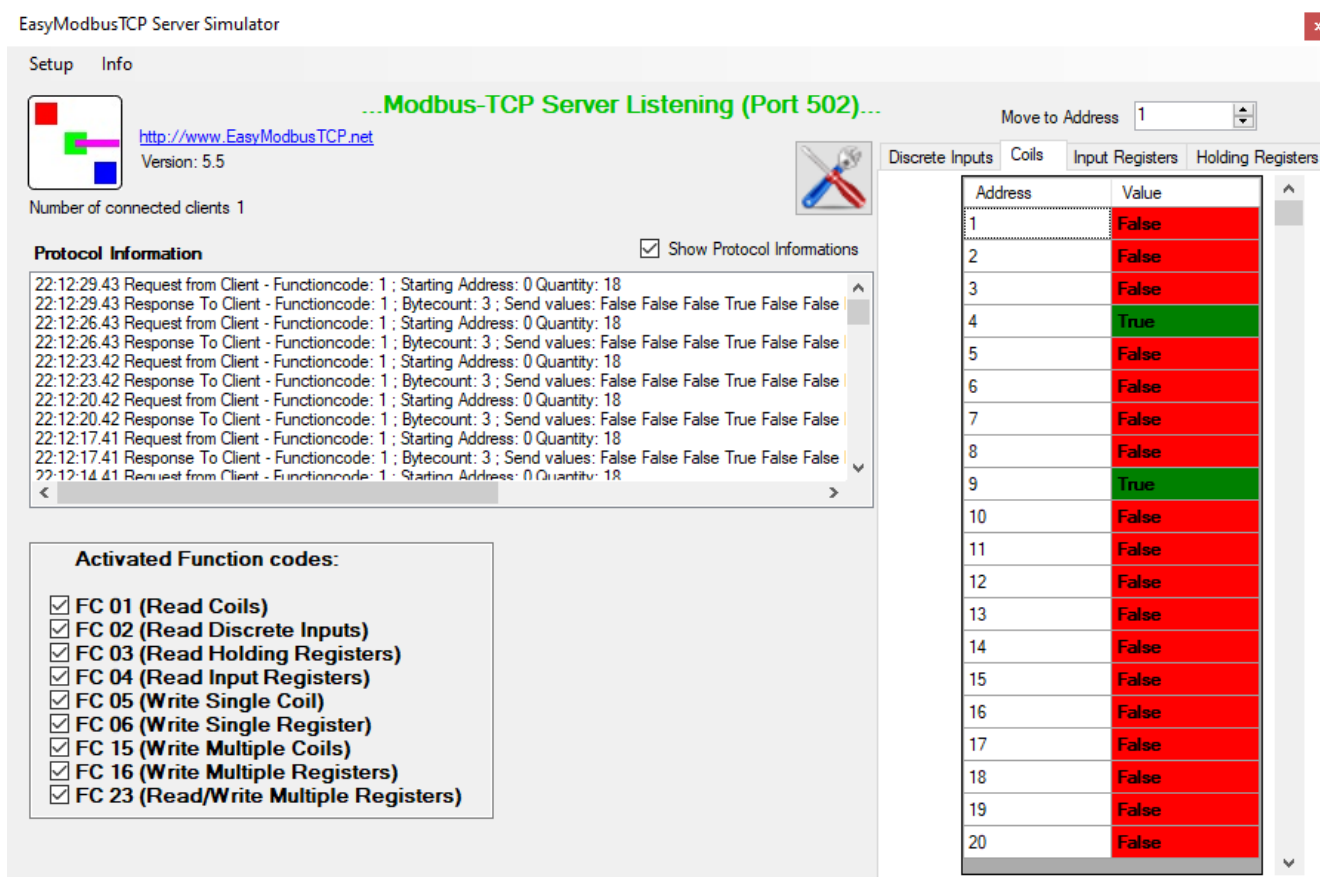


Slika 5.3 - Simulator pre starta SCADA servisa

Na slici 5.3 vidimo inicijalno stanje simulatora nakon prvog paljenja. Možemo da primetimo par bitnih prozora unutar simulatora:

1. Informaciona tabla – ovde informacije o vremenu kada je simulator primio podatke od SCADA servisa i kada je poslao vrednosti na SCADA servis
2. Tab Coils sa mogućim adresama za komandovanje. Vrednosti koje su moguće su otvoreno/zatvoreno tj. **true/false**.

Nakon startovanja SCADA servisa i kad se proslede vrednosti na simulator, izgled možemo da vidimo na slici 5.4. Informaciona tabla se popunila sa raznim informacijama, dok neke adrese na tab Coils-u su promenile svoju vrednosti. Komandovanje se vrši tako što kursor miša postavimo na željenu adresu i dvoklikom potvrdimo tu akciju.



Slika 5.4 - Simulator nakon starta SCADA servisa

5.3 MES

5.3.1 Inicijalno pokretanje

Nakon inicijalnog pokretanja *MES* servisa, pokreće se inicijalizacija modela tako što se čitaju podaci iz baze podataka i na osnovu podataka kreiraju se objekti koji se smeštaju u model. U sistemu objekti koji se mogu inicijalizovati su prekidači i alarmi. Pošto alarme koristimo samo kada generišemo izveštaje i kada se vrši obrada akvizicije i u oba ta slučaja vršimo upis i čitanje iz baze podataka. Tako da njih nije potrebno smeštati u model, dok kod prekidača je druga stvar. Da ne bi na svaku obradu akvizicije vadili podatke iz baze podataka i tako vršili proveru koji prekidač je promenio stanje, brže i lakše je da ti podaci nam se nalaze u modelu.

Znači nakon inicijalnog pokretanja izvlačimo podatke o prekidačima iz baze pomoću *Entity Framework*-a i čuvamo ih u listama koje se nam se nalaze u modelu. Vrednosti i polja koja karakterišu prekidač u sistemu možemo videti na tabeli 5.2.

