UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Aleš Glavina

Oddaljena konfiguracija servisov v sistemu SEP2W.NET

DIPLOMSKA NALOGA
NA UNIVERZITETNEM ŠTUDIJU

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

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

Št. naloge: 01535/2009
Datum: 15.01.2009

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

Kandidat: **ALEŠ GLAVINA**

Naslov: **ODDALJENA KONFIGURACIJA SERVISOV V SISTEMU SEP2W.NET**

THE REMOTE CONFIGURATION OF SEP2W.NET SYSTEM'S SERVICES

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Kandidat naj v svojem diplomskem delu opiše izgradnjo programske opreme za realizacijo oddaljene konfiguracije servisov v sistemu SEP2W.NET za daljinsko krmiljenje in odčitavanje električnih števcev. Pri tem naj kandidat predstavi sam sistem SEP2W.NET, njegovo zgradbo, delovanje konfiguratorja in analizira izvedeno programsko rešitev.

Mentor:

prof. dr. Miha Mraz

Dekan:

prof. dr. Franc Solina

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Aleš Glavina

Oddaljena konfiguracija servisov v sistemu SEP2W.NET

DIPLOMSKA NALOGA
NA UNIVERZITETNEM ŠTUDIJU

Izr. prof. dr. Miha Mraz
MENTOR

Ljubljana, 2009

Zahvala

Na tem mestu bi se najprej zahvalil družini in prijateljem, ki so mi pomagali in stali ob strani v času celotnega študija.

Zahvaljujem se mentorju izr. prof. dr. Mihi Mrazu, ki mi je izdatno svetoval in usmerjal pri izdelavi diplomskega dela. Hvala tudi podjetju Iskraemeco d.d., mentorju Primožu Kožuhu in ostalim sodelavcem za posredovanje napotkov, ki so mi pri izdelavi diplomskega dela zelo koristili.

Posebej bi se rad zahvalil Ani Jurkovič, ker je diplomsko nalogo slovnično pregledala in mi pri pisanju stala ob strani.

--- Aleš Glavina, Marezige, januar 2009

Kazalo

Seznam uporabljenih kratic

- **AMR** (angl. **Automated Meter Reading**): sistem avtomatskega branja števcev. Sistem AMR vključuje strojno in programsko opremo.
- **AS** (angl. **SEP2 Alarm Service**): storitev za odzivanje na alarme v sistemu SEP2W.
- **CS** (angl. **SEP2 Core Service**): storitev za verifikacijo uporabnikov in urejanje uporabniških pravic v sistemu SEP2W.
- **DLC** (angl. **Distribution Line Carrier**): sistem za prenos podatkov po električnih vodnikih srednje, nizke in hišne napetosti.
- **DOM** (angl. **Document Object Model**): razčlenjevalnik XML dokumentov, ki predstavi drevesno strukturo XML dokumenta, zapisanega v delovnem pomnilniku.
- **EWS** (angl. **Encryption Web service**): izdaja kupljene ključe.
- **GPRS** (angl. **General Packet Radio Service**): mobilna podatkovna storitev v okviru standarda GSM.
- **GSM** (angl. **Global System for Mobile communications**: izvorno od ***Groupe Spécial Mobile***): najpopularnejši svetovni standard mobilnih komunikacij.
- **IP** (angl. **Internet Protocol**): protokol za komunikacijo v omrežju.
- **IS** (angl. **SEP2 Integration Service**): omogoča izvajanje predplačniških operacij na spletu.
- **LAN** (angl. **Local Area Network**): interno računalniško omrežje.
- **LS** (angl. **SEP2 Listener Service**): storitev za sprejemanje alarmov v sistemu SEP2W.
- **MAS** (angl. **SEP2 Meter Access Service**): storitev dostopa do merilnih naprav v sistemu SEP2W.
- **MRS** (angl. **SEP2 Meter Reading Service**): storitev za branje naprav v sistemu SEP2W.
- **MSMQ** (angl. **Microsoft Message Queue**): Microsoftova izvedba sporočilnih vrst.
- **PDA** (angl. **Personal Digital Assistant**): dlančnik.
- **PLC** (angl. **Power Line Communication** ali **Power Line Carrier**): sistem za prenos podatkov po električnih vodnikih.
- **POS** (angl. **SEP2 POS**): preprost uporabniški vmesnik za nakup/plačevanje elektrike in tiskanje preprostih poročil.
- **PP** (angl. **SEP2 Prepayment Plugin**): grafični vmesnik, ki omogoča urejanje strank in ponudnikov električne energije, ter njen nakup v sistemu SEP2W.
- **PS** (angl. **SEP2 Prepayment Service**): storitev za izvajanje predplačniških operacij v sistemu SEP2W.
- **RS** (angl. **SEP2 Report Service**): storitev za izvajanje poročil v sistemu SEP2W.
- **SAX** (angl. **Simple API for XML**): razčlenjevalnik XML dokumentov, ki je dogodkovno naravnan.
- **SCH** (angl. **SEP2 Scheduler Service**): storitev za avtomatsko izvajanje opravil v sistemu SEP2W.
- **SEP** (**Sistem Elektronskega Postrojenja**): programski paket, ki omogoča oddaljeno branje in upravljanje z merilci energije na terenu.
- **SMS** (angl. **Short Message Service**): komunikacijski protokol, ki nam omogoča izmenjavo kratkih sporočil med mobilnimi telefoni.

- **XML** (angl. **Extensible Markup Language**): enostaven tekstovni format namenjen predvsem izmenjavi podatkov na internetu.
- **WAN** (angl. **Wide Area Network**): računalniško omrežje, ki povezuje med seboj ustanove, kraje, države in celine.

Povzetek

Kljub temu, da imamo danes razvito primerno tehnologijo, odčitavanje števcev poteka še vedno ročno. Avtomatično odčitavanje si sicer utira pot v elektrodistribucijska podjetja, vendar s počasnim tempom. Podjetja se sicer zavedajo potrebe po minimizirani porabi energije in natančnem upravljanju z njo. Podjetje Iskraemeco d.d. je razvilo sistem za avtomatsko odčitavanje in upravljanje z infrastrukturo. Le-ta naj bi bil sposoben upravljati z 1.000.000 števci na terenu. Še pred kratkim je bil sistem v fazi razvoja, v tem času pa se je večino truda vlagalo v funkcionalnost. Sedaj se nam obeta izdaja prve različice sistema z imenom *Base*. Kljub temu nadaljujemo z razvijanjem funkcionalnosti, izmed katerih je tudi daljinska konfiguracija servisov. Dandanes si ne moremo privoščiti, da bi vsak servis posebej konfigurirali na svojem računalniku, kajti ni nujno, da se le-ta nahaja v naši bližini. Ker nam tehnologija omogoča cenejše in pametnejše rešitve, smo se tudi v Iskraemecu odločili, da bomo servise konfigurirali z enega mesta. To nam omogoča grafični vmesnik, v katerem lahko upravljamo vse servise, ki tečejo na določenem računalniku. Poleg tega pa si olajšamo samo upravljanje servisov, kajti obdelan XML dokument je veliko bolj berljiv.
V diplomski nalogi sem tako predstavil sistem SEP2W.NET. Poudarek sem dal na servise, njihovo delovanje in konfiguracijske parametre, ki so najpomembnejši del naloge. Poleg opisa parametrov je bila moja naloga uporabniku olajšati delo, torej narediti grafični vmesnik, ki ni samostojen program, temveč del Managerja in omogoča urejanje konfiguracijskih datotek.

Ključne besede : *Oddaljena konfiguracija, SEP2W.NET, servisi, elektrika*.

Abstract

Manual reading of counters is still very common today, despite the availability of appropriate technological solutions. Automatic reading is gaining importance in electrodistribution companies, but at a very slow pace. The distribution companies are well aware of the needs to rationalize the use of electrical power and to minimize losses. Iskraemeco developed a system for automatic reading and infrastructure management which can remotely manage up to 1 million counters. Even a short time ago, the system was still in development and the biggest efforts were put in funcional improvement. The first version of the system, named *Base*, is to be released soon, but development and improvement are still ongoing. An important improvement is remote service configuration. It is not rational to configure each service on its own computer, as some can be far away and since today's techology allows better and cheaper alternatives, we at Iskraemeco decided to configure services remotely from one central station. A graphic interface allows us to configure all services that run simultaneously on one computer and, at the same time, eases the very managment of services, since a modified XML document is much more readable. My graduation thesis is aimed at presenting the SEP2W.NET system with emphasys on services, their functions and configuration parameters. Those are the crucial point of the thesis, as they are the ones who need to be continuously edited. Despite the very description of parameters, there was a need to improve the user's experience, so I created a graphical interface for configuration files editing, which is not an independent application, but is part of the Manager.

