



УНИВЕРЗИТЕТ У НОВОМ САДУ
**ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У
НОВОМ САДУ**



Дејан Субашић

Инкрементални импортер за ЦИМ модел

ДИПЛОМСКИ РАД
- Основне 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 документација
Тип записа, ТЗ:	Текстуални штампани материјал
Врста рада, ВР:	Завршни (Bachelor) рад
Аутор, АУ:	Дејан Субашић
Ментор, МН:	др Немања Недић
Наслов рада, НР:	Инкрементални импортер за CIM модел
Језик публикације, ЈП:	Српски / латиница
Језик извода, ЈИ:	Српски
Земља публикавања, ЗП:	Република Србија
Уже географско подручје, УГП:	Војводина
Година, ГО:	2021
Издавач, ИЗ:	Ауторски репринт
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, ФО: (поглавља/страна/ цитата/табела/слика/графика/прилога)	6/24/15/0/11/0/0
Научна област, НО:	Електротехника и рачунарство
Научна дисциплина, НД:	Примењене рачунарске науке и информатика
Предметна одредница/Кључне речи, ПО:	Smart Grid, дистрибутивни менаџмент системи, Заједнички информациони модел
УДК	
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, ВН:	
Извод, ИЗ:	У овом раду приказује се једна имплементација система за увид у елементе паметне мреже. Паметни системи се развијају последњих неколико деценија и њихова намера је употреба рачунарства и информационих технологија у побољшању ефикасности и спектра могућих услуга. Детаљно су приказани имплементациони детаљи имплементiranог пројекта и пример сценарио за сваку услугу.
Датум прихватања теме, ДП:	
Датум одбране, ДО:	
Чланови комисије, КО:	Председник: др Дарко Чапко
	Члан: др Срђан Вукмировић
	Члан, ментор: др Немања Недић
	Потпис ментора



KEY WORDS DOCUMENTATION

Accession number, ANO :	
Identification number, INO :	
Document type, DT :	Monographic publication
Type of record, TR :	Textual printed material
Contents code, CC :	Bachelor Thesis
Author, AU :	Dejan Subašić
Mentor, MN :	Nemanja Nedić, PhD
Title, TI :	Incremental importer za CIM model
Language of text, LT :	Serbian
Language of abstract, LA :	Serbian
Country of publication, CP :	Republic of Serbia
Locality of publication, LP :	Vojvodina
Publication year, PY :	2021
Publisher, PB :	Author's reprint
Publication place, PP :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, PD : (chapters/pages/ref./tables/pictures/graphs/appendixes)	6/24/15/0/11/0/0
Scientific field, SF :	Electrical and computer engineering
Scientific discipline, SD :	Applied computer science and informatics
Subject/Key words, S/KW :	Smart Grid, distributed management systems, Common Information Model
UC	
Holding data, HD :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, N :	
Abstract, AB :	This paper presents an implementation of a system for management of the elements of a Smart Grid. Smart systems have been evolving in the last few decades and their intention is to use computing and information technology to improve the efficiency and range of possible services. The details of the implemented project and an example of a scenario for each service are presented in detail.
Accepted by the Scientific Board on, ASB :	
Defended on, DE :	
Defended Board, DB :	President: Darko Čapko, PhD
	Member: Srđan Vukmirović, PhD
	Member, Mentor: Nemanja Nedić, PhD
	Mentor's sign

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

(Податке уноси предметни наставник - ментор)

Студијски програм:	
Руководилац стидијског програма:	

Студент:		Број индекса:	
Област:			
Ментор:			

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

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

НАСЛОВ ДИПЛОМСКОГ РАДА:

--

ТЕКСТ ЗАДАТКА:

--

Руководилац студијског програма:	Ментор рада:

Примерак за: ☐ - Студента; ☐ - Ментора

SADRŽAJ

1. Uvod.....	1
2. Opis problema.....	3
3. Korišćene tehnologije	4
3.1 Programski jezik C#	4
3.2 .NET	5
3.3 Windows Presentation Foundation.....	6
3.4 Structured Query Language	6
3.5 Entity Framework.....	7
3.6 Common Information Model	7
3.7 RDFS i CIM/XML	8
4. Inkrementalni importer	10
4.1 CIM profil	11
4.1.1 Profil Kreator	14
4.2 Importer.....	15
4.2.1 Opis resursa	16
4.2.2 Defaulter	17
4.2.3 Delta.....	18
4.3 Servis.....	18
4.4 Klijentska aplikacija.....	19
5. Zaključak	23
6. Literatura.....	24

SPISAK SLIKA

Slika 3-1 Primer klase <i>ACLineSegmentPhase</i> iz CIM/XML zapisa	9
Slika 4-1 Dijagram komponenti	10
Slika 4-2 Prikaz prozora <i>Project Model</i> aplikacije <i>CIMTool</i>	11
Slika 4-3 UML prikaz definisanog profila za ovaj zadatak	12
Slika 4-4 Izgled CIM Profile Creator aplikacije	14
Slika 4-5 Uspešno importovana inkrementalna datoteka.....	15
Slika 4-6 Redosled importovanja konkretnih klasa	16
Slika 4-7 Isečak koda metode <i>PopulateMutualCouplingProperties</i>	17
Slika 4-8 Values prozor klijentske aplikacije	19
Slika 4-9 Extent Values prozor klijentske aplikacije	20
Slika 4-10 Related Values prozor klijentske aplikacije	21

SKRAĆENICE

ID	- <i>Identification</i> , šifra koja jednoznačno identifikuje neki resurs
CIM	- <i>Common Information Model</i> , Zajednički informacioni model
GUI	- <i>Graphical User Interface</i> , Grafički korisnički interfejs
SQL	- <i>Structured Query Language</i> , Struktuirani jezik upita
RDF	- <i>Resource Description Framework</i> , Frejmwork za opis resursa
DLL	- <i>Dynamic link library</i> , Biblioteka dinamičkih veza
GDA	- <i>Generic Data Access</i> , Sprega za pristup generičkim podacima
IEC	- <i>International Electrotechnical Commission</i> , internacionalna elektrotehnička komisija

1. Uvod

Kako se ljudsko društvo modernizuje tako sve više zavisi od električne energije. Energetsku industriju treba promeniti kako bi se pozabavila pitanjima modernog digitalnog društva. Potrošači zahtevaju visok kvalitet električne energije i kritičniji asortiman dodatnih usluga. Iz toga proizlazi potreba da se svi sistemi elektrodistribucije učine inteligentnijim i efikasnijim, a to su: generisanje energije, njen prenos i distribucija, kao i neposredna potrošnja. Štaviše, potrošačima je potrebno da troškovi budu niži.

Razvijanjem računarstva i informacionih tehnologija nastaju takozvani pametni mrežni sistemi (*Smart Grid*). Pametna mreža je električna mreža koja koristi digitalne i druge napredne tehnologije za praćenje i upravljanje transportom električne energije iz svih izvora proizvodnje kako bi se zadovoljile različite potrebe krajnjih korisnika za električnom energijom. Pametne mreže omogućavaju da se sredstva obnovljivih izvora energije bezbedno povežu sa sistemom. Pametne mreže koordiniraju potrebe i mogućnosti svih proizvođača, mrežnih operatera, krajnjih korisnika na tržištu električne energije da rade što je moguće efikasnije, minimizirajući troškove i uticaje na životnu sredinu dok maksimiziraju pouzdanost, otpornost i stabilnost sistema. Digitalna revolucija u ekonomiji takođe utiče na sektor električne energije. Sve više vlada, komunalnih preduzeća i proizvođača prihvata digitalne tehnologije, od primene hardverskih sredstava kao što su pametna brojila, digitalne podstanice i pametna infrastruktura za punjenje električnih vozila do korišćenja softverskih rešenja kao što su veštačka inteligencija i blokčejn tehnologija. Kako se razvijaju nove usluge i tehnološke platforme, povećava se potreba za uređajima da komuniciraju i rade neprimetno na svim nivoima mreže [14].

Integracija obnovljivih izvora energije u elektroenergetski sistem zahteva njihovu delikatnu ravnotežu. Ovo se može rešiti tako što električne mreže rade pametno i ekonomično. Da bi se to postiglo, neophodna je besprekorna i efikasna razmena informacija u različitim

fazama, između sve većeg broja kompanija. *Common Information Model* (CIM)[1] je standard koji se koristi u elektrodistribuciji za definisanje elemenata sistema u *Smart Grid*-u. Potreba za zajedničkim standardom proizilazi iz ogromne kompleksnosti ovakvih sistema i zbog toga što se često sistemi koji su se originalno odvojeno razvijali naknadno povezuju u veći sistem. CIM standardi se neprestano razvijaju kako bi ispunili promenljive zahteve za razmenu podataka, koji se povećavaju i po učestalosti i po vrsti.