Id (<i>int</i>)	Označava jedinstveni identifikacioni broj prekidača u sistemu
Name (<i>nvarchar(max)</i>)	Označava naziv prekidača
Current state (<i>bool</i>)	Označava trenutno stanje prekidača
Last state (<i>bool</i>)	Označava prethodno stanje prekidača

Tabela 5.2 – Prikaz redova tabele prekidača

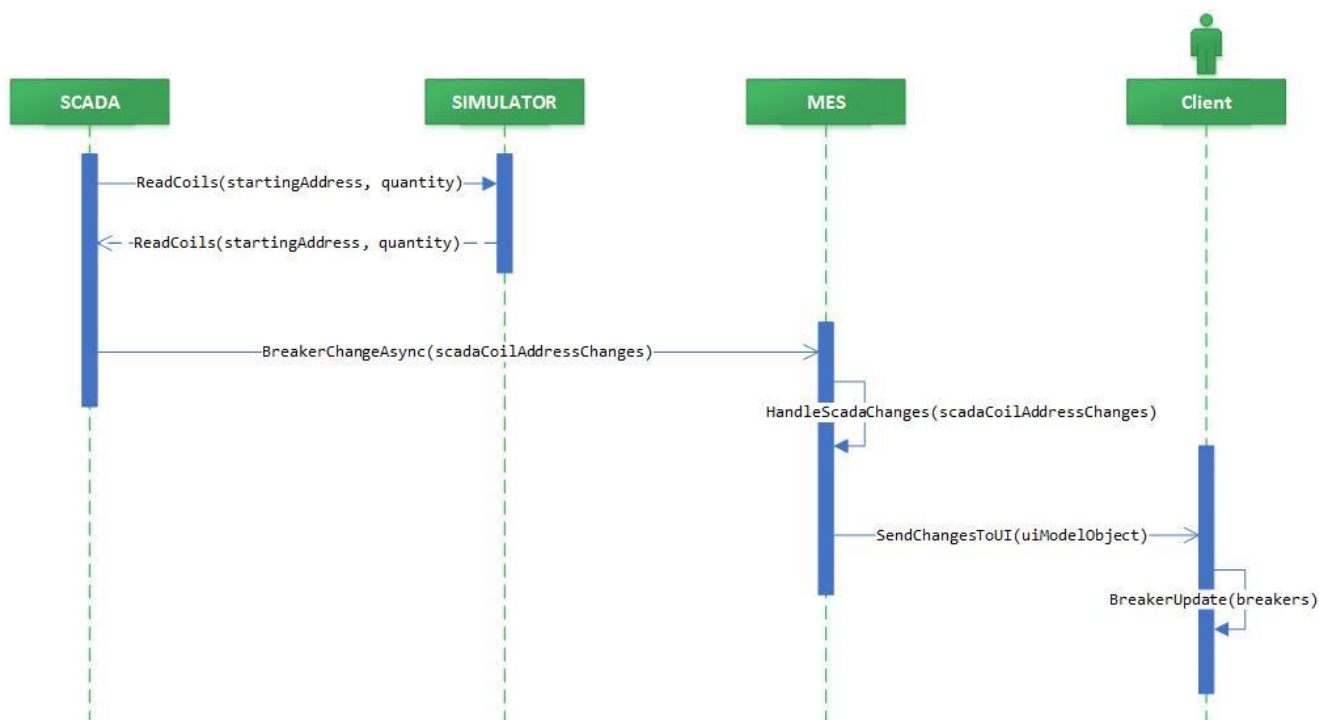
5.3.2 Obrade akvizicije

Po prijemu podataka obrađenih u poglavlju 5.1.2 potrebno je da se ti podaci interpretiraju i da im se da neka semantika. Pošto je SCADA dizajnirana da nema nikakvu biznis logiku, odgovornost za tu logiku je prebačena na MES.

Obrada se sastoji iz tri segmenta:

1. Čuvanje stanja pristiglih vrednosti
2. Upisivanje stanja u bazu podataka
3. Obaveštavanje klijenta o promenama

Ova akcija je opisana preko dijagrama sekvenci na slici 5.5.



Slika 5.5 - Obrada akvizicije

5.3.2.1 Čuvanje stanja pristiglih vrednosti

Od SCADA servisa stiže informacija na kom ID-u prekidača je koje novo stanje. Za dobijeni ID se iz MES modela uzima konkretan objekat prekidača u kom se nalaze podaci potrebni za obradu. U slučaju da prekidač ne postoji u modelu, baca se izuzetak i tako se zna da SCADA i MES nisu u konzistentnom stanju. Nakon dobijanja konkretnog prekidača njegovo stanje se ažurira. Treba primetiti da je ovo samo u radnom modelu i da još nije završilo u istorijskoj bazi podataka.

5.3.2.2 Upisivanje stanja u bazu podataka

Obradeni podaci se moraju čuvati u istorijskoj bazi iz razloga što se istorija koristi za:

- Inicijalizaciju MES servisa
- Kreiranje izveštaja

Zbog performansi podaci ne završavaju odmah u bazi podataka, nego se upisuju u liste. Količina podataka u jednoj listi zavisi od pristiglih promena. Kad se obrade sve pristigle promene, vrši se upis u istorijsku bazu podataka i liste se nakon toga očiste da bi se nove promene mogle primenjivati.