Key words: *remote configuration, SEP2W.NET, services, electricity.*

1 Uvod

Vse večja okoljevarstvena osveščenost je tudi v elektroindustriji pokazala potrebe po natančnejšem vpogledu nad porabo in smotrnim upravljanjem z električno energijo. Podjetja bi rada nenehno nadzorovala porabo električne energije, njene izpade, nedovoljene porabe, itd. Vse to bi pripomoglo k hitrem ugotavljanju in odpravljanju težav ter zmanjšanju porabe električne energije, kar posledično pomeni manjše onesnaževanje okolja. Ravno zaradi tega vidim v tej smeri možnosti za razvijanje. V Iskraemecu d.d. smo tako razvili sistem, ki omogoča avtomatsko odčitavanje števcev in upravljanje z električnim omrežjem. Sicer na tržišču že obstajajo podobni sistemi, ki pa ne zmorejo upravljati več kot 30.000 naprav. Naš sistem se imenuje SEP2W.NET in omogoča upravljanje z 1.000.000 napravami. Ker sistem ni še popolnoma dokončam, se mu dodajajo funkcionalnosti, se ga še spreminja in prilagaja željam uporabnikov.

Diplomska naloga obravnava daljinsko konfiguracijo servisov, ki je ena novih funkcionalnosti v sistemu. Omogoča pa preprosto nadziranje servisov, ki delujejo v sistemu. Trenutno je njihovo upravljanje zelo nerodno, konfiguriramo jih lahko samo na računalniku, kjer so nameščeni, tako da ročno obdelamo XML dokument. Z grafičnim vmesnikom oddaljene konfiguracije pa vse te nevšečnosti odpravimo in uporabniku olajšamo delo.

V drugem poglavju sem najprej predstavil arhitekturo elektrodistribucijskega omrežja in možnosti, ki jih ta struktura ponuja. Nadaljeval pa sem z opisom sistema SEP2W.NET, ki s to arhitekturo upravlja.

Tretje poglavje se začne z opisom, ki nam pove, kako je realizirana konfiguracija servisov, zakaj je ravno takšna in kako upravljamo z njo. Sledi opis posameznih servisov in njihovih konfiguracijskih parametrov. V nadaljevanju sem se posvetil sami izvedbi daljinske konfiguracije, temu, kako je sprogramirana, kako deluje in zakaj ne drugače.

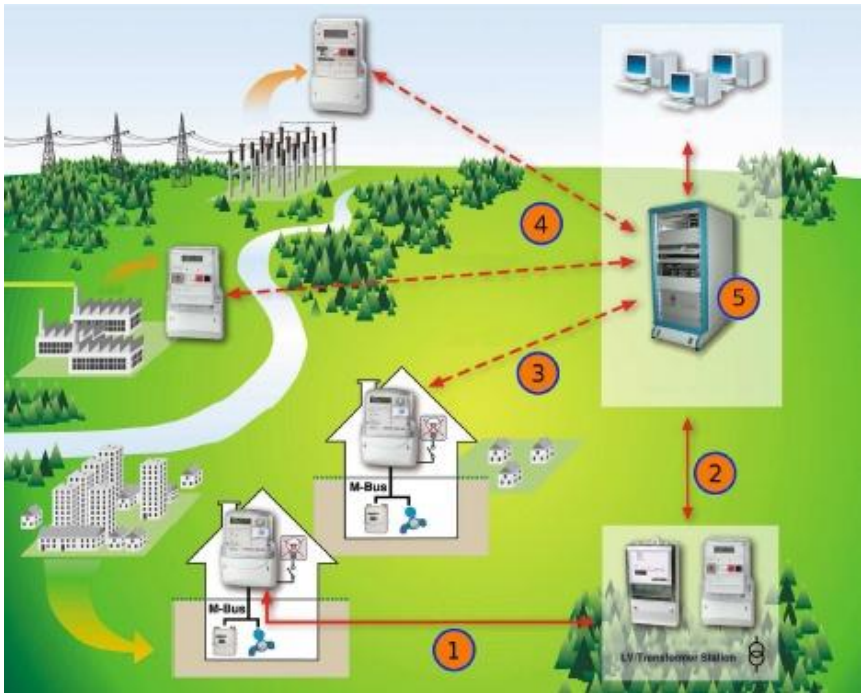
Četrto poglavje posveča nekoliko pozornosti problemom, ki so nastali, in možnim izboljšavam, torej, kaj bi lahko naredili lepše, boljše, uporabniku bolj prijazno.

Zadnje poglavje pa zaključuje moje delo s krajšim pogledom na dosežene cilje in pričakovanjem uporabnikovih odzivov na nov SEP2W.NET sistem.

2 Opis sistema

2.1 Splošno o elektroindustriji

Elektrodistribucijska podjetja električno energijo po večini zaračunavajo pavšalno, torej določijo nek znesek, ki se ga tekom leta plačuje, in je zasnovan na podlagi porabe v predhodnem letu, na koncu leta pa naredijo obračun. To zahteva ročno popisovanje stanja na števcu, kar pa je dolgotrajen in ne povsem varen postopek. Popisovalci se srečujejo z različnimi problemi, od tega, da nimajo dostopa do električnega števca, ker stranke ni doma, do tega, da popisovalca napade pes, itd. To ni edini način plačevanja, saj se lahko odločimo tudi za drug način; vsak mesec distribuciji javimo novo stanje na števcu in tako dobimo na dom položnico za tekočo porabo. Vendar ta način elektrodistribuciji ne prihrani nič dela, kajti enkrat letno je potrebno popisati stanje na števcu, v nasprotnem primeru bi stranke hitro spoznale, da lahko sporočajo bistveno manjše zneske porabe in bi na tak način oškodovale distributerja. Potem pa imamo še industrijo, ki postaja vse bolj zahtevna in bi rada imela podatke o porabi električne energije vsakih petnajst minut, radi bi vedeli, kdaj so bili izpadi, kdaj konice v porabi in še mnogo drugih stvari.



Slika 1: AMR sistem.

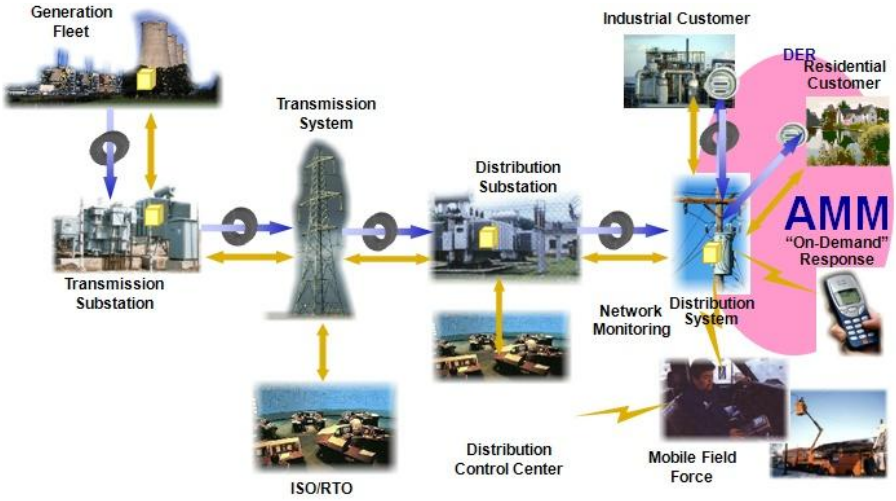
Te težave so podjetja pripeljala do dejstva, da je na tem področju potrebno nekaj spremeniti. Pojavila se je ideja o sistemu, ki je zadolžen za avtomatsko zbiranje podatkov s števcev, ne

nujno električnih, lahko tudi plinskih, vodnih (skratka vse, kar se meri) ter hranjenje in obdelavo podatkov. Poznamo več sistemov, ki nam to omogočajo, mi pa se bomo osredotočili na AMR (angl. *Automated Meter Reading*), ki predstavlja sistem za avtomatsko branje, merjenje in obračun porabe električne energije (glej sliko 1). Za prenos podatkov z električnega števca proti zbirnemu AMR centru podatkov se uporablja že obstoječe komunikacijske sisteme, kot sta GSM, GPRS itd. Znotraj AMR sistema je lahko uporabljenih več tehnologij, razdelil pa jih bom v dve večji skupini:

- Ročno branje: ta način se pravzaprav ne razlikuje od klasičnega popisovanja števcev. Uslužbenec podjetja neko napravo (prenosni računalnik, PDA ...) z uporabo fizične povezave (žice) poveže s števcem in sproži njegovo branje.