Razmena modela mreže je složen proces koji pokriva različite slučajeve upotrebe, koji uključuju razmenu:

- Informacije o opremi, koje sadrže podatke o opremi elektroenergetskog sistema;
- Informacije o topologiji, koje sadrže informacije vezane za topologiju za elemente mreže;
- Informacije o promenjivama stanja elektroenergetskog sistema, koje sadrže rezultate inicijalne simulacije toka opterećenja sistema;
- Informacije o hipotezi stacionarnog stanja, koje su važeće za novije standarde i pružaju informacije o vrednostima opterećenja i proizvodnje, kao i drugim ulaznim parametrima neophodnim za izvođenje simulacija toka opterećenja.[15]

2. Opis problema

Cilj rada je razvijanje sistema pametne električne mreže, koristeći .NET platformu i programski jezik C#. Sistem se sačinjava od kontrolnog servisa, kreatora profila, importera i klijentske aplikacije. Servis je terminalska aplikacija a ostale komponente nude grafički interfejs razvijen pomoću framework-a WPF. Upotrebom aplikacije „Profil Kreator“ definišu se objekti od kojih se model sačinjava i njihovi međusobni odnosi. Servis trajno skladišti stanje modela mreže i loguje sve akcije nad sistemom. Nudi spregu za definisanje upita ka stanju modela mreže. Postoje tri vrste upita: pretraga određenog resursa preko njegovog ID-a (*Get Values*), pretraga svih resursa određenog tipa (*Get Extent Values*) i pretraga svih resursa koji su spregnuti sa određenim resursom (*Get Related Values*). Zadavanje i prikaz rezultata upita moguće je izvršiti direktno nad terminalom serverske aplikacije ili grafički preko interfejsa klijentske aplikacije.

Akcent rada je na komponenti importer kojoj je zaduženje osvežavanje servisa. Detektuje promene u realnom sistemu i generiše upit ka servisu sa porukom o izmeni. Taj upit nadalje se naziva inkrementalna datoteka ili samo delta. Importer konvertuje klase definisane CIM standardom u generičke resurse sa kojima radi servis. Delta se definiše tako da se samo izmene propagiraju a ne celokupno stanje realnog sistema. To se postiže pamćenjem prošlog stanja u SQL bazi podataka. Pristup podacima realizovan je pomoću framework-a *EntityFramework*. Delta se popunjava razlikom između trenutnog stanja realnog sistema i snimkom iz baze. Delta razvrstava resurse po tome koje resurse treba dodati, osvežiti ili obrisati.

U narednom delu rada prvo će biti objašnjene teorijske osnove za korišćene tehnologije a zatim detaljan pregled funkcionalnosti sistema po komponentama.

3. Korišćene tehnologije

3.1 Programski jezik C#

C# je visoki jezik opšte namene koji podržava više programerskih paradigmi [2]. Razvijan je od strane Microsoft-a 2000. god. i od trenutka pisanja ovog rada najnovija verzija je 9.0 izbačena 10. Novembra 2020. C# je prvenstveno objektno-orijentisan jezik.

Još neke glavne karakteristike su :

- garbage collection,
- nullabilni tipovi,
- hvatanje izuzetaka
- lambda izrazi koristeći LINQ sintakse.

Kompajler prevodi C# jezik u *Intermediate Language* [3] koji se izvršava na virtuelnoj .NET mašini.

Navode se sledeći ciljevi dizajna jezika C# [4] :

- C# jezik je namenjen da bude jednostavan, moderan, objektno orijentisan programski jezik opšte namene.
- Jezik i njegove implementacije treba da obezbede podršku za principe softverskog inženjeringa kao što su jaka provera tipa, provera granica niza, otkrivanje pokušaja korišćenja neinicijalizovanih promenljivih i automatsko sakupljanje smeća. Robusnost softvera, izdržljivost i produktivnost programera su važni.
- Jezik je namenjen za upotrebu u razvoju softverskih komponenti pogodnih za primenu u distribuiranim okruženjima.

- Prenosivost izvornog koda je veoma važna, kao i prenosivost programera, posebno za one programere koji su već upoznati sa C i C++.
- Podrška internacionalizaciji je veoma važna.
- C# je namenjen da bude pogodan za pisanje aplikacija i za hostovane i za ugrađene sisteme, u rasponu od veoma velikih koji koriste sofisticirane operativne sisteme, do veoma malih sa namenskim funkcijama.
- Iako su C# aplikacije namenjene da budu ekonomične u pogledu zahteva za memorijom i procesorskom snagom, jezik nije imao nameru da se direktno takmiči u performansama i veličini sa C ili asemblerskim jezikom.

C# naglašava verzionisanje kako bi se osiguralo da programi i biblioteke mogu da se razvijaju tokom vremena na kompatibilan način. Aspekti dizajna C#-a na koje su direktno uticala razmatranja o verzionisanju uključuju odvojene *virtual* i *override* modifikatore, pravila za rešavanje preopterećenja metoda i podršku za eksplicitne deklaracije članova interfejsa.

3.2 .NET

C# programi se pokreću na .NET-u [2], virtuelnom sistemu za izvršavanje koji se zove *Common Language Runtime* (CLR) i skupu biblioteka klasa. CLR je Microsoft implementacija zajedničke jezičke infrastrukture (CLI), međunarodnog standarda. CLI je osnova za kreiranje okruženja za izvršavanje i razvoj u kojima jezici i biblioteke neometano rade zajedno.

Izvorni kod napisan u C# se kompajlira u srednji jezik (IL) koji je u skladu sa CLI specifikacijom. Kada se C# program izvrši, sklop se učitava u CLR. CLR vrši *Just-In-Time* (JIT) kompilaciju da konvertuje IL kod u izvorne mašinske instrukcije. CLR pruža druge usluge koje se odnose na automatsko sakupljanje smeća, rukovanje izuzecima i upravljanje resursima. Kod koji izvršava CLR se ponekad naziva „upravljanim kodom“ (*managed code*). „Neupravljeni kod“ (*unmanaged code*) je kompajliran u mašinski jezik koji cilja na određenu platformu.

Interoperabilnost jezika je ključna karakteristika .NET-a. IL kod koji proizvodi C# kompajler je u skladu sa *Common Type Specification* (CTS). IL kod generisan iz C# može da komunicira sa kodom koji je generisan iz .NET verzija jezika F#, Visual Basic i C++.

Postoji više od 20 drugih jezika kompatibilnih sa CTS. Jedan sklop može da sadrži više modula napisanih na različitim .NET jezicima. Tipovi se mogu pozivati jedni na druge kao da su napisani na istom jeziku.

Pored usluga vremena izvršavanja, .NET takođe uključuje opširne biblioteke. One su organizovane u prostore imena (*namespace*) koji pružaju širok spektar korisnih funkcija. Biblioteke obuhvataju sve, od unosa i izlaza datoteke do manipulacije stringovi ma do XML parsovanja, preko okvira veb aplikacija do kontrola *Windows Forms*-a. Tipična C# aplikacija u velikoj meri koristi .NET biblioteku klasa za rukovanje uobičajenim poslovima.

3.3 Windows Presentation Foundation

Windows Presentation Foundation (WPF) je open-source framework, razvijen od strane Microsoft-a i deo .Net Framework-a [5], koji je izabran jer je pogodan za realizaciju jednostavnih desktop aplikacija. WPF koristi *Extensible Application Markup Language* (XAML) za definisanje UI komponenti.

WPF aplikacije se mogu primeniti kao samostalni desktop programi ili hostovani kao ugrađeni objekat na web lokaciji. WPF ima za cilj da objedini niz zajedničkih elemenata korisničkog interfejsa, kao što su 2D/3D renderovanje, fiksni i adaptivni dokumenti, tipografija, vektorska grafika, animacija tokom izvršavanja i unapred renderovani mediji. Ovi elementi se zatim mogu povezati i manipulirati na osnovu različitih događaja, interakcija korisnika i povezivanja podataka (*data binding*). WPF biblioteke su uključene u sve verzije Microsoft Windows-a od Windows Vista i Windows Server 2008.

WPF izlaže sistem svojstava za objekte koji nasleđuju *DependencyObject*, koji je svestan zavisnosti između potrošača svojstva i može pokrenuti akcije zasnovane na promenama svojstava. Svojstva mogu biti ili zakucane kodirane vrednosti ili specifični izrazi koji daju rezultat. Vrednost svojstava se takođe može naslediti od nadređenih objekata. Svojstva WPF-a podržavaju obaveštenja o promeni, koja pozivaju spregnute funkcije kao reakciju na promenu nekog svojstva. Prilagođena ponašanja se mogu koristiti za širenje obaveštenja o promeni svojstva preko skupa WPF objekata. Skoro sve, od podešavanja boja i pozicija do animiranih elemenata, može postići postavljanjem vrednosti svojstava [6].

3.4 Structured Query Language

SQL (*Structured Query Language*, strukturirani jezik upita) je standardizovani jezik za upravljanje relacionim bazama podataka [7]. SQL je veoma visok jezik u smislu da korisnik veoma opšto definiše šta da se uradi a ne kako da se to uradi.

SQL se koristi od strane ne samo administratora baza podataka, već i programera koji pišu skripte za integraciju podataka i analitičara podataka koji žele da postave i pokrenu analitičke upite.