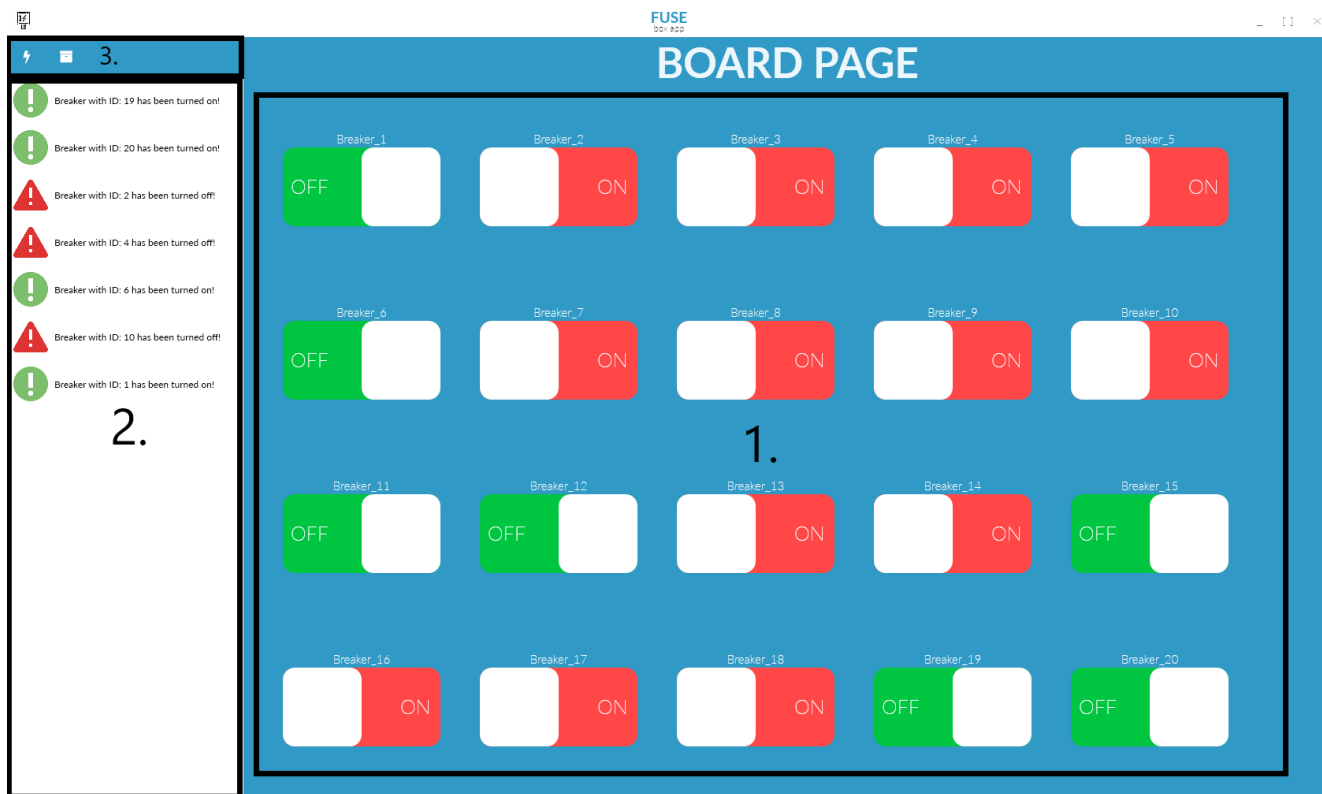
5.3.2.3 Obaveštavanje klijenta o promenama

Odmah nakon obrade podataka, promene se „pakuju“ u format podataka poznat klijentu tj u objekat **UIModelObject**. U taj objekat se pakuju svi afektovani prekidači, kao i generisani alarmi vezani za njih. Slanje je asinhrono.

5.4 Klijentska aplikacija

Preko klijentske aplikacije, korisniku se omogućava nekoliko akcija kao što su: komandovanje prekidača, uvid u alarme i generisanje izveštaja. Na slici 5.6 možemo da vidimo izgled aplikacije, takođe možemo da primetimo nekoliko različitih prozora:

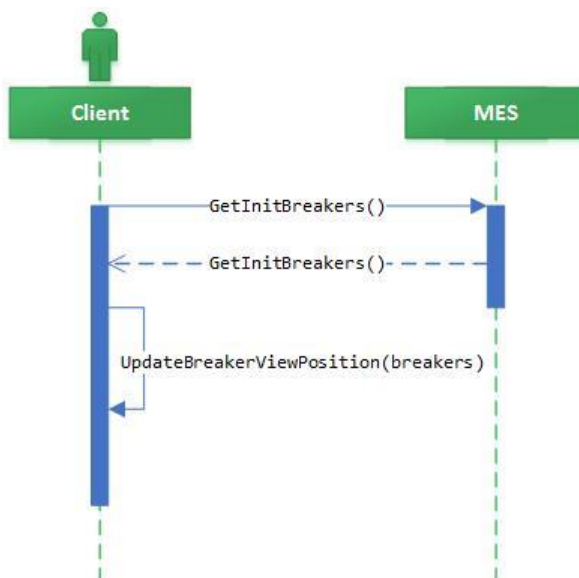
1. Glavni prozor
2. Lista sa alarmima
3. Tab pomoću kojeg menjamo glavni prozor. Dozvoljeni prozori su tabela sa prekidačima i prozor za generisanje izveštaja



Slika 5.6 - Klijentska aplikacija

5.4.1 Inicijalno pokretanje

Nakon prvog pokretanja, klijentska aplikacija traži od MES-a svu postojeću opremu u sistemu sa njegovom trenutnom vrednošću. Što bi značilo da trebaju prekidači sa vrednostima da li je otvoren/zatvoren dati prekidač. Nakon prijema podataka, klijentska aplikacija vrši ažuriranje glavnog prozora. Prave se objekti u obliku dugmeta za svaki prekidač, gleda se koja mu je trenutna vrednost. Ako je prekidač zatvorenog kola, beli kvadratić unutar dugmeta će prekriti ispis **ON** i pozadina dugmeta će preći u zelenu. U slučaju da je prekidač otvorenog kola, beli kvadratić će ovaj put prekriti ispis **OFF** i pozadina dugmeta biće crvene boje. Na slici 5.7 možemo da vidimo dijagram sekvenci ove opisane akcije. Nakon ažuriranja glavnog prozora, može se nastaviti sa daljim akcijama koje klijentska aplikacija podržava.