- Oddaljeno branje *z ročnimi napravami*: je precej podobno ročnemu branju, le da tu med napravo za zbiranje podatkov in števcem komunikacija poteka prek brezžične povezave – *prek fiksnega omrežja:* v tem primeru se za prenos podatkov uporablja že vzpostavljeno ali posebej v ta namen vzpostavljeno omrežje. Izkoristi se lahko brezžični ethernet, GSM in GPRS omrežja, PLC komunikacijo, razno kombiniranje omrežij, itd. Topologija omrežja je najpogosteje zvezdasta. Več števcev je tako ali drugače v navezi s koncentratorjem, ki podatke iz števcev zbira in jih posreduje naprej zbirnemu centru, števci pa s koncentratorjem komunicirajo prek DLC tehnologije.

Ker pa ljudje težimo k izboljšavam, so podjetja naredila še korak naprej in izboljšala sistem AMR. Dodali so mu še nekaj funkcionalnosti in tako je nastal sistem AMM (angl. *Automatic Meter Management*).

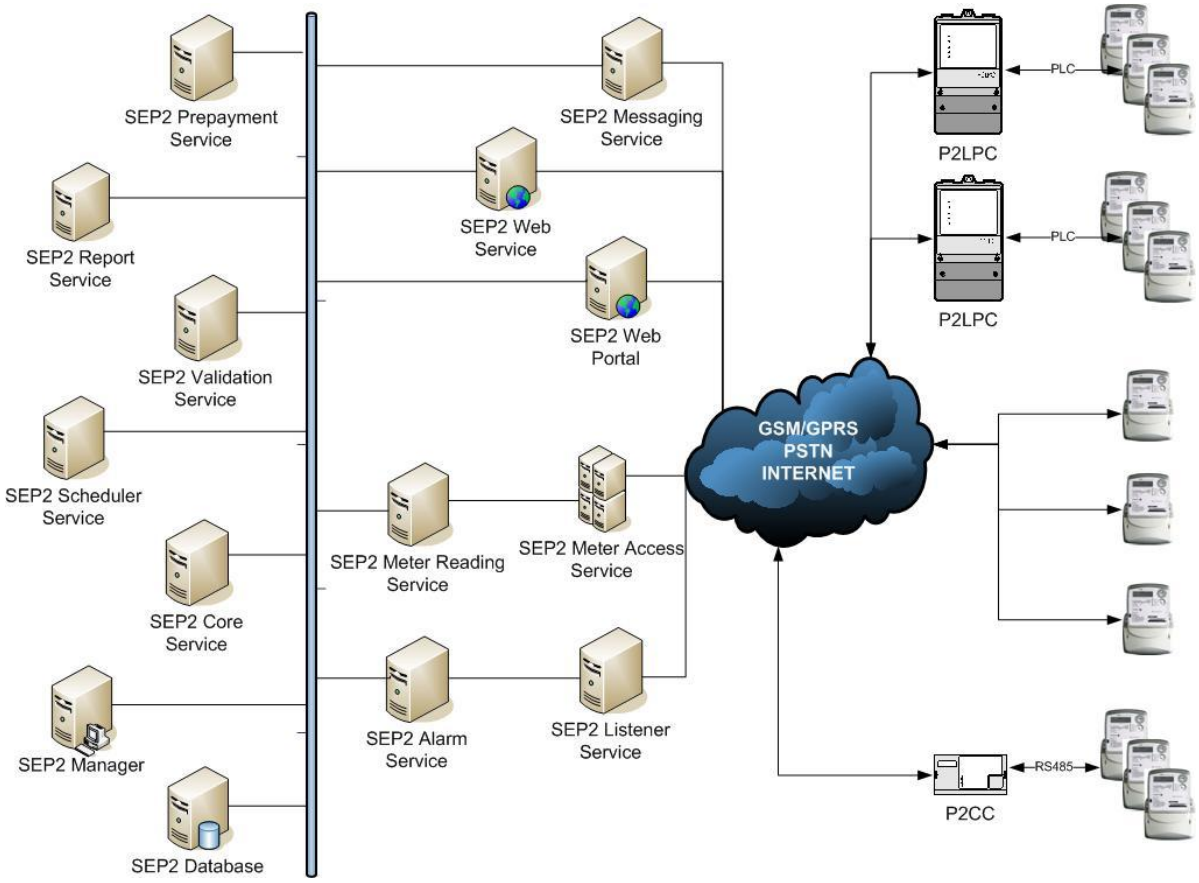


Slika 2: AMM sistem.

AMM sistem (glej sliko 2) poleg funkcionalnosti sistema AMR omogoča še upravljanje z omrežjem. S tem sistemom je omogočeno ugotavljanje napak na terenu, nekatere izmed njih je mogoče odpraviti, mogoče je realizirati predplačniško porabo energije, itd.

2.2 SEP2W.NET

Naj najprej povem, kaj pomeni ime SEP. Kratica je akronim za Sistem Elektronske Postrojenosti, izvira pa še iz časov samoupravljanja in je celo avtorsko zaščitena. Prvi SEP je bil napisan za okolje DOS, naslednji SEPW pa že za operacijski sistem Windows. Sledil je SEP2W (druga verzija za Windows okolje), najnovejša pa se imenuje SEP2W.NET (glej sliko 3). Dodatek .NET pomeni, da je zasnovan na .NET tehnologiji.



Slika 3: SEP2W.NET sistem.

Da ne bo nesporazumov, dovolite, naj pojasnim še to: SEP2W.NET ni nadgradnja sistema SEP2W. Deluje na popolnoma drugi bazi in nima s SEP2W ničesar skupnega, zato bi ga lahko poimenovali tudi SEP3W.

Zdaj pa preidimo na opis sistema. Od tu dalje bom uporabljal kratico SEP, s čimer je mišljen sistem SEP2W.NET.