Upotreba SQL-a uključuje modifikovanje struktura tabele i indeksa baze podataka; dodavanje, ažuriranje i brisanje redova podataka; i preuzimanje podskupova informacija iz baze podataka za obradu transakcija i analitičke aplikacije. Upiti i druge SQL operacije imaju oblik komandi napisanih kao naredbe – najčešće korišćeni SQL izrazi uključuju *select*, *add*, *insert*, *update*, *delete*, *create*, *alter* i *truncate*.

Naredbe se dele na:

- DML (jezik za manipulisanje podacima)
- DLL (jezik za definisanje naredbi)
- transakcione kontrole
- bezbednosne kontrole

SQL je postao standardni programski jezik za relacione baze podataka nakon što su se pojavile kasnih 1970-ih i ranih 1980-ih. Takođe poznati kao SQL baze podataka, relacioni sistemi se sastoje od skupa tabela koje sadrže podatke u redovima i kolonama. Svaka kolona u tabeli odgovara kategoriji podataka – na primer, imenu ili adresi klijenta – dok svaki red sadrži vrednost podataka za kolonu koja se ukršta.

3.5 Entity Framework

Entity Framework (frejmwork entiteta) je sprega otvorenog koda za .NET aplikacije koja mapira objekte i relacije. Omogućava programerima da rade sa podacima koristeći objekte klase specifičnih za domen bez fokusiranja na osnovne tabele i kolone baze podataka u kojima se ovi podaci čuvaju. Programeri mogu da rade na višem nivou apstrakcije kada rade sa podacima i mogu da kreiraju i održavaju aplikacije orijentisane na podatke sa manje koda u poređenju sa tradicionalnim aplikacijama.

Prva verzija izašla je 2008. godine a od trenutka pisanja ovog rada najnovija stabilna verzija je *Entity Framework* v6.4.4, odnosno *Entity Framework Core*: v5.0.0.

3.6 Common Information Model

CIM je standard koji definiše kako su elementi u IT okruženju predstavljeni kao zajednički skup objekata i odnosa između njih [1]. CIM je UML model. Prihvaćen od strane IEC-a (*International Electrotechnical Commission*, internacionalna elektrotehnička komisija). Definisan pod standardima IEC 61970-301. CIM standard se sastoji od specifikacije, šeme i metamodela. Šema određuje konkretne modele i konfiguracije. Specifikacija definiše kako se CIM model integriše sa drugim modelima. Metamodel definiše semantiku za proširenje CIM šeme.

IEC61968 [8] je serija standarda u razvoju koji će definisati standarde za razmenu informacija između električnih distributivnih sistema. Ove standarde razvija Radna grupa 14 Tehničkog komiteta IEC-a. IEC61968 je namenjen da podrži međuplikacionu integraciju preduzeća koje treba da prikuplja podatke iz različitih aplikacija koje su inicijarno dizajnirane sa različitim interfejsima i okruženjima u toku rada. IEC61968 definiše interfejse za sve glavne elemente arhitekture interfejsa za DMS i namenjen je za implementaciju *middleware* servisa koji prenose poruke među aplikacijama.

Serija standarda IEC61970 [9] bavi se interfejsima aplikacijskih programa za sisteme upravljanja energijom (*Energy Managment Systems*, EMS). Standardi pružaju skup smernica koji olakšavaju integraciju aplikacija koje su razvili različiti dobavljači u okruženju kontrolnog centra, razmenu informacija sa sistemima koji su eksterni u odnosu na okruženje kontrolnog centra, uključujući sisteme za prenos, distribuciju i proizvodnju van kontrolnog centra koji treba da razmenjuju podatke u realnom vremenu sa kontrolnim centrom i obezbeđivanje odgovarajućih interfejsa za razmenu podataka kroz stare i nove sisteme.

IEC62325 [10] je skup standarda koji se odnose na deregulisanu komunikaciju tržišta energije. Definiše klase namenjene sklapanju poslovnih i pravnih dogovora.

3.7 RDFS i CIM/XML

Resource Description Framework (RDF) [11] je standardizovani model za razmenu podataka na web-u. Formalna sintaksa je bazirana na XML-u, te je ljudski čitljiv. Podatak se sastoji od trojke: subjekta, objekta i predikata. *RDF Schema* (RDFS) [12] je semantičko proširenje RDF-a i služi za modelovanje konkretnog sistema. RDFS datoteka opisuje klase, attribute i njihove veze. Pojam klase je sličan kao kod objektno-orijentisanih programskih jezika. CIM/XML je RDF datoteka koja služi za razmenu podataka između klijenta i servisa.

ACLineSegmentPhase

Represents a single wire of an alternating current line segment.

Native Members

phase	1..1	SinglePhaseKind	The phase connection of the wire at both ends.
ACLineSegment	0..1	ACLineSegment	The line segment to which the phase belongs.

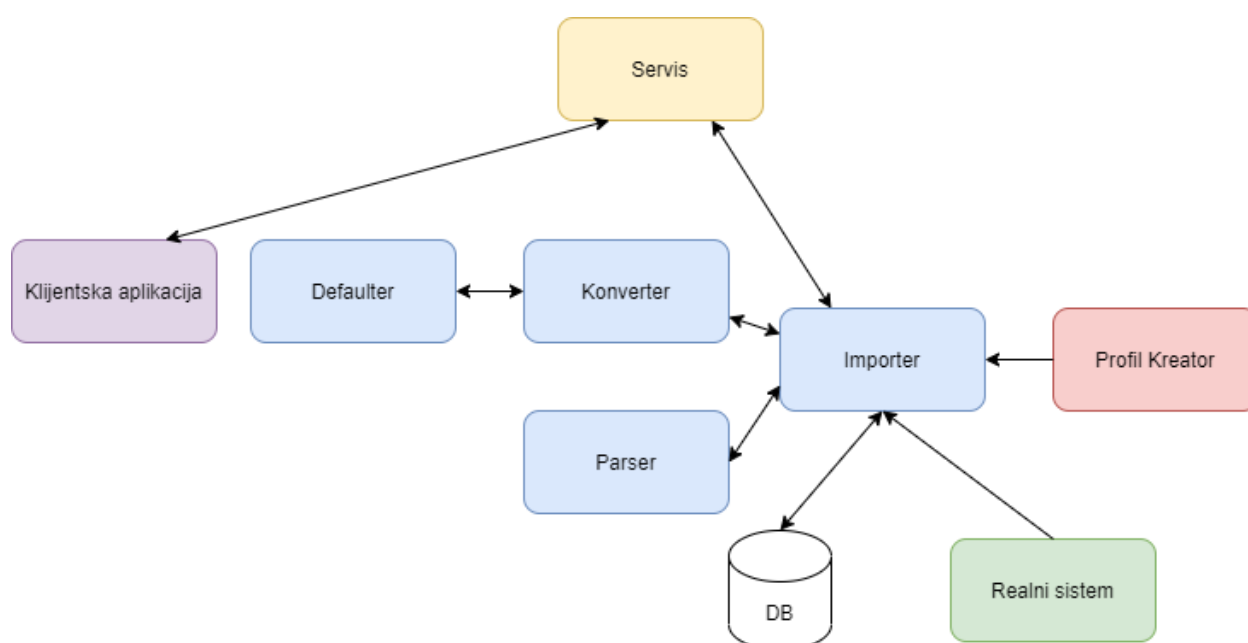
Inherited Members

mRID	1..1	string	see IdentifiedObject
aliasName	1..1	string	see IdentifiedObject
name	1..1	string	see IdentifiedObject

Slika 3-1 Primer klase *ACLineSegmentPhase* iz CIM/XML zapisa

Na slici 3-1 se vidi prikaz jedne klase iz sistema. Klasi *ACLineSegmentPhase* su nativna polja *phase* i *ACLineSegment*, dok su polja *mRID*, *aliasName* i *name* nasleđena od roditeljske klase *IdentifiedObject*. Pored svakog polja izlistan je njen domen odnosno tip, kao i komentar.