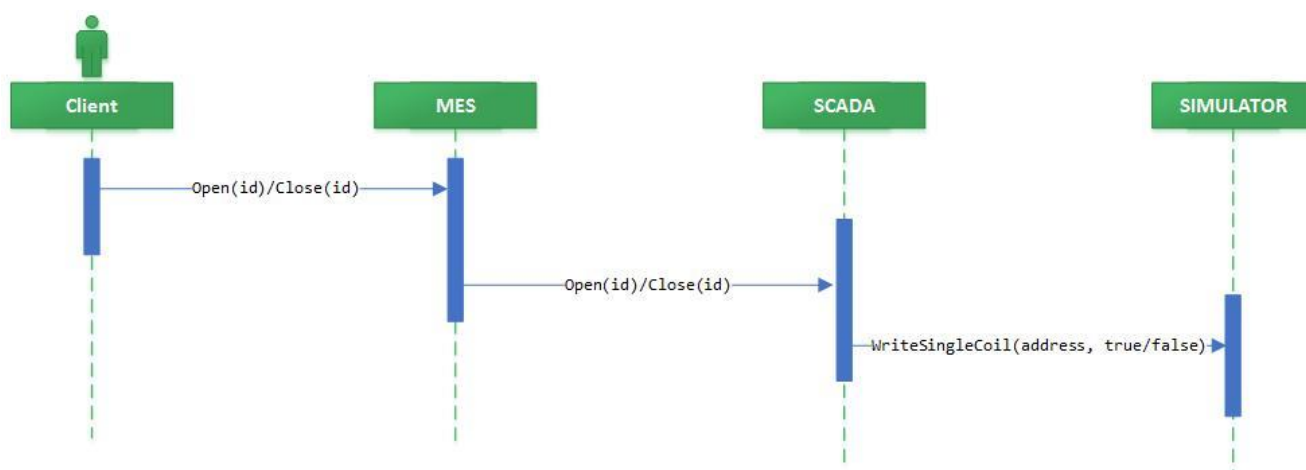


Slika 5.7 - Inicijalni start klijentske aplikacije

5.4.2 Komandovanje

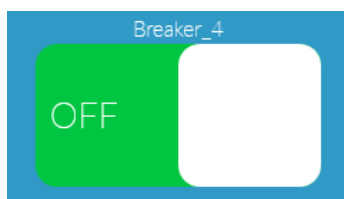
Komandovanje je akcija koja prolazi kroz sve komponente sistema. Akcija počinje na klijentskoj aplikaciji kliktanje miša na prekidač. Klikom na željeni prekidač šaljemo na *MES*, id prekidača koji želimo da komandujemo. Na osnovu njegovog trenutnog stanja tj. da li je otvoren ili zatvoren pozivamo odgovarajuće funkcije. Zatim *MES* informacije prosleđuje na *SCADA* servis. *SCADA* zatim na osnovu id-a iz modela traži adresu prekidača i tu adresu zajedno sa komandom šalje na simulator.

Ova akcija je opisana dijagramom sekvenci koja se nalazi na slici 5.8.

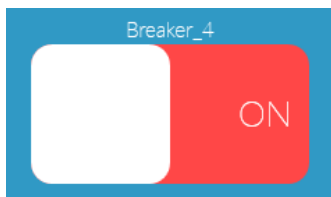


Slika 5.8 - Komandovanje preko klijentske aplikacije

Komandovanje prekidača na klijentskoj aplikaciji se vrši tako što se kursor miša postavi na dugme prekidača i levim klikom miša potvrdimo akciju. Da bi se akcija izvršila miš je potrebno postaviti ili na dugme ili na beli kvadratić unutar dugmeta. Ako je prekidač pre izvršenja akcije bio u zatvorenom kolu, on sada prelazi u otvoreno kolo, dugme prelazi iz zelene boje u crvenu, a beli kvadratić sa desne strane prelazi na levu. Ako je prekidač pre izvršenja akcije bio u otvorenom kolu, postupak se ponavlja ali u obratnom smeru. Na slici 5.9 i slici 5.10 možemo da vidimo izgled dugmadi sa klijentske aplikacije kada su ona u zatvorenom i otvorenom kolu.



Slika 5.9 - Zatvoren prekidač



Slika 5.10 - Otvoren prekidač

5.4.3 Alarmi

Alarmi u implementaciji aplikacije predstavljaju odziv na događaje u sistemu. Pod tim podrazumevamo na svaku akciju komandovanja prekidača, bilo to sa klijentske aplikacije ili sa

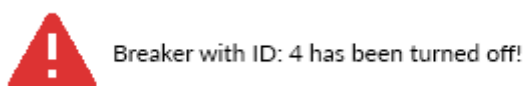
simulatora, se generiše novi alarm. Time želimo da obavestimo krajnjeg korisnika o promenama u sistemu. Ovo ima posebnu važnost kada je reč o komandovanju sa simulatora, jer preko alarma možemo korisnika obavestiti o promenama u sistemu.

Objekat alarm sadrži nekoliko polja i atributa koja ga opisuju, te vrednosti su opisane u tabeli 5.3.