2.2 SEP2W.NET

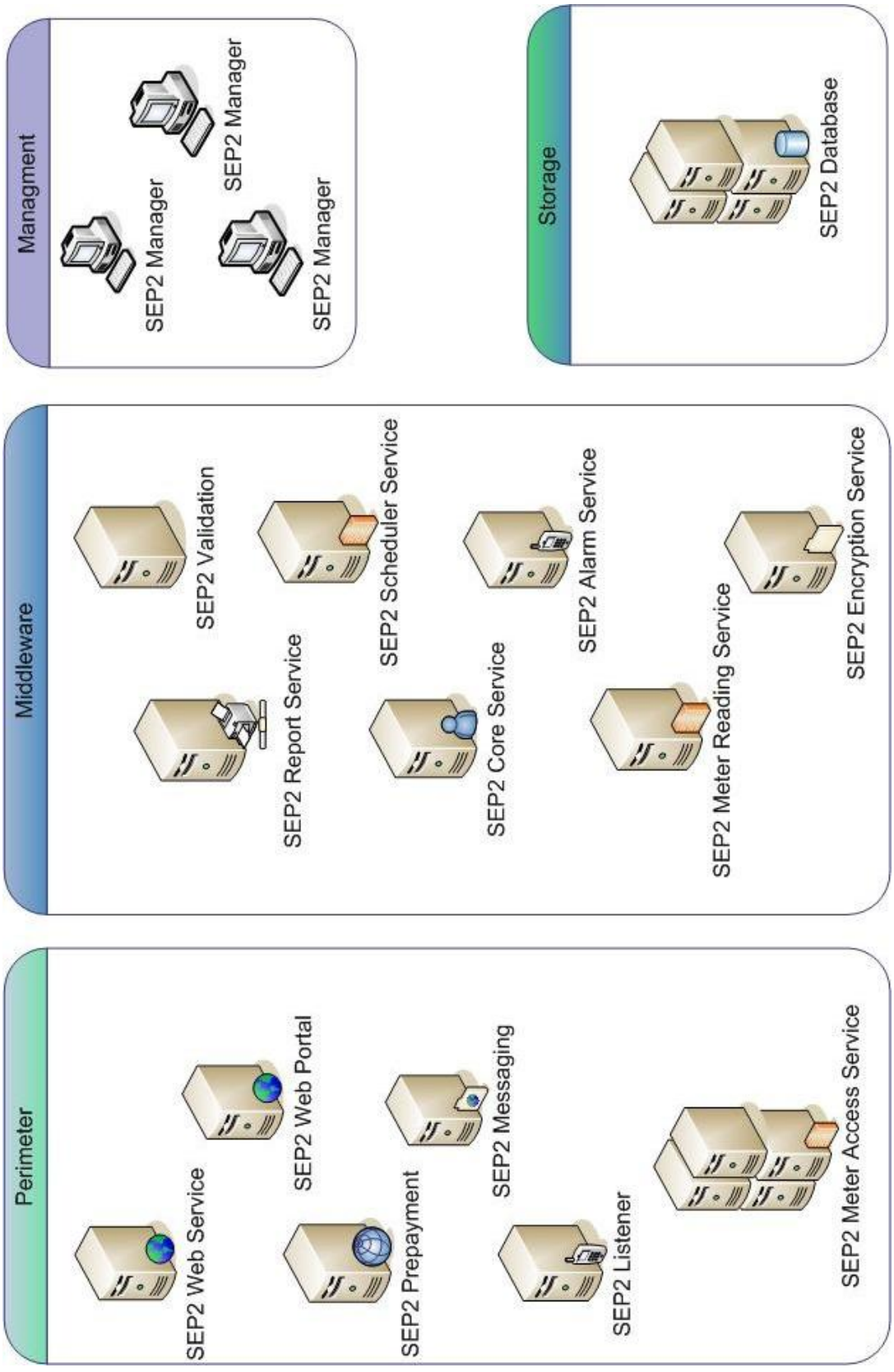
Sistem SEP2 (glej sliko 3) predstavlja implementacijo AMM sistema, ki opravlja prenos in obdelavo podatkov, ki jih zajemamo z merilnimi napravami. Sistem je sestavljen iz več komponent, kar olajša načrtovanje, implementacijo in predvsem vzdrževanje. Komponente sistema so razdeljene v štiri varnostne sekcije (glej sliko 4): *Perimeter, Middleware, Storage, Managment*.

Prvo sekcijo, z imenom *Perimeter,* sestavljajo komponente sistema, ki komunicirajo z merilnimi napravami, sistemi zunanjih strank in internetnimi uporabniki. Da bi minimizirali možne varnostne težave, se morajo komponente na tem nivoju podrejati varnostni politiki sistema SEP2. Posledično storitve na tem nivoju nimajo neposrednega dostopa do SEP2 podatkovne baze, lahko pa komunicirajo s storitvami srednjega nivoja. Komponente *Perimeter* sekcije so:

- **SEP2 Meter Access Service**: uporablja modeme PSTN/GSM, omrežje GPRS in serijska vrata za dostop do vseh naprav v sistemu.
- **SEP2 Listener Service**: uporablja modeme GSM in omrežje GPRS za sprejemanje alarmnih sporočil iz različnih naprav.
- **SEP2 Messaging Service**: uporablja FTP, elektronsko pošto, datotečni sistem in sporočilne vrste MSMQ za pošiljanje oziroma sprejemanje sistemskih podatkov.
- **SEP2 Web Service**: predstavlja zunanji vmesnik v obliki spletnih storitev, ki omogoča integracijo sistema SEP v zunanje sisteme strank.
- **SEP2 Web Portal**: predstavlja spletni portal namenjen končnim odjemalcem, ki omogoča pregled nad obstoječo, trenutno in predvideno porabo, itd.

Drugi sekciji, z imenom *Middleware,* ki predstavlja tako imenovani srednji sloj, bi lahko rekli tudi jedro sistema. V njej se nahajajo moduli, ki omogočajo komunikacijo s komponentami prve sekcije in vršijo dostop do SEP podatkovne baze. Moduli *Middleware* sekcije so:

- **SEP2 Meter Reading Service**: generira zahteve za branje naprav z uporabo podatkov o napravah shranjenih v SEP podatkovni bazi, jih posreduje SEP2 Meter Access strežnikom in pridobljene podatke shrani nazaj v SEP podatkovno bazo.
- **SEP2 Alarm Service**: interpretira alarme, ki jih sporoča SEP2 Listener Service, jih shrani v SEP podatkovno bazo ter obvesti administratorja sistema oziroma stranko o morebitnih težavah.



Slika 4: Sistem SEP2W.NET razdeljen na 4 sekcije.

- **SEP2 Validation Service**: preveri pridobljene podatke z uporabo predhodno definiranih testov. Preverjene podatke ustrezno označi in pripravi poročilo o uspehu, ki omogoča naknadno ročno preverjanje podatkov.
- **SEP2 Report Service**: izdeluje uporabniško definirana poročila bazirana na podatkih v SEP2 podatkovni bazi. Poročila se lahko shranijo v SEP2 podatkovno bazo, na datotečni sistem, ali se prikažejo na ekranu, itd.
- **SEP2 Scheduler Service**: izvaja naloge v sistemu SEP v skladu z definiranimi urniki. Naloga predstavlja delovni tok sestavljen iz opravil organiziranih v drevesno strukturo. Posamezno opravilo predstavlja operacijo izvršeno v enem izmed sistemskih modulov (SEP2 Meter Reading Service, SEP2 Validation Service, SEP2 Report Service ali SEP2 Messaging Service).
- **SEP2 Core Service**: opravlja nadzor nad licencami v sistemu, upravlja s pravicami uporabnikov in storitev ter implementira podporne funkcije za SEP2 Web Service in SEP2 Web Portal modula.

Tretjo sekcijo, z imenom *Storage,* predstavlja sistem za hrambo podatkov. V sistemu SEP se praktično vsi podatki hranijo v SEP podatkovni bazi, ki hrani podatke o:

- topologiji merilnega omrežja,
- merilnih rezultatih,
- dogodkih v sistemu,
- urnikih, po katerih se izvajajo naloge v sistemu,
- opravilih, ki sestavljajo naloge
- itd.

SEP2 podatkovna baza sestoji iz več ohlapno povezanih tabel, kar omogoča, da se podatki hranijo na enem ali več podatkovnih strežnikih. V prvem podatkovnem strežniku se na primer lahko hranijo podatki o merilnem sistemu in porabi, na drugem pa informacije o strankah, izdelana poročila, itd. Ne glede na to, ali so podatki porazdeljeni med več strežnikov ali ne, jih uporabnik sistema vedno vidi na enoten način.

Zadnja sekcija, z imenom *Managment*, omogoča nadzor nad sistemom SEP. Ker je iz te sekcije možen dostop do vseh, prej naštetih, sekcij, je možno z uporabo aplikacije SEP2 Manager (glej sliko 5) preprosto upravljati in nadzorovati delovanje celotnega sistema.

Slika 5: Aplikacija SEP2 Manager.

Programiranje je potekalo v programskem jeziku C#, kot razvojno orodje pa je bil na začetku uporabljen Microsoftov Visual Studio 2005, pred kratkim pa smo prešli na Microsoftov Visual Studio 2008. Menim, da bomo v naslednjih letih prešli na Microsoftov Visual Studio 2010. Oba zadnja konstrukta se naslanjata na .NET 2.0, oziroma sedaj na .NET 3.5 tehnologijo. Kar zadeva uporabniški vmesnik, smo predvsem zaradi uporabe knjižnic Krypton Toolkit in CodeJock omejeni na Windows okolje, medtem ko naj bi se komponente prvega in drugega sklopa prevedle tudi za operacijski sistem Linux.

2.3 Jezik XML

XML je jezik za opis podatkov. Omogoča opis podatkov v obliki teksta, kjer je struktura podatkov predstavljena na drevesni način. Večina podatkovnih formatov pred XML je bilo binarnih in so običajno služili le določenemu namenu. Osnovna enota dokumenta XML je znak, čigar kodiranje je navedeno na začetku dokumenta. Obliko dokumenta definira shema XML, ki je prav tako shranjena v XML dokumentu. Shema določa omejitve v strukturi in v podatkovnih tipih dokumenta, lahko pa vsebuje tudi privzete vrednosti za elemente. Shema lahko določa tudi imenski prostor, ki bo opisan v nadaljevanju. Za izdelavo shem obstajajo različna orodja (npr. XMLSpy), ustrezno orodje pa vsebuje tudi Visual Studio 2005 (XMLDesigner). XML dokument je sestavljen iz elementov. Tipi elementov so definirani v shemi, posameznemu elementu pa lahko pripada tudi eden ali več atributov.
Dokument XML mora za veljavnost izpolnjevati dva pogoja:

- sintaktična pravilnost: dokument se drži vseh XML sintaktičnih pravil, npr. vsak neprazen element ima začetno in končno označbo,
- podatkovna pravilnost: podatki v dokumentu ne kršijo omejitev definiranih v shemi.