4. Inkrementalni importer



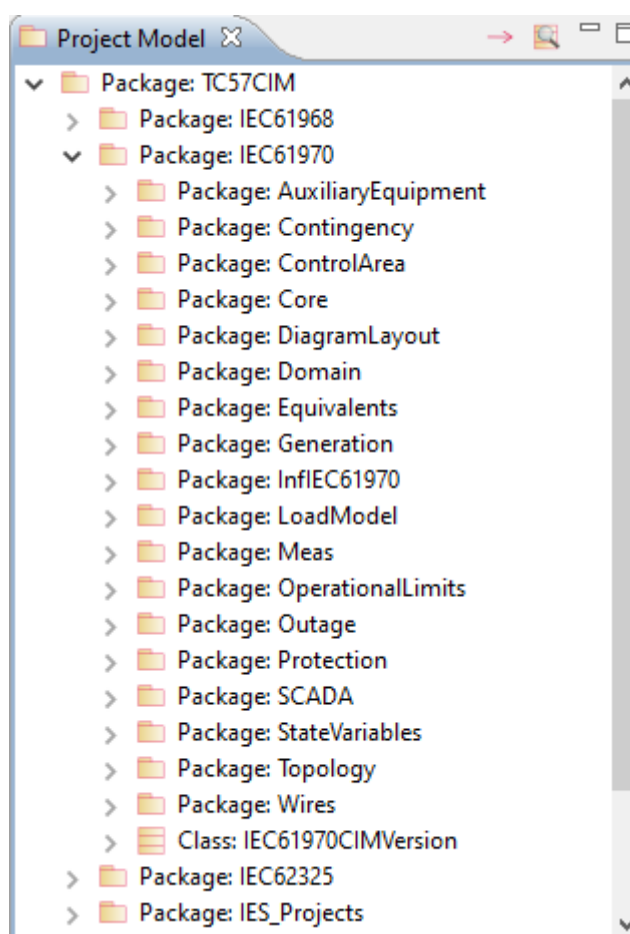
Slika 4-1 Dijagram komponenti

Slika 4-1 prikazuje dijagram komponenti servisa. Profil kreator, u odnosu na CIM profil, generiše biblioteku CIM klasa za upotrebu od strane importera. Importer koristi svoje podkomponente za generisanje inkrementalne delta datoteke, vodeći računa o stanju trenutnog i prethodnog stanja realnog sistema, prepisujući CIM objekte u generičke resurse. Koristeći *EntityFramework* komunicira sa bazom podataka u svrhu pamćenja prethodnog stanja. Komponenta parser se sačinjava od pomoćnih metoda za čitanje i zapisivanje podataka iz XML formata. Klijentska aplikacija služi za grafički prikaz stanja mreže, implementirajući serversku GDA spregu. U narednom delu rada sledi detaljan opis komponenti.

4.1 CIM profil

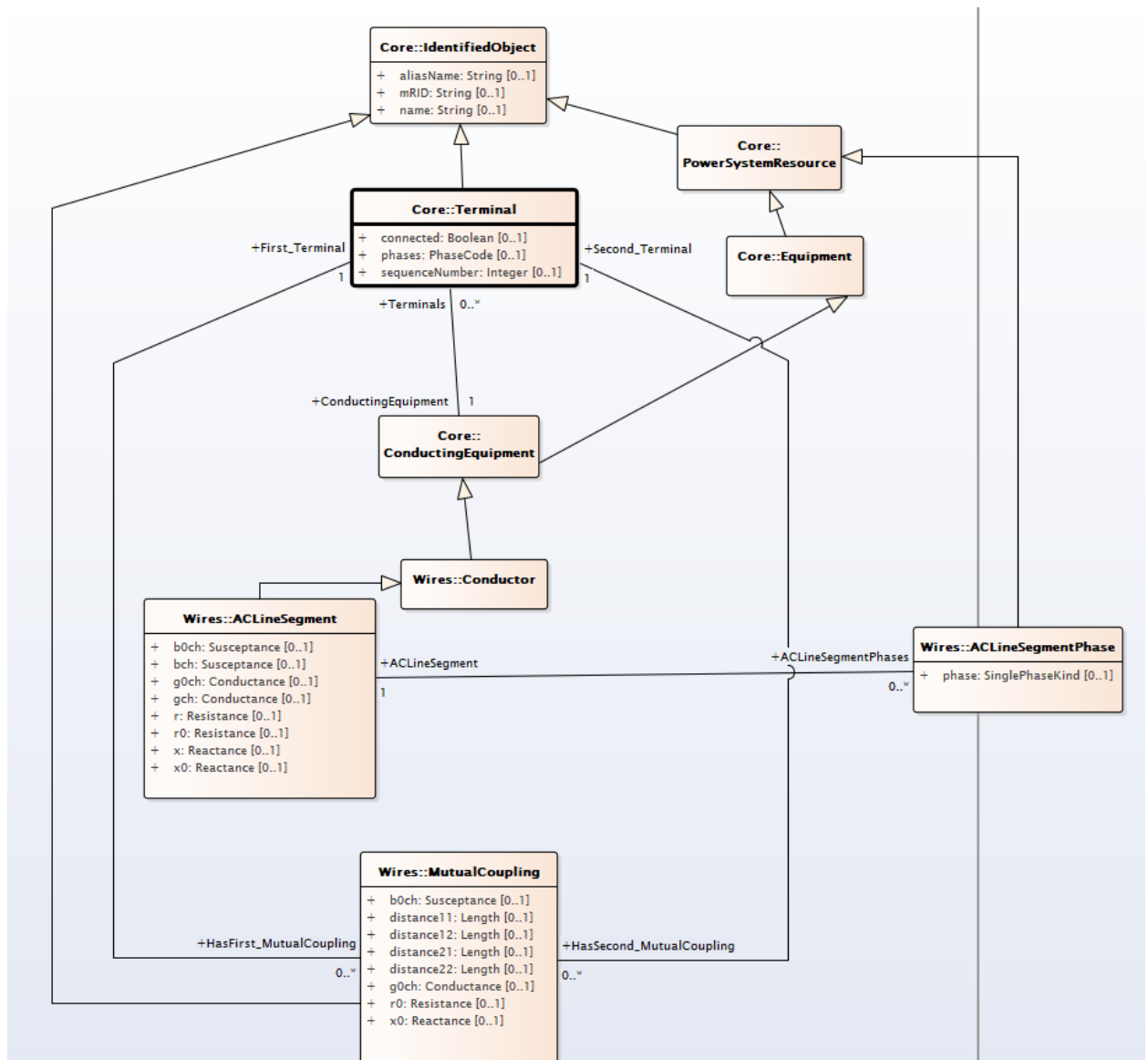
Pošto je celokupan CIM model preobilan i nepotreban za većinu primena, kreiraju se takozvani profili koji se sačinjavaju od tih klasa i odnosa koji su modelaru od interesa. Jedan od alata za kreiranje CIM profila je CIMTool, alatka za Eclipse IDE.

CIMTool [13] omogućava kreiranje profila većeg, globalnog informacionog modela. Globalni model je CIM model sa proširenjima. Profil je mali model koji se koristi za određenu primenu koji je u skladu sa većim modelom. *CIMTool* nudi funkcije čitanja i uvezivanja CIM i lokalnih UML modela u XML formatu, pregledavanje modela i provera nekonzistentnosti, generisanje šema i ontologija.



Slika 4-2 Prikaz prozora *Project Model* aplikacije *CIMTool*

Na slici 4-2 se vidi istraživač klasa modela CIM standarda. Klase su organizovane po standardima IEC61968, IEC61970 i IEC62325. Detaljnije o ovim standardima se govori u sekciji 3.6.



Slika 4-3 UML prikaz definisanog profila za ovaj zadatak

Kompleksnost profila ne igra ulogu u prikazu funkcionalnosti zadatka ovog rada te je definisan jednostavan profil. Profil se sastoji od ukupno devet klasa i to pet konkretnih i četiri apstraktne.

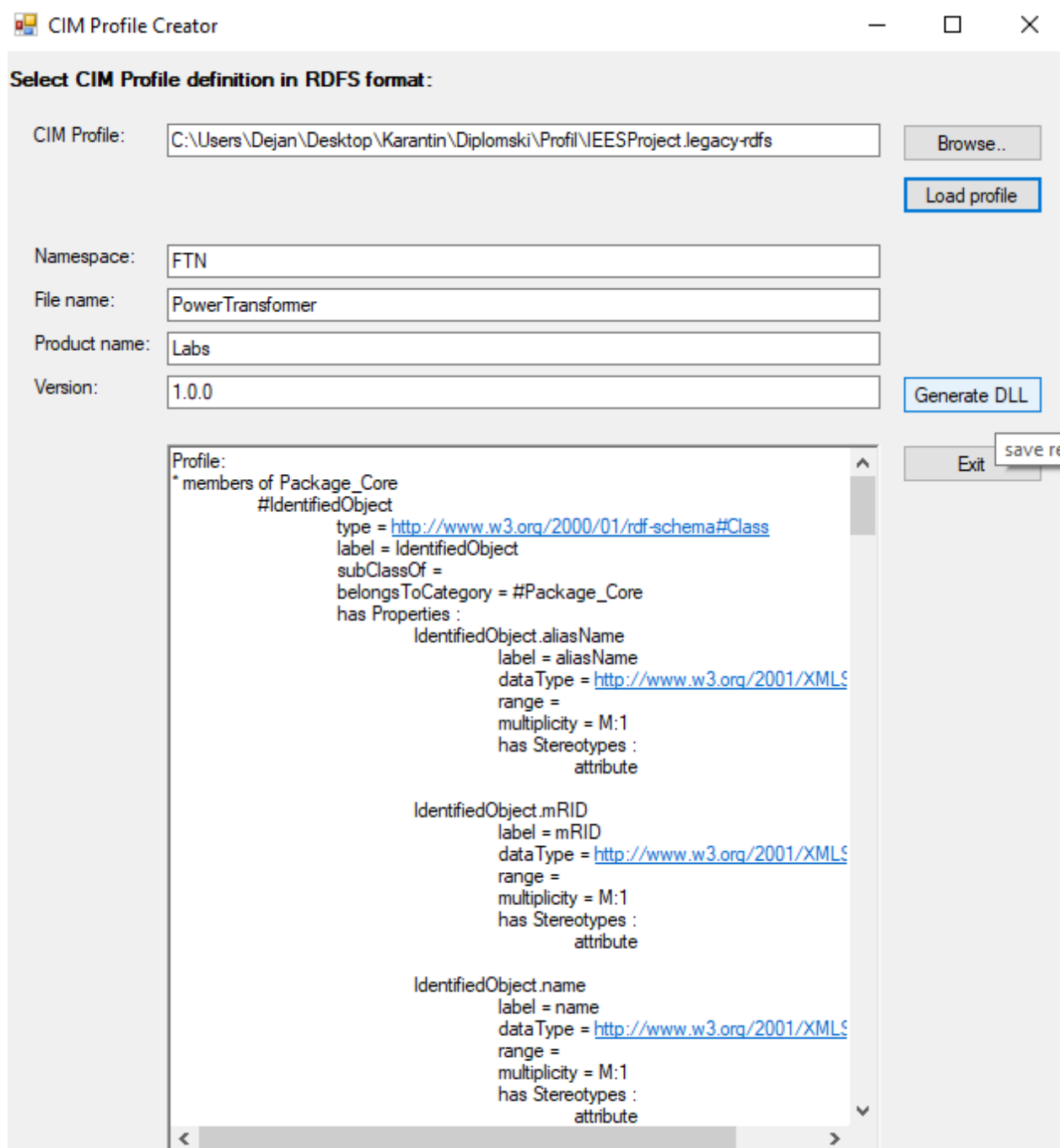
Klase i njihova svojstva su:

- *IdentifiedObject*, čvorna klasa koju nasleđuju sve ostale
 - *mRID*, identifikator resursa, string vrednost
 - *name*, ljudski čitljivo ime resursa, string vrednost
 - *aliasName*, alternativno ime, ako postoji, string vrednost
- *Terminal*, predstavlja kontakt među dva elektronska uređaja, konkretna klasa
 - *connected*, stanje terminala, boolski tip
 - *phases*, šifra faze, enumeracija
 - *sequenceNumber*, broj sekvence, celobrojna vrednost

- *ConductingEquipment*, provodni vod terminala, više terminala mogu biti povezani na jedan *ConductingEquipment*, referentni tip
- *PowerSystemResource*, može biti deo opreme kao što je prekidač, kontejner opreme koji sadrži mnogo pojedinačnih delova opreme kao što je podstanica, ili organizacioni entitet kao što je pod-kontrolna oblast
- *ACLineSegmentPhase*, predstavlja fazu jedne žice segmenta naizmenične struje, konkretna klasa
 - *Phase*, šifra faze, enumeracija
 - *ACLineSegment*, objekat kome ova faza pripada, referentni tip
- *Equipment*, modeluje fizičke uređaje elektroenergetskog sistema, bilo mehaničke ili elektronske
- *ConductingEquipment*, nosioci električne energije
- *Conductor*, provodni materijal sa konstantnim električnim karakteristikama koji se koristi za prenos struje između tačaka u elektroenergetskom sistemu
- *ACLineSegment*, žica ili kombinacija žica koje grade jedan električni sistem, koji se koristi za prenos naizmenične energije između tačaka u sistemu, konkretna klasa
 - *bch*, *b0ch*, električna susceptansa
 - *gch*, *g0ch*, električna provodljivost
 - *r*, *r0*, električna otpornost
 - *x*, *x0*, električna reaktansa
- *MutualCoupling*, modeluje međusobnu spregu vodova, konkretna klasa
 - *b0ch*, električna susceptansa,
 - *g0ch*, električna provodljivost
 - *r0*, električna otpornost
 - *x0*, električna reaktansa
 - *distance11*, razdaljina od terminala prvog voda do početka sprege
 - *distance12*, razdaljina od terminala prvog voda do kraja sprege
 - *distance21*, razdaljina od terminala drugog voda do početka sprege
 - *distance22*, razdaljina od terminala drugog voda do kraja sprege
 - *First_Terminal*, prvi terminal sprege, referentni tip
 - *Second_Terminal*, drugi terminal sprege, referentni tip

4.1.1 Profil Kreator

Cilj aplikacije je da se učitava i parsira CIM profil, te kreira dinamička biblioteka klasa za modelovanje CIM definisanih klasa. Ulazna datoteka za aplikaciju je izlaz *CIMTool* alatke, odnosno RDFS datoteka sa podacima o profilu.

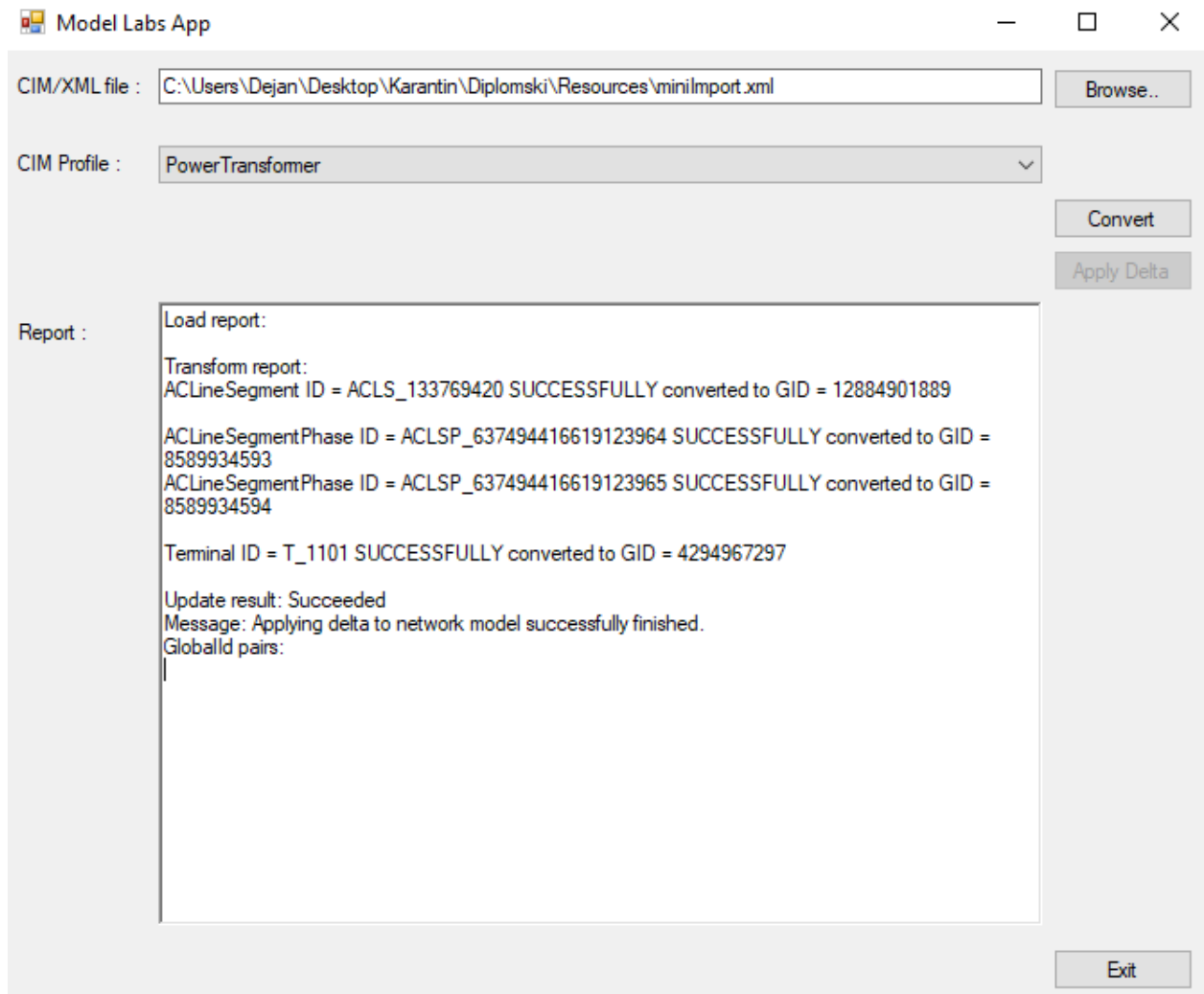


Slika 4-4 Izgled CIM Profile Creator aplikacije

Klikom na „Browse“ taster iz istraživačkog menija odabira profil u RDFS formatu. „Load profile“ taster parsira profil i prikazuje strukturu modela u tekstualnom obliku na ekranu.

Izlistani su resursi, njihovi atributi i veze između njih. „Generate DLL“ taster zapisuje model u obliku dinamičke biblioteke sa modelima klasa po CIM standardu. Biblioteka se dalje koristi od strane importera za zapis resursa.

4.2 Importer



Slika 4-5 Uspešno importovana inkrementalna datoteka

Cilj importera je da generiše inkrementalnu poruku (dalje: delta) kako bi se stanje mreže na servisu osvežilo i odgovaralo stanju realnog sistema. Delta se popunjava generičnim resursima (*ResourceDescription*) koji se dobijaju konverzijom iz CIM klasa za opis komponenti mreže. Detaljnije u sekciji 4.2.1.