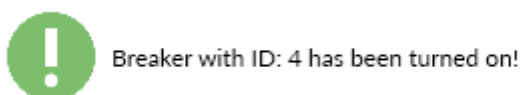
Id (int)	Označava jedinstveni identifikacioni broj alarma u sistemu
BreakerId (int)	Označava jedinstveni identifikacioni broj prekidača za koji je dati alarm vezan
Timestamp (datetime2)	Označava vreme kada se desila akcija koja je pokrenula generisanje alarma
Message (nvarchar(max))	Označava dodatni opis alarma

Tabela 5.3 - Prikaz redova tabele alarma

Na klijentskoj aplikaciji postoji poseban tab u kome su alarmi smešteni i vidljivi korisniku. Taj odsek je objašnjen u poglavlju 5.4. Izgled alarma može imati dva slučaja, kada je prekidač prešao u otvoreno kolo tj. kada je isključen (Slika 5.11) i kada je u zatvorenom kolu tj. kada je uključen (Slika 5.12). Izgled alarma se sastoji od ikonice koja zavisi od statusa prekidača. Ako je prekidač isključen ikonica je u obliku trougla, crvene boje je i u sredi ima znak upitnika, u slučaju da je prekidač uključen ikonica je u tom slučaju kružnog oblika, zelene boje i sa znakom upitnika u sredi. Pored ikonice se nalaz i dodatni opis alarma. Poruka u kojoj piše u kom je trenutnom statusu prekidač.

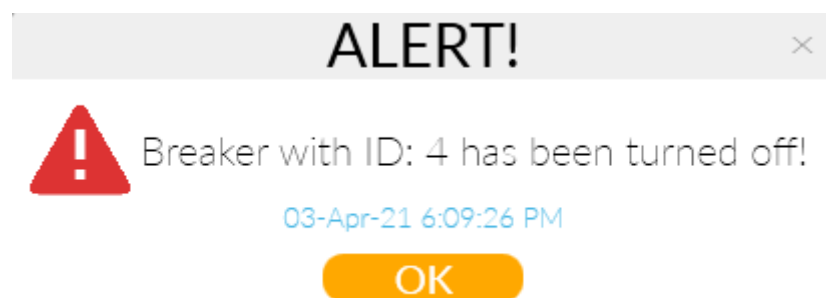


Slika 5.11 - Izgled alarma kada je prekidač isključen

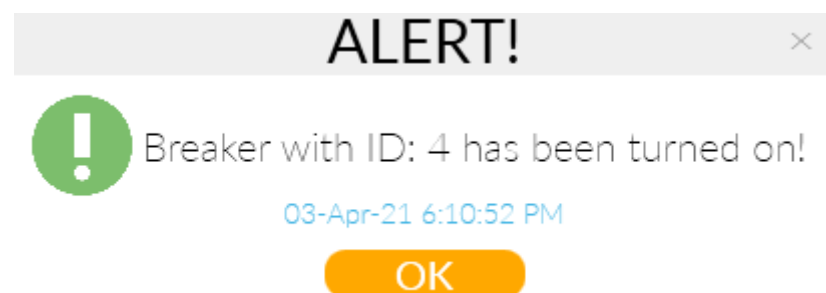


Slika 5.12 - Izgled alarma kada je prekidač uključen

Ako želimo neke dodatne informacije koje su vezane za alarme, onda moramo postati kursor miša na alarm i akciju potvrditi levim klikom miša. Odgovor te akcije javlja se u obliku novog mini prozora koji zamrzava glavni prozor sve dok se taj mini prozor ne ugasi. Mini prozor prati neki šablonski izgled glavnog prozora i u sebi sadrži sve informacije kao i alarm sa glavnog prozora. Pored toga kao dodatna informacija je vreme generisanja datog alarma. Vreme je opisano u formatu: *Dan – Mesec – Godina (Sati:Minute:Sekunde)*. Izlaz iz ovog prozora i povratak na glavni prozor je moguć levim klikom miša na dugme **OK** ili na dugme **X**. Izgled ovog prozora, možemo da vidimo na slikama 5.13 i 5.14.



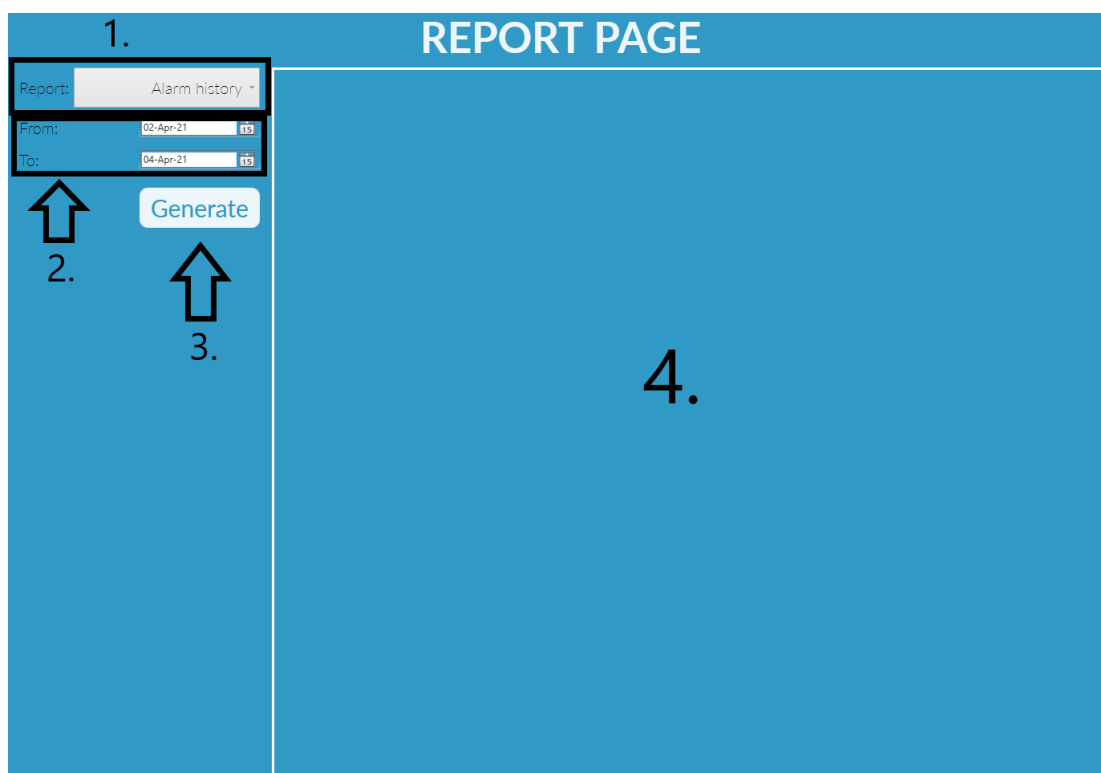
Slika 5.13 - Izgled prozora alarma sa dodatnim informacijama kada je prekidač isključen