Za preverjanje pravilnosti dokumentov obstajajo različna orodja, programski vmesnik za preverjanje pravilnosti XML dokumentov pa najdemo tudi v ogrodju .NET (npr. razred XmlValidatingReader). Odsotnost imenskih prostorov (angl. *namespace*) je ena večjih pomanjkljivosti v specifikaciji jezika XML 1.0. Z uporabo imenskih prostorov je ime vsakega elementa enolično, določeno kot imenski_prostor:ime_elementa. Brez imenskih prostorov pride do zmede pri imenovanju elementov, saj lahko primernih imen hitro zmanjka, še težavnejše pa je razčlenjevanje. Razčlenjevalnik namreč ne more ločiti med različnimi elementi z enakim imenom. Primer takega elementa je <*ime*>. Le-ta lahko v enem primeru pomeni ime osebe, v drugem pa ime neke strukture. Zmedi se izognemo, če zapišemo npr. <oseba:ime>. Imenski prostor deklariramo v elementu z uporabo rezervirane besede xmlns. Na primer *xmlns:programiranje=“abc“* deklarira imenski prostor programiranje z označbo abc. Ime imenskega prostora mora biti enolično. Enoličnost najlažje dosežemo tako, da na spletu registriramo domeno in jo uporabimo za ime. Jezik XML ima tudi nekatere slabosti. Glavna med njimi je, da je sintaksa preobširna in pogosto odvečna (angl. *redundant*). Ta težava je najbolj izrazita pri prenosu, ko imamo opraviti z omejeno pasovno širino. Slabost je tudi zahtevno sestavljanje in razčlenjevanje dokumentov. Za razčlenjevanje dokumentov XML sta na voljo dva vmesnika: DOM in SAX.

2.3.1 Model DOM

DOM je model predstavitve dokumentov XML z objekti. Definira tudi vmesnik, ki omogoča pregled in urejanje dokumentov. Z uporabo DOM-a so dokumenti predstavljeni v drevesni strukturi. Taka predstavitev zahteva, da je v pomnilniku naložen celoten dokument. Uporaba DOM-a je najbolj smotrna, ko do elementov dostopamo naključno. Zaradi velike porabe pomnilnika pa model ni priporočljiv, ko je potrebna obdelava velikih dokumentov. Uporaba ni primerna niti za zaporedno branje celotnih dokumentov. Model DOM je v ogrodju .NET implementiran v razredu XmlDocument, ki omogoča upravljanje nad objekti tipa XmlNode. Vsak element v dokumentu je v pomnilniku predstavljen kot objekt tipa XmlNode.

2.3.2 Model SAX

SAX je vmesnik za zaporedni dostop do dokumentov XML. V tem primeru je razčlenjevanje dokumenta možno le v smeri branja. Ko se nahajamo na elementu *n* in želimo ponovno prebrati element *n-1*, je potrebno dokument ponovno razčleniti od začetka. Po drugi strani pa lahko razčlenjevanje prekinemo, ko je želeni element prebran. Ker v pomnilniku ni potrebno hraniti celotne drevesne strukture naenkrat (ampak samo trenutni element), je SAX hitrejši in pomnilniško manj zahteven od DOM-a. Razlika postane izrazita zlasti pri večjih dokumentih. SAX razčlenjevalniki so dogodkovno usmerjeni, kar pomeni, da omogočajo proženje določenih dogodkov, ko naletijo na določen element v dokumentu. Glavna slabost SAX-a je, da omogoča le branje in ne tudi urejanja dokumentov. V ogrodju .NET je SAX implementiran v razredu XmlReader.

3 Oddaljena konfiguracija

Cilj pričujočega dela je izdelava uporabniku prijaznega vmesnika za konfiguracijo posameznega servisa, ki je aktiven na določenem računalniku. Konfiguracija je pomembna za samo delovanje in stabilnost sistem, kajti ravno servisi so tisti, ki opravljajo branje s števcev na terenu. Za lažje razumevanje bom najprej pojasnil, kako lahko servise sploh konfiguriramo, nato bom podal njihov opis in delovanje, predvsem pa možne nastavitve. Na koncu se bom osredotočil še na uporabniški vmesnik, njegovo zgradbo ter implementacijo.

3.1 Splošne konfiguracijske lastnosti

V Iskraemeco d.d. smo se odločili, da bomo za konfiguracijo uporabili XML datoteke. Vsi konfiguracijski atributi so shranjeni v tej datoteki. Ko servis opravlja svoja opravila, v konfiguracijski datoteki pridobi vse potrebne podatke za pravilno delovanje. Poleg tega je datoteka pregledno strukturirana in čitljiva tudi za uporabnika. Vsak servis ima svojo konfiguracijsko datoteko, ki se nahaja v mapi, kjer je servis nameščen. Do sedaj je bilo te datoteke potrebno nastavljati ročno, torej jih je moral uporabnik odpreti v urejevalniku XML datotek ali preprosto v Beležki, tam popraviti želene parametre in jih shraniti. Prav tako pa je bilo potrebno servis predhodno ustaviti in ga kasneje, ko je bila konfiguracija spremenjena, ponovno zagnati. V nasprotnem primeru se konfiguracija ni spremenila.

Aplikacija, ki je obenem sestavni del moje diplomske naloge, pa ves postopek poenostavi, uporabniku pa prihrani kar nekaj tipkanja in rokovanja z dokumenti. V nadaljevanju sledi vsebina konfiguracijske datoteke. V njej lahko vidimo, kako so posamezni parametri ločeni v sekcije, vsaka sekcija pa je namenjena določenim nastavitvam, kot primer sem eno vejo (Databases) pustil razširjeno tako, da so vidni vsi parametri.

```
<?xml version="1.0" encoding="utf-8" ?>
- <configuration>
- <configSections>
  <section name="sep2wProtectedSettings" type="Iskraemeco.Core.Config.Sep2WSection, SEP2Core, Version=2.0.0.0,
      Culture=neutral, PublicKeyToken=47efd62a7df96795" allowLocation="true" allowDefinition="Everywhere"
      allowExeDefinition="MachineToApplication" overrideModeDefault="Allow" restartOnExternalChanges="true"
      requirePermission="true" />
  <section name="sep2wSettings" type="Iskraemeco.Core.Config.Sep2WSection, SEP2Core, Version=2.0.0.0,
      Culture=neutral, PublicKeyToken=47efd62a7df96795" />
    </configSections>
+ <sep2wProtectedSettings configProtectionProvider="RsaProtectedConfigurationProvider">
- <sep2wSettings>
- <root name="root">
- <keys>
```

```
  <clearKeys />
- <addKey key="ServiceLinkDate" type="Iskraemeco.Core.SystemDateTime, SEP2Core">
  <SystemDateTime>2008-11-25T01:50:26Z</SystemDateTime>
      </addKey>
      </keys>
- <nodes>
  <clearNode />
        - <addNode name="Databases">
        - <keys>
          <clearKeys />
        - <addKey key="TimeOut" type="System.Int32, mscorlib">
          <int>5000</int>
              </addKey>
        - <addKey key="ConnectionPoolSize" type="System.Int32, mscorlib">
          <int>10</int>
              </addKey>
        - <addKey key="SqlScriptPath" type="System.String, mscorlib">
          <string>\SQLScripts\</string>
              </addKey>
              </keys>
        - <nodes>
          <clearNode />
        - <addNode name="Counters">
        - <keys>
          <clearKeys />
        - <addKey key="Enabled" type="System.Boolean, mscorlib">
          <boolean>true</boolean>
              </addKey>
        - <addKey key="Category" type="System.String, mscorlib">
          <string>SEP2 Database Counters</string>
              </addKey>
        - <addKey key="Instance" type="System.String, mscorlib">
          <string>Instance{0}</string>
              </addKey>
              </keys>
        - <nodes>
          <clearNode />
              </nodes>
              </addNode>
              </nodes>
              </addNode>
        + <addNode name="Security">
        + <addNode name="EventLog">
        + <addNode name="Database">
        + <addNode name="Remoting">
        + <addNode name="Reading">
              </nodes>
              </root>
              </sep2wSettings>
              </configuration>
```

Kot lahko vidimo, je datoteka dokaj preprosta. Razdeljena je na dva dela in sicer na kriptirane ter nekriptirane parametre. Kot vidimo, imamo za parametre podano ime, tip in vrednost, poleg tega pa so postavljeni v eno sekcijo.