Delta propagira samo promene na sitemu a ne celu kopiju sistema. Da bi se delta korektno popunila potrebno je uporediti novo stanje realnog sistema sa zapamćenom slikom prošlog stanja. Stanje mreže se snima u SQL bazu podataka uz pomoć framework-a za pristup podacima

EntityFramework. Resursi se razvrstavaju po tome koje resurse treba dodati, osvežiti ili obrisati. Ako se servis uspešno osveži i vrati potvrdnu poruku importer snima novo stanje mreže u bazu podataka.

Importer pruža grafički korisnički interfejs (slika 4-5). Preko interfejsa aplikacije moguće je ručno izabrati CIM/XML datoteku sa zapisom modela mreže, parsirati taj model, definisati deltu i proslediti je servisu. U ovoj implementaciji moguće je izabrati datoteku koji predstavlja stanje mreže klikom na „Browse“ (slika 4-5). Izveštaj operacije se zapisuje u log datoteku i prikazuje korisniku na interfejsu aplikacije. Klikom na „Convert“ parsira se stanje sistema koristeći komponentu Parser (videti dijagram na slici 4-1) koja pruža pomoćne funkcije za obradu fajlova XML formata. Detalji implementacije parsera nisu ključni radu importera te su izostavljeni iz rada. Rezultat je interni model zapisan preko CIM klasa.

Importer prepoznaje one klase i odnose koji su definisani CIM profilom. Komponenta adapter prepisuje objekte iz CIM klasa u generički opis resursa (*ResourceDescription*). Upakuje resurse u inkrementalnu delta datoteku i šalje servisu. Detaljnije u sekciji 4.2.3.

4.2.1 Opis resursa

Opis resursa (*ResourceDescription*) je generički model za predstavu elemenata sistema. Sačinjava se od ID-a resursa i liste njegovih atributa. Može da sadrži sve ili samo odabrane attribute. Atributi se opisuju svojom šifrom i vrednošću. Šifra atributa nosi sa sobom informaciju o domenu vrednosti.

Komponenta *Converter* vrši konvertovanje konkretnih klasa iz CIM klasa tako što kreira instancu resursa, dodeli mu ID i redom popunjava attribute iz originalne CIM klase. Od ključne je važnosti redosled obrada tipova klasa, neophodno je da se one klase koje su referencirane od strane drugih obrade prve.

```
ImportACLineSegment(fileName);
ImportACLineSegmentPhase(fileName);
ImportTerminal(fileName);
ImportMutualCoupling(fileName);
```

Slika 4-6 Redosled importovanja konkretnih klasa

Na slici 4-6 prikazuje se redosled importovanja u implementaciji ovog servisa. Na primer, ako bi se zamenio redosled pozivanja metoda *ImportTerminal(...)* i *ImportMutualCoupling(...)* došlo bi do izuzetka jer bi neki objekat klase *MutualCoupling* preko svog polja *First_Terminal*

(jasno je da slučaj važi i za *Second_Terminal*) pokušao da referencira resurs nekog od terminala a još uvek nije obrađen ni jedan od njih.

Posmatrana je jedna od metoda za importovanje, *ImportMutualCoupling(...)*. Nakon uspešnog parsiranja stanja sistema (sekcija 4.2) stvara se rečnik svih klasa modela. Od tog rečnika uzeće se one klase koje su tipa *MutualCoupling*. Iterativno za svaki od tih resursa stvara se objekat klase *ResourceDescription* metodom *CreateMutualCouplingResourceDescription(...)* i svojstva resursa se popunjavaju metodom *PopulateMutualCouplingProperties(...)*. Na slici 4-7 se prikazuje isečak te metode. Rezultat operacije se loguje u datoteku izveštaja.

4.2.2 Defaulter

Omogućeno je automatsko popunjavanje podrazumevanih vrednosti komponentom Defaulter. Za svaki nenaveden a obavezan atribut se uzima njemu odgovarajuća podrazumevana vrednost iz predefinisane CIM/XML datoteke.

```
public static void PopulateMutualCouplingProperties(MutualCoupling cim, ResourceDescription rd,
                                                    ImportHelper importHelper, TransformAndLoadReport report)
{
    if ((cim != null) && (rd != null))
    {
        PopulateIdentifiedObjectProperties(cim, rd, importHelper, report);

        if (cim.B0chHasValue)
        {
            rd.AddProperty(new Property(ModelCode.MUTUALCOUPLING_B0CH, cim.B0ch));
        }
        else if (MutualCoupling.IsB0chMandatory)
        {
            DefaultProperty(rd, ModelCode.MUTUALCOUPLING_B0CH);
        }

        if (cim.G0chHasValue)
        {
            rd.AddProperty(new Property(ModelCode.MUTUALCOUPLING_G0CH, cim.G0ch));
        }
        else if (MutualCoupling.IsG0chMandatory)
        {
            DefaultProperty(rd, ModelCode.MUTUALCOUPLING_G0CH);
        }
    }
}
```

Slika 4-7 Isečak koda metode *PopulateMutualCouplingProperties*

Na slici 4-7 se prikazuje isečak funkcije koja popunjava objekat resursa *ResourceDescription* svojstvima *B0ch* i *G0ch* klase *MutualCoupling* definisane u CIM modelu. Pri importovanju stanja sistema za svako prepoznato svojstvo se podesi *HasValue* bufska vrednost. Za pozivne vrednosti uzima se učitana vrednost a u slučaju da nje nema proverava se da li je zadata podrazumevana vrednost u komponenti Defaulter metodom *DefaultProperty(...)*.

4.2.3 Delta

Zadatak delta objekta je da nosi informaciju o tome koje promene treba da se izvrše na servisu kako bi mu stanje odgovaralo stanju realnog sistema.

Delta objekat se sačinjava od tri vrste operacija:

- insert
- update
- delete

Operacije se modeluju listama resursa. Delta objekat se popunjava sledećim algoritmom: Za svaki objekat proveriti da li se ID datog objekta nalazi zapamćen u bazi. Ako se ne nalazi, objekat je novi i kao takav treba se dodati u insert operaciju delte. Ako se nalazi, objekat već postoji na servisu i njegove izmenjene attribute treba dodati u update operaciju. Ako postoje ID-jevi u bazi za objekte kojih nema u trenutnom stanju sistema, takve treba označiti za brisanje delete operacijom.

„Apply Delta“ tasterom (slika 4-5) ovako kreirana inkrementalna datoteka se prosleđuje serveru za obradu i prikazuje se rezultat. Ako servis vrati uspešni rezultat, modifikuje se stanje baze upisom i brisanjem ID-jeva objekata koji su dodati odnosno obrisani.

4.3 Servis

Servis je odgovoran za snimanje stanja o sistemu putem čuvanja objekata dobijenih od strane importera. Tumači inkrementalne datoteke i vrši osvežavanje stanja. Definiše spregu za pristup podacima *Generic Data Access* (GDA). GDA omogućava pristup podacima bez potrebe da klijent poznaje implementaciju skladištenja podataka. Svaki podatak definiše se kao resurs, atribut i vrednost tog atributa. Svaki objekat koji se može identifikovati je resurs. Atribut je neka osobina tog resursa koja se može opisati. Vrednost svakog atributa mora poštovati domen odnosno dozvoljene vrednosti. Loguje svaku akciju koja se izvršava i njen rezultat u izveštajnu datoteku.