Slika 5.14 - Izgled prozora alarma sa dodatnim informacijama kada je prekidač uključen

5.4.4 Izveštaji

Report ili izveštaj je dokument koji predstavlja informacije koje su organizovane u formatu za određenu svrhu. Iako se sažeci izveštaja mogu dostavljati usmeno, potpuni izveštaji su gotovo uvek u obliku pisanih dokumenata. Do prozora sa izveštajima se dolazi levim klikom miša na ikonicu kutije koja se nalazi u levom gornjem uglu glavne stranice. Izgled glavne stranice i ikonice kutije za izveštaje je objašnjeno u poglavlju 5.4. Klikom prozor sa prekidačima nestaje i pojavljuje se prozor za izveštaje. Izgled ovog prozora je opisan na slici 5.15.



Slika 5.15 - Izgled prozora sa izveštajima

Na slici vidimo nekoliko bitnih elemenata koji pomažu da lakše i što preciznije generišemo željeni izveštaj:

1. Combox sa nazivima izveštaja koje sistem podržava
 - a. Alarm History – istorija alarma
 - b. Current Equipment – svi trenutni prekidači u sistemu
2. Birači datuma za pomoću kojih se određuje vremenski opseg generisanja alarma
3. Dugme pomoću kojeg potvrđujemo opcije
4. Glavni prozor sa podacima izveštaja

Izveštaj istorija alarma (*Alarm History*) ispisuje sledeće kolone:

- ID prekidača za koji je alarm vezan
- Vreme generisanja alarma
- Dodatni opis

Na slici 5.16 se nalazi izgled glavnog prozora sa datim izveštajem.

REPORT PAGE			
Breaker ID	Timestamp	Message	
1	05-Mar-21 1:35:39 PM	Breaker with ID: 1 has been turned on!	
4	05-Mar-21 1:36:07 PM	Breaker with ID: 4 has been turned on!	
1	05-Mar-21 1:36:21 PM	Breaker with ID: 1 has been turned off!	
10	05-Mar-21 1:36:31 PM	Breaker with ID: 10 has been turned on!	
10	05-Mar-21 1:36:33 PM	Breaker with ID: 10 has been turned off!	
9	02-Apr-21 10:11:42 PM	Breaker with ID: 9 has been turned on!	
2	03-Apr-21 12:01:09 AM	Breaker with ID: 2 has been turned on!	
2	03-Apr-21 12:01:15 AM	Breaker with ID: 2 has been turned off!	
9	03-Apr-21 12:01:33 AM	Breaker with ID: 9 has been turned off!	
4	03-Apr-21 12:01:39 AM	Breaker with ID: 4 has been turned off!	
1	03-Apr-21 12:01:45 AM	Breaker with ID: 1 has been turned on!	
12	03-Apr-21 12:01:57 AM	Breaker with ID: 12 has been turned on!	
5	03-Apr-21 12:02:03 AM	Breaker with ID: 5 has been turned on!	
1	03-Apr-21 12:02:09 AM	Breaker with ID: 1 has been turned off!	
10	03-Apr-21 12:02:25 AM	Breaker with ID: 10 has been turned on!	
15	03-Apr-21 12:02:32 AM	Breaker with ID: 15 has been turned on!	
5	03-Apr-21 12:02:52 AM	Breaker with ID: 5 has been turned off!	
11	03-Apr-21 12:13:05 AM	Breaker with ID: 11 has been turned on!	
1	03-Apr-21 4:54:13 PM	Breaker with ID: 1 has been turned on!	
2	03-Apr-21 4:54:19 PM	Breaker with ID: 2 has been turned on!	
1	03-Apr-21 4:54:26 PM	Breaker with ID: 1 has been turned off!	
19	03-Apr-21 6:09:06 PM	Breaker with ID: 19 has been turned on!	
20	03-Apr-21 6:09:06 PM	Breaker with ID: 20 has been turned on!	
2	03-Apr-21 6:09:08 PM	Breaker with ID: 2 has been turned off!	
4	03-Apr-21 6:09:08 PM	Breaker with ID: 4 has been turned on!	
6	03-Apr-21 6:09:10 PM	Breaker with ID: 6 has been turned on!	
10	03-Apr-21 6:09:10 PM	Breaker with ID: 10 has been turned off!	
1	03-Apr-21 6:09:16 PM	Breaker with ID: 1 has been turned on!	
1	03-Apr-21 6:09:20 PM	Breaker with ID: 1 has been turned off!	
1	03-Apr-21 6:09:26 PM	Breaker with ID: 1 has been turned on!	
4	03-Apr-21 6:09:26 PM	Breaker with ID: 4 has been turned off!	
4	03-Apr-21 6:10:52 PM	Breaker with ID: 4 has been turned on!	