3.2 Servisi in njihovi konfiguracijski parametri

3.2.1 SEP2 Core Service

Core Service (CS) je servis, na katerega se priklopijo vsi SEP-ovi servisi. CS ima podatkovno bazo, v kateri se hranijo licence. Ko se določen servis priklopi na CS, le-ta na podlagi njegovega imena poišče ustrezno licenco, ki mu jo dodeli, če licence ni, pa prijava »pade«. CS ima kar nekaj konfiguracijskih parametrov, ki so – drug za drugim – opisani v spodnji tabeli, slednji pa se nahajajo tudi pri vseh ostalih servisih. Za lažjo preglednost jih bom razdelil v spodaj naštete sekcije:

- Event log (je ena izmed najbolj kompleksnih sekcij, zato jo bom razdelil na podsekcije ter vsako posebej opisal in razložil),
- Remoting,
- Secutity in
- Database.

Event log sekcija
V prvi podsekciji se nahajajo osnovne nastavitve, ki so vedno prisotne in nam zagotavljajo, da logiranje ne bi »ubilo« samega sebe oziroma ne bi imeli prevelike porabe pomnilnika in se nam servisi ne bi ustavili zaradi trenutne nezmožnosti zapisovanja dogodkov.

StartUpLogLevel	*S tem ključem nastavimo dogodke, ki se bodo beležili. Samo dogodki z vrednostjo ključa LogLevel, ki je manjša od vrednosti StartUpLogLevel, bodo beleženi, vsi ostali pa bodo spregledani. Ta parameter je pomemben samo pri zagonu, ko še nimamo podatkov o podatkovnih bazah, o CS in ostalih podatkih. V XML dokumentu je to 32-bitno število.*
SoftPunishmentEnabled	*Ta ključ nam omogoča delovanje varnostnih parametrov oziroma ustavljanje in upočasnjevanje logiranja dogodkov. V XML dokumentu je to boolean spremenljivka.*

SoftPunishmentLimit	*SoftPunishmentLimit je namenjem nastavitvi meje, pri kateri začnemo upočasnjevati prihod novih dogodkov za logiranje. To naredimo tako, da niti, ki to pošilja, postavimo »sleep« komando, tako jo počasi zaustavljamo. V XML dokumentu je to 32-bitno število.*
HardPunishmentLimit	*Dani ključ nam omogoča določitev meje, kjer bomo nit, ki pošilja nove dogodke za logiranje, popolnoma ustavili. Ko bo meja ponovno padla pod SoftPunishmentLimit, se bo niti dovolilo nadaljevati z delom. Vrednost bo lahko padala, ko se bodo dogodki, ki čakajo v vrsti, vpisali. V XML dokumentu je to 32-bitno število.*
MaxTracesPending	*Zadnji ključ prve sekcije pa je uporaben pri nastavitvi maksimalnega števila sledenj, ki se lahko naberejo v pomnilniku. Ko to število prekoračimo, se sledenje ustavi. Kljub temu, da je servis izgubil sledenje, nemoteno nadaljuje s svojim delom. To je namenjeno predvsem temu, da ne bi prišlo do zasedanja celotnega razpoložljivega pomnilnika. V XML dokumentu je to prav tako 32-bitno število.*

V drugi podsekciji Event loga določimo tip beleženja dogodkov, ki jih bomo uporabljali; pri tem ni nujno, da uporabljamo samo enega, naredimo jih lahko poljubno mnogo, vendar pa moramo vsakemu posebej določiti tip in ustrezne parametre, ki so opisani v spodnji preglednici.

Type	*Type je ključ, s katerim določimo tip Event loga oziroma mesto beleženja dogodkov. V SEP-u imamo podprte tri tipe in sicer:* • *FileEventLogWriter,* • *DatabaseEventLogWriter,* • *WindowsEventLogWriter.*
LogLevel	*LogLevel določa dogodke, ki se bodo beležili. Vsak dogodek ima določeno pomembnost, ki je definirana s številkami od 1 do 30. Če bo vrednost, ki jo ima dogodek, manjša kot je LogLevel, se bo dogodek zapisal, v nasprotnem primeru pa ne. V XML dokumentu to zasledimo kot 32-bitno število.*

FileEventLogWriter zajema sledeče ključe:

FileName	*Ta ključ določa ime dokumenta, saj ko se bo »Log« ustvaril, bo dobil to ime. V XML dokumentu je to niz znakov.*
WriteDescription	*S tem ključem določimo, ali bomo v dokumentu zapisali tudi opise dogodkov. Če je vrednost spremenljivke »true«, bomo opise zapisali v dokument, v nasprotnem primeru pa ne. V XML dokumentu je to boolean spremenljivka.*
MaxSize	*Že samo ime tega ključa nam nakaže, da MaxSize določa največjo dovoljeno velikost dokumenta, izraženo v bajtih. Ko je velikost prekoračena, se le-ta »odreže« na MinSize, vendar le na začetku, kar nam omogoča, da so novejši dogodki shranjeni, starejši pa se izgubijo. V XML dokumentu je to 64-bitno število.*
MinSize	*Z njim lahko določimo minimalno velikost dokumenta, vendar ne med ustvarjanjem dokumenta, saj je tedaj njegova vsebina*

	prazna, temveč ko pride do t.i. prekoračitve maksimalne vrednosti. Takrat se bo dokument odrezal na minimalno velikost. V XML dokumentu je tudi to 64-bitno število.

Pri WindowsEventLogWriter imamo sledeče ključe:

LogName	*Z danim ključem določimo ime, pod katerim se bodo dogodki shranjevali. To je glavna sekcija, v katero lahko zapisujemo več različnih »logov«. Kot primer naj podam Application Event Log, v katerega se običajno zapisujejo dogodki vseh servisov ob zagonu, šele kasneje prevzamejo lastnosti, ki so zapisane v konfiguracijski datoteki. V XML dokumentu je to niz znakov.*
Source	*Source je prav tako namenjen določanju imen dogodkov, tako da jih bomo v glavni sekciji lahko lažje iskali. Kot primer naj podam eno možno nastavitev: LogName je Application Event Log, source pa SEP2 Meter Access Service. To pomeni, da bomo imeli v Application Event Log še druge dogodke, ki jih naš servis ne beleži, vendar bo v tem primeru ločevanje določeno s Source ključem. V XML dokumentu je to niz znakov.*
Computer	*V tem ključu navedemo ime računalnika, na katerega se bomo povezali (. pomeni lokalni računalnik). V XML dokumentu ga določa niz znakov.*
Create	*S tem ključem določimo, ali se bo »Log« ustvaril v primeru, da ne obstaja. Torej, če bo vrednost ključa »true«, se bo »Log«, če še ne obstaja, ustvaril. V XML dokumentu je to boolean spremenljivka.*

DatabaseEventLogWriter vsebuje naslednje ključe:

DataSourceName	*Z omenjenim ključem določimo ime podatkovne baze, v katero bomo pisali prihajajoče dogodke. V XML dokumentu je to niz znakov.*

Remoting sekcija

V njej najdemo spodnje parametre:

Protocol	*S tem ključem definiramo protokol, ki se bo uporabljal za .NET Remoting. V XML dokumentu je to Iskraemeco.Core.Service.ChannelProtocol.*
Formatter	*Ključ določa format, ki se bo uporabljal z .NET Remoting. V XML dokumentu je to Iskraemeco.Core.Service.ChannelFormatter.*
Port	*Dani parameter določa številko porta, na katerem bo servis poslušal in čakal na zahtevo. Že pri namestitvi se nastavijo privzete vrednosti, ki pa so odvisne od servisa (8090, 8091, 8094, itd.). V XML dokumentu je to 32-bitno število.*
EnsureSecurity	*S tem ključem odločimo, ali želimo imeti varen kanal ali je le-ta trenutno nepotreben. V XML dokumentu je to boolean spremenljivka.*