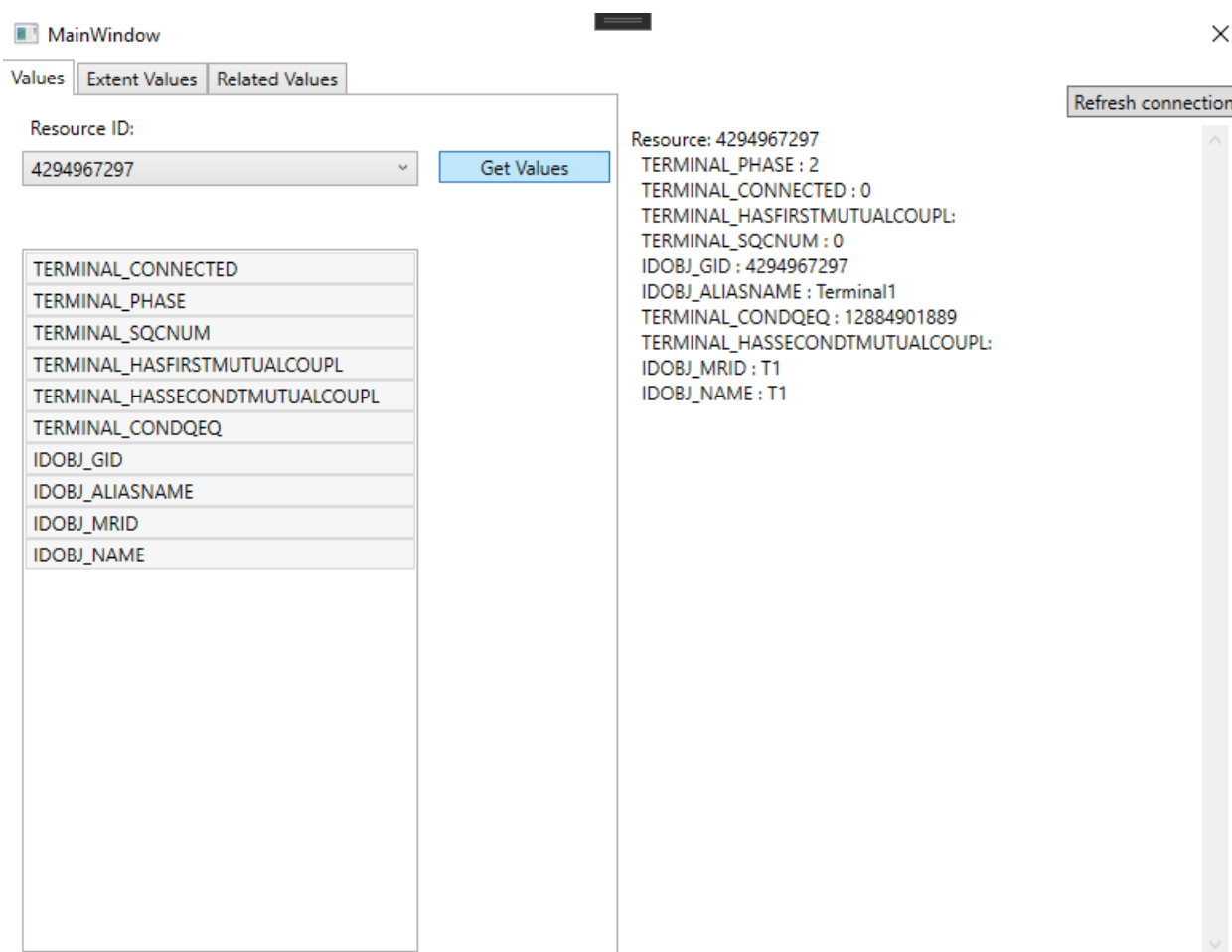
Stanje sistema modeluje klasom *Container*. Ključno svojstvo ove klase je rečnik koji mapira ID resursa na instancu objekta *IdentifiedObject*. Ranije je spomenuto kako je *IdentifiedObject* čvorna klasa koju sve ostale nasleđuju, te se svakoj instanci bilo koje CIM klase može pristupiti referencom na *IdentifiedObject*. Pored ovog rečnika klasa poseduje sve potrebne pomoćne metoda za upravljanjem kontejnera. Sam servis poseduje rečnik koji mapira tip klase (šifrovan enumeracijom *DMSType*) na kontejner svih resursa tog tipa.

Na ovaj način, za implementaciju *Get Extent Values* usluge (usluge su definisane u sekciji 2.) usluge sve što je potrebno je proveriti mapirani kontejner za određeni tip. Taj kontejner sadrži sve neophodne resursa koje treba vratiti klijentu servisa.

Za implementaciju *Get Values* usluge potrebno je utvrditi tip traženog resursa. To je moguće zbog načina definisanja identifikatora, šifrovan je na takav način da se u samom identifikatoru nalazi i šifra tipa. Nakon što je tip određen, sve što je preostalo je da se preuzme odgovarajući kontejner i u njemu pronađe objekat sa korektnim identifikatorom. Taj objekat se vraća klijentu kao povratna vrednost.

U slučaju *Get Related Values* usluge neophodno je definisati asocijaciju između izvornog resursa i rezultujućih resursa. To se vrši klasom *Association*. Kod nje su ključna polja *propertyId* i *type*. Polje *propertyId* ima vrednost nekog referentnog atributa resursa. Traže se takvi resursi da u svojem polju tipa *propertyId* referenciraju izvorni resurs (dalje rezultujući resursi). Polje *type* služi za filtriranje rezultujućih resursa. Moguće je ne definisati polje *type* i u tom slučaju se ne vrši filtriranje nad rezultujućim resursima. Nakon završetka ovog algoritma svi rezultujući resursi se vraćaju klijentu servisa.

4.4 Klijentska aplikacija

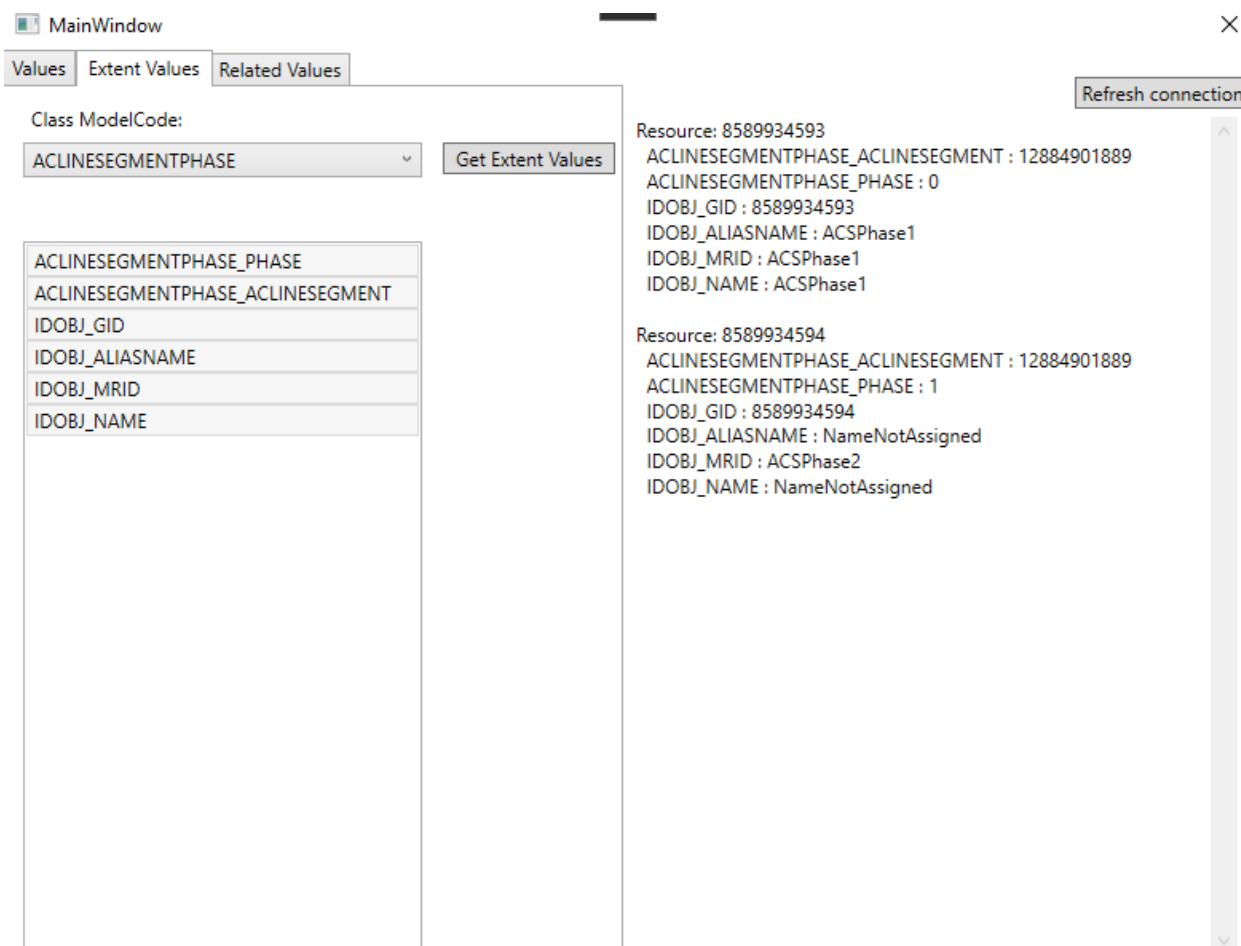


Slika 4-8 Values prozor klijentske aplikacije

Klijentska aplikacija omogućava uvid u sačuvane objekte na servisu. Aplikacija se povezuje sa servisom pri pokretanju. U slučaju otkaza ili neke greške moguće je osvežiti konekciju tasterom „Refresh connection“. Sve akcije u aplikaciji se loguju i zapisuju u izveštajnu datoteku. Aplikacija se sastoji od tri glavna prozora: „Values“, „Extent Values“ i „Related Values“. Ovi prozori odgovaraju implementaciji zahteva definisanih u sekciji 2.

U „Values“ prikazu (slika 4-8), aplikacija izlistava ID-jeve svakog objekta i omogućava selekciju jednog. Prepoznaje se tip izabranog objekta i izlistavaju imena njegovih atributa. Moguće je izabrati proizvoljan broj tih atributa i pri klikom na taster „Get Values“ izlistavaju se aktivne vrednosti tih atributa izabranog objekta.