Slika 5.16 - Izveštaj istorije alarma

Izveštaj svih trenutnih prekidača u sistemu (*Current Equipment*) ispisuje sledeće kolone:

- ID prekidača
- Naziv prekidača
- Trenutno stanje prekidača
- Prethodno stanje prekidača

Na slici 5.17 se nalazi izgled glavnog prozora sa datim izveštajem.

REPORT PAGE

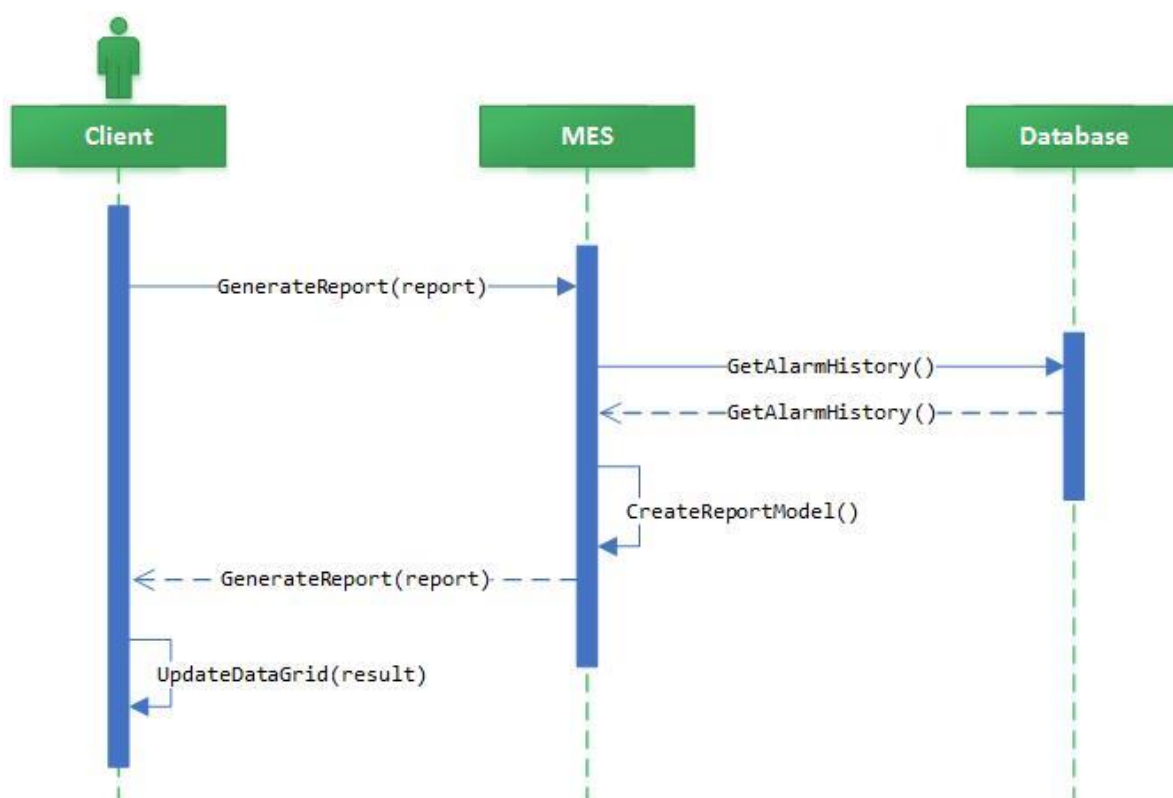
Report: Current equipment ▾

Generate

ID	Name	Current state	Last state
1	Breaker_1	True	False
2	Breaker_2	False	True
3	Breaker_3	False	False
4	Breaker_4	True	False
5	Breaker_5	False	True
6	Breaker_6	True	False
7	Breaker_7	False	False
8	Breaker_8	False	False
9	Breaker_9	False	True
10	Breaker_10	False	True
11	Breaker_11	True	False
12	Breaker_12	True	False
13	Breaker_13	False	False
14	Breaker_14	False	False
15	Breaker_15	True	False
16	Breaker_16	False	False
17	Breaker_17	False	False
18	Breaker_18	False	False
19	Breaker_19	True	False
20	Breaker_20	True	False

Slika 5.17 - Izveštaj svih trenutnih prekidača u sistemu

Način na koji se izveštaji generišu jeste taj da korisnik preko klijentske aplikacije unese odgovarajuće parametre za generisanje izveštaja. Parametri se šalju na *MES* servis koji zatim na osnovu željenog izveštaja, izvlači iz baze podataka odgovarajuće podatke. Zatim te podatke “pakuje” u **ReportModel** objekat i vraća ih nazad klijentskoj aplikaciji. Kad ti podaci pristignu na klijentsku aplikaciju vrši se ispis na glavni prozor. Ilustraciju opisane akcije možemo da vidimo na slici 5.18.



Slika 5.18 - Generisanje izveštaja

6. ZAKLJUČAK

Aplikacija koja je opisana i testirana u ovom radu predstavlja simulaciju komandovanja opreme u polju pomoću simulatora ili preko implementirane klijentske aplikacije, primenjujući principe i smernice ISA-95 standarda. Objašnjen je način rada aplikacije, kao i uloga svih komponenti i delova sistema.