Security sekcija

Tretja sekcija se ponaša s sledečimi parametri:

Url	*Z Url določimo naslov, kjer se nahaja CS. V praksi vpišemo IP številko računalnika, na katerem se izvaja CS. V XML dokumentu je to niz znakov.*
UseDefaultCredentials	*S tem parametrom določimo, ali bomo za*

	prijavljanje na CS uporabljali kar podatke trenutno prijavljenega uporabnika ali pa bomo vpisali druge podatke. Če je ključ nastavljen na vrednost »true«, potem uporabljamo trenutno prijavljenega uporabnika na Windows sistemu. V XML dokumentu je to boolean spremenljivka.
Username	*Username določa uporabniško ime, s katerim se bomo prijavljali na CS, le-to pa mora obstajati v bazi, v nasprotnem primeru avtentikacija pade. Ključ je aktiven samo, če je vrednost UseDefaultCredentials »false«. V XML dokumentu je to niz znakov.*
Domain	*Z navedenim ključem nastavimo ime domene. Ključ je aktiven samo, če je vrednost parametra UseDefaultCredentials false. V XML dokumentu je to niz znakov.*
password *(optional!)*	*Že sam izraz Password nam pove funkcionalnost danega parametra. Z njim nastavimo geslo, s katerim se prijavljamo, seveda pa mora biti to geslo povezano z uporabniškim imenom, ki je zapisano v bazi in ga uporabljamo. Ključ je aktiven samo, če je UseDefaultCredentials false. V XML dokumentu je to niz znakov.*

Database sekcija

Sledijo parametri:

TimeOut	*TimeOut določa čas za pridobivanje resursov, kot je na primer povezovanje. V XML dokumentu je to 32-bitno število.*
ConnectionPullSize	*Ta ključ je namenjen določitvi maksimalnega števila sočasnih povezav na bazo. V XML dokumentu je to 32-bitno število.*

SqlScriptPath	*S slednjim parametrom določimo lokacijo, kjer so nameščene SQL skripte. V XML dokumentu je to niz znakov.*
Enabled	*Enabled je ključ, s katerim določimo, ali so performančni kazalniki omogočeni ali ne. V XML dokumentu je to boolean spremenljivka.*
Category	*Category definira kategorijo performančnih kazalnikov.*
Instance	*Zadnji parameter Database sekcije je namenjen določitvi instance performančnega kazalnika.*

3.2.2 SEP2 MeterAccess Service

MAS nam omogoča dostop do merilnih točk, zato mora biti sposoben komunicirati z njimi. Omogočati mora transparentnost, zato je razdeljen na dva sloja; nižjega in višjega. Višji sloj skrbi za komunikacijo s števci, komunikatorji in koncentratorji, nižji sloj (Connection Server) pa skrbi za vzpostavitev fizične povezave. Namen slednjega je predvsem v tem, da različne komunikacijske tipe naredi transparentne za višji sloj. Tu pa se nam hitro ponudi velika količina konfiguracijskih parametrov. MAS omogoča sledeče:

ExecuteTimeOut	*S tem ključem nastavimo največji dovoljeni čas izvajanja posamezne zahteve. Če je le-ta presežen, MAS samodejno zaključi zahtevo. V XML dokumentu je to 32-bitno število.*
MaxRequestsInExecution	*Z naslednjim parametrom nastavljamo največje dovoljeno število zahtev, ki se sočasno vršijo. Če je vrednost prekoračena, nam MAS odgovori: ToManyRequestsInExecution. V XML dokumentu je to 32-bitno število.*
ConnectTimeOut	*ConnectTimeOut nastavi čas, ki ga MAS pusti gonilniku, da uspešno vzpostavi povezavo z napravo. Če v okviru tega časa povezava ni vzpostavljena, se zahteva ne izvrši. V XML dokumentu je to 32-bitno število.*

OptimizeOperationsTimeOut	*Z danim ključem predpišemo čas, ki ga MAS nameni izvajanju optimizacij operacije. Če je čas prekoračen, se optimizacija ne izvede. V XML dokumentu je to 32-bitno število.*
ExecuteOperationTimeOut	*Ta parameter nam služi, da nastavimo čas, v katerem MAS še dovoli, da je posamezna operacija (npr. branje registra) izvršena. V XML dokumentu je to 32-bitno število.*
CorrectAbortTimeOut	*S sledečim ključem nastavimo čas, ki ga MAS da na razpolago, da se operacije pravilno zaključijo, potem ko je bil poklican »abort«. Če se v tem času ne zaključijo, so »ubite«. V XML dokumentu je to 32-bitno število.*
BeforeConnectTimeOut	*Z zadnjim ključem te sekcije nastavimo čas, ki ga MAS nameni izvajanju podoperacij posamezne operacije. V XML dokumentu je to 32-bitno število.*

3.2.3 SEP2 MeterReading Service

Izvajanje zajema podatke iz merilnih naprav, definiranih v SEP podatkovni bazi, ki jo opravlja MRS. Storitev omogoča delo z več SEP podatkovnimi bazami in več MAS strežniki, ki dejansko vzpostavljajo fizične povezave in izvedejo zajem podatkov iz naprav v sistemu SEP. Z uporabo SCH storitve je mogoče doseči avtomatizirano branje naprav v sistemu. Osnovna naloga MRS storitve je:

- da z uporabo komunikacijskih in protokolarnih lastnosti generira bralne zahteve za MAS strežnike v skladu s komunikacijskimi potmi,
- pošlje te zahteve enemu ali več MAS strežnikom,
- pridobi odgovore na zahteve od MAS strežnikov in
- ovrednoti odgovore ter shrani pridobljene podatke v SEP podatkovno bazo.

MRS omogoča poleg standardnih nastavitev, tudi nastavljanje vseh internih parametrov, ki so razdeljeni v 4 sekcije:

SchedulingInterface

S to sekcijo določimo obnašanje IScheduling vmesnika.

MaxParallelExecutions	*Ta parameter služi določitvi maksimalnega števila sočasno izvajajočih se opravil. V XML dokumentu je to 32-bitno število.*
WaitQueueLength	*Funkcionalnost danega ključa je v določitvi maksimalnega števila opravil v čakalni vrsti. Ko vrednost prekoračimo, bodo vsa naknadna opravila zavržena. V XML dokumentu je to 32-bitno število.*
DoneQueueLength	*Ta ključ je namenjen temu, da uporabnik določi dolžino čakalne vrste za opravljena opravila. Ko se opravilo zaključi, se podatki shranijo v to čakalno vrsto. Le-ta pa se obnaša kot FIFO pomnilnik. V XML dokumentu je to 32-bitno število.*

Statistics

Sekcija določa, ali bo MAS statistika ohranjena ali ne.

UpdatePhysicalConnectionStatistics	*Namembnost tega ključa se kaže v tem, da z njim določimo, ali bomo statistiko fizične povezave posodobili ali ne. Torej, v bazi hranimo podatke, o aktivnost, ki so se dogajale na povezavi. Če je vrednost »true«, jo posodobimo in v bazo vpišemo zadnje stanje. V XML dokumentu je boolean spremenljivka.*
UpdateLogicalConnectionStatistics	*Ta ključ določa, ali bomo statistiko logične povezave posodobili ali ne. V XML dokumentu je boolean spremenljivka.*
UpdateMeasurementPointResultTypeStatistics	*S sledečim parametrom določimo, ali bomo statistiko prebranih rezultatov in statistiko manjkajočih operacij posodobili ali ne. V XML dokumentu je boolean spremenljivka.*

UpdateEventLogStatistics	*Zadnji segment Statistic sekcije je parameter, s katerim določimo, ali bo statistika dogodkov na merilnih napravah posodobljena ali ne. V XML dokumentu je boolean spremenljivka.*

Optimizations

Ta sekcija določa, kdaj je smotrno uporabiti optimizacijo operacij.

DeviceOptimizationLimit	*S tem ključem določimo število, pri katerem ne bomo več brali posamezne naprave, ampak drevesno strukturo naprav. S tem pohitrimo zadevo, kajti če želimo napravo brati, jo moremo najprej poiskati, kar je potratna operacija. Predstavljate si namreč, da imamo 1.000.000 naprav. V XML dokumentu je to 32-bitno število.*
MeasurementPointOptimizationLimit	*Ob pomoči danega parametra določimo mejo, pri kateri bomo uporabili branje drevesne strukture merilnih točk. V XML dokumentu je to 32-bitno število.*
MeasurementPointResultTypeOptimizationLimit	*Ta ključ je namenjen temu, da določimo mejo, pri kateri uporabimo drevesno strukturo za branje. V XML dokumentu je to 32-bitno število.*

ReadingExecutor

Ta sekcija določa obnašanje MRS bralne logike. Naslednji parametri bi morali biti vedno sinhronizirani med MRS servisi.