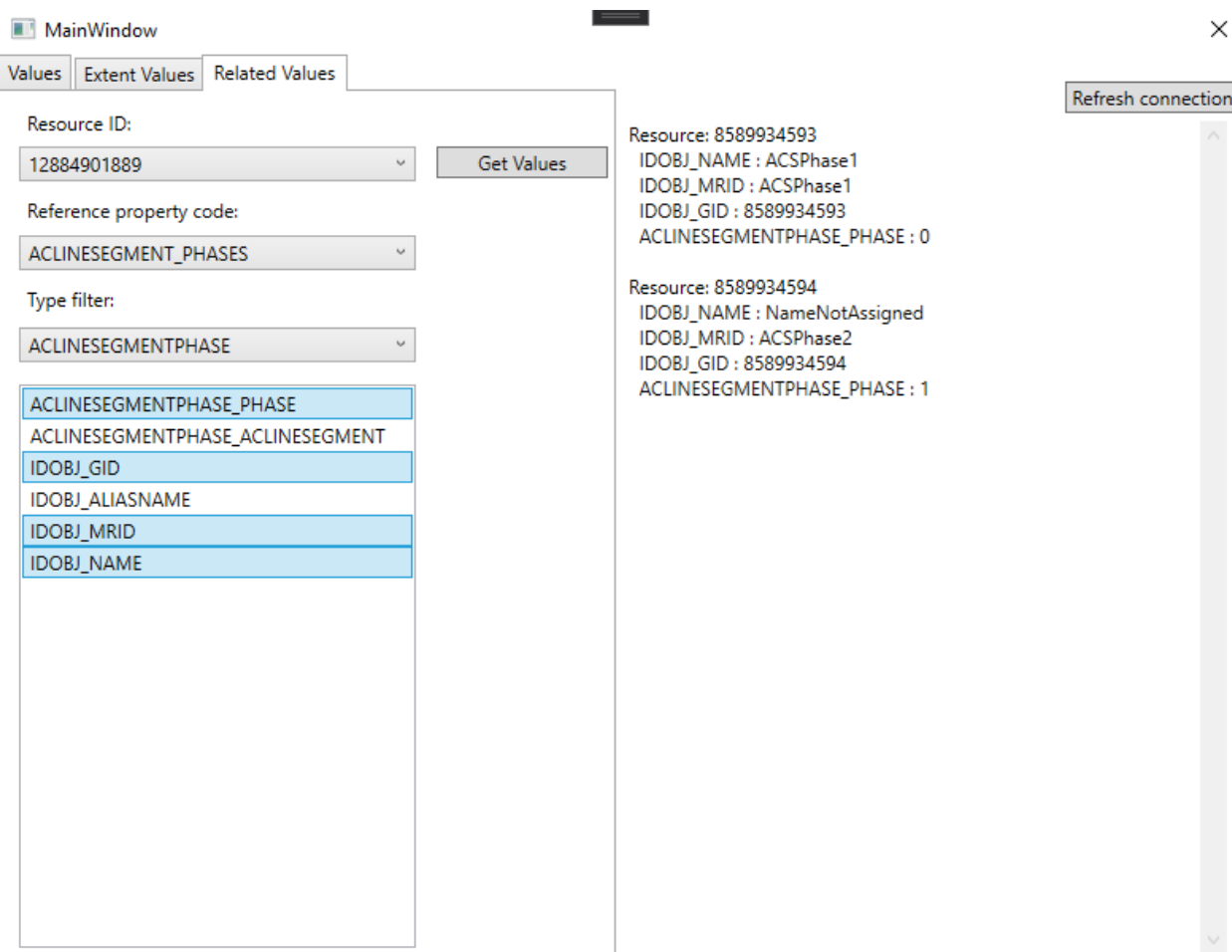
Na slici 4-8 se kao primer uzima scenario u kojem je izabran resurs sa ID-jem 4294967297. Aplikacija prepoznaje da se radi o resursu tipa *Terminal* te izlistava sve attribute ove klase. Izabrani su svi ponuđeni atributi i pritiskom na „Get Values“ taster izlistavaju se vrednosti svih ovih atributa sa desne strane aplikacije.



Slika 4-9 Extent Values prozor klijentske aplikacije

„Extent Values“ prozor (slika 4-9) izlistava sve prepoznate tipove u sistemu i omogućava odabir jednog. Ispod se izlistavaju svi atributi za taj tip i moguće je odabrati proizvoljan broj. Klikom na „Get Extent Values“ taster izlistavaju se svi objekti izabranog tipa i izabrani atributi.

Na slici 4-9 se vidi scenario gde je izabran tip *ACLLineSegmentPhase*. Aplikacija prepoznaje sve njegove attribute i izlistava ih ispod. Nakon pritiska na taster „Get Extent Values“ sa desne strane izlistavaju se vrednosti ovih atributa svih resursa klase *ACLLineSegmentPhase*. Na servisu se prepoznata dva objekte ovog tipa te su izlistana oba.



Slika 4-10 Related Values prozor klijentske aplikacije

„Related Values“ prozor (slika 4-10) izlistava ID-jeve svih objekata, kao i „Values“ prikaz. Zatim prepoznaje njegov tip i izlistava svaki njegov referentni atribut i omogućava izbor jednog. Referentni atributi mogu pokazivati na jedan objekat ili na listu više njih. Aplikacije ne prepoznaje tipove referenciranih objekata te nudi izbor između svih atributa nad svim tipovima u sistemu. Ako korisnik zna koji su tipovi referenciranih objekata, može podesiti „Type filter“

tako da se u ponudi prikažu samo atributi izabranog filtera. Klikom na „Get Values“ taster izlistavaju se svi referencirani objekti sa izabranim atributima.

Na slici 4-10 je za primer uzet resurs sa ID-jem 12884901889. Aplikacija prepoznaje da se radi o resursu tipa *ACLineSegment*. Za referentni atribut izabran je *ACLineSegment_Phases*. Za filter je ručno izabran tip koji odgovara referentnom, *ACLineSegmentPhase*. Nakon odabira filtera izlistani su samo odgovarajući atributi. Ovaj put nisi odabrani svi, od njih šest izabrana su četiri. *ACLineSegment_Phases* može da pokazuje na više resursa, u ovom slučaju dva, te su prikazani atributa za oba resursa.

5. Zaključak

U ovom radu prikazana je jedna implementacija sistema elektrodistributivne mreže. Sistem se sastoji od servisa, klijentske aplikacije, importera i kreatora profila. Specifikacija zahteva data je u sekciji 2.

U sekciji 3. objašnjene su potrebne teorijske osnove za razumevanje tehnologija korišćenih u radu, uključujući programski jezik C#, .Net, *EntityFramework*, WPF i SQL. Objašnjeno je šta je *Common Information Model* i koji je njegov značaj u modernom društvu. Na slici 3-1 prikazan je jedan primer zapisa klase u CIM/XML formatu.

U sekciji 4. prikazuju se detalji implementacije sistema. Na slici 4-1 se prikazuje dijagram realizacije sistema. Definisan je pojam CIM profila i detaljno je pojašnjen profil u implementaciji ovog rada. Prikazana je scenario upotrebe aplikacije Profil Kreator u svrsi generisanja DLL biblioteke sa definicijama klasa CIM modela. Prikazana je aplikacija Importer i objašnjen je scenario učitavanja datoteke stanja realnog sistema. Zatim je objašnjeno kako se vrši konverzija klasa CIM modela u generični opis resursa (klasa *ResourceDescription*). Objašnjena je svrha i definicija inkrementalne delta datoteke. Detaljno je objašnjen serverski mehanizam za ispunjenje usluga GDA korisničke sprege. Pokazale su se karakteristike aplikacije za pregled stanja mreže koristeći serversku GDA spregu i dat je primer scenario za upotrebu svake definisane usluge.

Pokazan je način da se preko delta fajlova dinamički i efikasno osvežava stanje mreže. Ovakvom implementacijom smanjuje se teret komunikacione mreže zbog manjeg upita kao i same serverske obrade. Moguće je da server efikasno pamti svaku pristiglu deltu u cilju redundantnosti podataka. Za ove benefite plaćena je mala cena u teretu obrade importera i zauzeća memorije.

6. Literatura

- [1] <https://www.dmtf.org/standards/cim/> (novembar 2021)
- [2] <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/> (novembar 2021)
- [3] <https://docs.microsoft.com/en-us/dotnet/standard/managed-code> (novembar 2021)
- [4] <https://www.java-samples.com/showtutorial.php?tutorialid=1425> (novembar 2021)
- [5] <https://docs.microsoft.com/en-us/visualstudio/designers/getting-started-with-wpf>
(novembar 2021)
- [6] <https://docs.microsoft.com/en-us/dotnet/desktop/wpf/advanced/wpf-architecture?redirectedfrom=MSDN&view=netframeworkdesktop-4.8> (novembar 2021)
- [7] <https://searchdatamanagement.techtarget.com/definition/SQL> (novembar 2021)
- [8] http://webstore.iec.ch/preview/info_iec61968-1%7Bed1.0%7Den.pdf (novembar 2021)
- [9] <http://www.iec.ch/search/?q=61970> (novembar 2021)
- [10] <http://tc57.iec.ch/index-tc57.html> (novembar 2021)
- [11] <https://www.w3.org/RDF/> (novembar 2021)
- [12] <https://www.w3.org/TR/rdf-schema/> (novembar 2021)
- [13] <https://wiki.cimtool.org/> (novembar 2021)
- [14] <https://www.iea.org/reports/smart-grids> (novembar 2021)
- [15] <https://www.entsoe.eu/digital/common-information-model/> (novembar 2021)