Neke od prednosti korišćenja ovakvog standarda tokom implementacije neke aplikacije predstavlja **nezavisna implementacija svake komponente sistema**, gde svaka komponente sistema je implementirana nezavisno od ostalih komponenti. Ovakva vrsta prednosti omogućava da SCADA sistem vrši periodičnu akviziciju podataka sa simulatora, dok klijent traži od MES-a uviđaj u istorijske podatke sistema u obliku izveštaja. U principu, rad jedne komponente ne utiče na rad druge komponente. **Apstrakcija** predstavlja još jednu vrstu prednosti ovakvog sistema. Primer možemo predstaviti u obliku komandovanja prekidača sa klijentske aplikacije. Kada vršimo komandovanje nas, kao krajnjeg korisnika ne zanima cela logika iza te akcije, koje sve funkcije obavlja i na koji način ih obavlja. Nas samo zanima krajnji rezultat, a to je prebacivanje prekidača iz jednog stanja u drugo. Pomoću apstrakcije to je moguće. Pomoću ovakvog sistema **komunikacija** između učesnika (komponenti) sistema je olakšana. Potrebno je definisati interfejs i portove za komunikaciju da bi ona bila uspešna. U slučaju kada se sistem širi i kada se dodavaju neke nove funkcije sistema ukoliko želimo da sve komponente budu koizistente i obaveštene, potrebno je postojeće interfejse proširiti i otvoriti nove portove za komunikaciju. Korišćenje ovakvog načina komuniciranja, omogućeno je komponentima sistema pričanje na zajedničkom "jeziku".

Takođe uvek postoji prostora za napredak i usavršavanje što je upravo slučaj i sa ovim radom. Određeni segmenti aplikacije bi mogli biti rešeni na bolji i efikasniji način. Jedan od mogućih unapređenja ili izmena je da klijenta aplikacija bude urađena u nekoj drugoj tehnologiji koja nije WPF npr. Windows Form ili ASP.NET. Kada sistem krene da se ponaša kako ne treba, mogla bi se implementirati logging ideja praćenja event-ova. U logove bi se upisivali greške sistema za koje korisnik ne bi trebao da zna. Npr. zašto neki servis ne radi.

Još jedna vrsta unapređenja aplikacije je mogla biti uvođenje opreme sa analognim signalima. Pod tim se podrazumeva uvođenje neke vrste brojača ili merila koja može da prati vreme rada prekidača i te vrednosti da ispisuje u simulator i na klijentsku aplikaciju.

Dodatno moguće unapređenje sistema javlja se u obliku podržavanja višekorisničke aplikacije, gde bi mogli otvoriti željeni broj korisničkih aplikacija. Uticaj komandovanja sa jedne aplikacije bi uticalo da sve otvorene aplikacije u sistemu, dok generisanje izveštaja bi bila individualna stvar. Jedna od tehnologija koja bi pomogla u implementaciji ovakvog unapređenja je *Pub/Sub* mehanizam.

7. LITERATURA

- [1] Bianca Scholten, *Road to Integration*, ISBN-13: 978-0979234385
- [2] TechTarget, *Definition, ANSI/ISA-95*, datum pristupa: 25/03/2021
<<https://searcherp.techtarget.com/definition/ANSI-ISA-95>>
- [3] Siemens, *ISA 95 Framework & Layers*, datum pristupa: 26/03/2021
<<https://www.plm.automation.siemens.com/global/en/our-story/glossary/isa-95-framework-and-layers/53244>>
- [4] Intraratio, *3 Key Differences Between ERP and MES Systems*, datum pristupa: 26/03/2021
<<https://intraratio.com/post/key-differences-between-erp-and-mes>>
- [5] ISA-95, *Overview of advantages*, datum pristupa: 27/03/2021 <<https://isa-95.com/advantages>>
- [6] Guru99, *What is Microsoft .Net Framework?*, datum pristupa: 27/03/2021
<<https://www.guru99.com/net-framework.html>>
- [7] GeeksforGeeks, *Introduction to Visual Studio*, datum pristupa: 27/03/2021
<<https://www.geeksforgeeks.org/introduction-to-visual-studio>>
- [8] Jon Skeet, *C# in Depth, 3rd Edition*, ISBN-13: 978-1617291340
- [9] WPF Tutorial, *What is WPF?*, datum pristupa: 27/03/2021 <<https://www.wpf-tutorial.com/aboutwpf/what-is-wpf>>
- [10] Wintellect, *Model-View-ViewModel (MVVM) Explained*, datum pristupa: 27/03/2021
<<https://www.wintellect.com/model-view-viewmodel-mvvm-explained>>
- [11] Microsoft, *What Is Windows Communication Foundation*, datum pristupa: 28/03/2021
<<https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf>>
- [12] Wikipedia, *Microsoft SQL Server*, datum pristupa: 28/03/2021
<https://wikipedia.org/wiki/Microsoft_SQL_Server>
- [13] Entity Framework Tutorial, *What is Entity Framework?*, datum pristupa: 28/03/2021
<<https://www.entityframeworktutorial.net/what-is-entityframework.aspx>>
- [14] Wikipedia, *Modbus*, datum pristupa: 28/03/2021 <<https://wikipedia.org/wiki/Modbus>>
- [15] Easy Modbus TCP, *Modbus Server Simulator*, datum pristupa: 28/03/2021
<<http://easymodbustcp.net/modbus-server-simulator>>
- [16] Source Making, *Design Patterns*, datum pristupa: 28/03/2021
<https://sourcemaking.com/design_patterns>
- [17] Source Making, *Command Design Pattern*, datum pristupa: 28/03/2021
<https://sourcemaking.com/design_patterns/command>
- [18] Source Making, *Singleton Design Pattern*, datum pristupa: 28/03/2021
<https://sourcemaking.com/design_patterns/singleton>

PODACI O KANDIDATU

Kandidat Nenad Zelenović je rođen 01.04.1995. godine u Somboru. Osnovnu školu "Nikola Tesla" u Bačkom Brestovcu završio je 2010. godine. Završio je srednju ekonomsku školu u Somboru 2014. godine. Fakultet Tehničkih Nauka u Novom Sadu je upisao 2014. godine, smer Elektroenergetski Softverski Inženjering. 2019. godine završava osnovne studije i upisuje master akademske studije na Fakultetu Tehničkih Nauka, smer Primenjeno Softversko Inženjerstvo. Ispunio je sve obaveze i položio sve ispite predviđene studijskim programom.