RequestSaveFolder	*Ta ključ nam koristi pri določitvi lokacije, v katero bo MRS shranjeval zahteve. V XML dokumentu je to niz znakov.*
ResponseSaveFolder	*Z naslednjim parametrom določimo lokacijo,*

	na katero bo MRS shranjeval odgovore na zahteve. V XML dokumentu je to niz znakov.
MaxRepetitions	*MaxRepetitions določa, kolikokrat se bo določena zahteva ponovila v primeru nepričakovanih usodnih napak (MAS response loss, MAS restart, itd.). V XML dokumentu je to 32-bitno število.*
SendRequestPoolTime	*S tem ključem povemo časovni interval, v katerem bo MRS poskušal začeti zahtevo na MAS strežniku. V XML dokumentu je to 32-bitno število.*
BlockedRequestSleepTime	*Ta ključ postane aktualen, ko MRS ne uspe pridobiti zaklepanja branja ali povezave z MAS strežnikom. Zahteva bo označena kot blokirana in bo počakala toliko sekund, kot je določeno v samem ključu. V XML dokumentu je to 32-bitno število.*
MaxSendRequestTrials	*Danega parametra se poslužimo, ko določimo število možnih ponovitev trenutno blokirane zahteve. V XML dokumentu je to 32-bitno število.*
ReadingLockUpdatePeriod	*S tem parametrom nastavimo čas, po katerem bo zaklepanje za branje osveženo. V XML dokumentu je to 32-bitno število.*
ReadingLockExpireTime	*Z naslednjim ključem nastavimo čas, po katerem bo zaklepanje za branje poteklo, če ni bilo predhodno osveženo. V XML dokumentu je to 32-bitno število.*
MeterAccessServiceBlockDuration	*S tem ključem določimo, koliko časa bo MAS označen kot blokiran. MRS označi MAS kot blokiranega, če le-ta ni dosegljiv. V XML dokumentu je to 32-bitno število.*
MaxShutdownTime	*Z MaxShutdownTime določimo maksimalni čas, ki ga bo ReadingExecutor čakal, preden se bo zaključil. V XML dokumentu je to 32-bitno število.*

3.2.4 SEP2 Scheduler Service

SCH storitev združuje napreden razporejevalnik opravil in upravljalnik delovnih tokov, kar omogoča izvajanje kompleksnih nalog v sistemu SEP. Posamezne naloge so lahko izvedene avtomatsko in/ali ročno. Razporejevalnik opravil zagotavlja, da se bodo avtomatična izvajanja nalog začela ob predvidenem času. Za vsako nalogo je poleg začetnega časa izvajanja podan tudi izvajalni interval, za katerega se bo naloga izvršila (npr. če je naloga branje podatkov, potem časovni interval definira časovno območje, za katerega se morajo pridobiti podatki o porabi iz merilnih naprav). Vsaka naloga je definirana tako, da je neodvisna od objektov, za katere naj se izvrši (npr. branje naprav je definirano tako, da je neodvisno od tipa merilnih naprav, za katere se bo branje izvedlo; to pomeni, da se lahko ista definicija naloge večkrat uporabi za različne merilne naprave). Če je namesto enega objekta (naprave) definirana množica objektov (naprav) se naloga izvede za to množico objektov.

SCH omogoča poleg standardnih nastavitev še nastavitve, ki so zanj specifične in opisujejo njegovo obnašanje:

ScheduledJobUpdatePeriod	*Dani ključ je namenjen temu, da z njim določimo, kako pogosto bo SCH v bazi preveril, ali čakajo nove naloge in na koliko časa bo posodobil stanje izvajajočih. V XML dokumentu je to 32-bitno število.*
ScheduledJobLockExpireTime	*S sledečim parametrom določimo, kako pogosto bomo osveževali zaklepanje, kajti zaklepanje se, če ni osveženo po določenem času, ukine. V XML dokumentu je to 32-bitno število.*
DoneTaskInfoLifeTime	*Dani ključ nam omogoča, da z njim določimo, koliko časa bo SCH hranil podatke o opravljeni nalogi. V XML dokumentu je to 32-bitno število.*
DefaultTaskPriority	*Namen DefaultTaskPriority je, da z njim določimo privzeto prioriteto nalog. Če ni drugače določeno, bodo opravila imela to prioriteto, sicer pa jim lahko dodelimo tudi drugo prioriteto. V XML dokumentu je to 32-bitno število.*

MaxExecutingJobThreads	*S tem ključem določimo število nalog, ki se lahko sočasno izvajajo. V XML dokumentu je to 32-bitno število.*
MaxLoadBallancerThreads	*Naslednji ključ je po funkcionalnosti podoben prejšnjemu, vendar z razliko, da le-ta določa število nalog, ki jih lahko sočasno zaženemo. V XML dokumentu je to 32-bitno število.*
CheckTaskStatusPeriod	*Sledeči parameter določa, kako pogosto bo SCH preverjal stanje izvajajočih se nalog. V XML dokumentu je to 32-bitno število.*
ExpandGroupLevel	*ExpandGroupLevel je ključ, s katerim določimo, kako globoko bomo šli po grupah. Če bomo vnesli vrednost 0, pomeni, da bo uporabljena samo izbrana grupa, če pa bo vneseno število 1, bomo vzeli še njene »sinove«. V XML dokumentu je to 32-bitno število.*
ExpandDeviceLevel	*Z naslednjim ključem določimo, kako globoko bomo šli po napravah. Princip je enak, kot sem že zapisal pri zgornjem parametru. V XML dokumentu je to 32-bitno število.*
TaskSleepBeforeRetryTime	*Ta ključ nam služi, da lahko z njim določimo, koliko časa bo minilo med ponovitvami dveh opravil. V XML dokumentu je to 32-bitno število.*
TaskSleepBeforeRetryTimeScaler	*Zadnji parameter določa, kako se bo stopnjeval čas ponovitev med dvema opraviloma. Naj navedem primer: če je »TaskSleepBeforeRetryTimeScaler« = 2 in je »TaskSleepBeforeRetryTime« = 5, potem bo čas med prvim in drugim opravilom 5 enot, med drugim in tretjim 10 enot ... V XML dokumentu je to 32-bitno število.*

3.2.5 SEP2 Alarm Service

AS je servis, ki omogoča prejemanje dogodkov in asinhrono branje podatkov. Alarm je dogodek, ki ga pošlje merilna naprava v naš sistem. Dolžnost AS je, da se odziva na alarme, ki pa jih ne prejme neposredno od merilne naprave, ampak jih dobi od LS. Za odzivanje smatramo vpisovanje alarmov v bazo, pošiljanje preprostega SMS sporočila itd. Servis se obnaša kot strežnik. Preprosto čaka na zahtevo, ki jo dobi v obliki alarma in jo potem izvrši. Poleg standardnih parametrov ima AS nastavljiv samo še en parameter in sicer:

DSN	*Ime podatkovne baze, s katero servis dela. V XML dokumentu je to niz znakov.*

3.2.6 SEP2 Listener Service

LS je servis, ki sprejema klice (alarme) z merilnih naprav, jih obdela in pošlje naprej AS. LS naredi sporočila neodvisna od naprave, tako da AS ni potrebno vedeti, kakšna naprava jih je poslala.

3.2.7 SEP2 Report Service

RS omogoča izvajanje poročil shranjenih v XML formatu, SEP sistemu ali datotekah. Uporablja SEP2 Database-ove in SEP2 Core-ove knjižnice za povezovanje in komunikacijo z bazo ter za vpisovanje dogodkov. Za komunikacijo s klienti RS uporablja Microsoft Remoting. Več klientov lahko sočasno uporablja RS oziroma mu pošilja zahteve. Zahteve in odgovori so posredovani v obliki XML datoteke. Vsaka zahteva ima tri osnovne gradnike:

- seznam poročil, ki bodo izvedena,
- lokacijo, kjer bodo izvedena poročila shranjena ter
- seznam vseh merilnih naprav, nad katerimi se zahteva izvajanje poročila.

RS lahko izvaja različne tipe poročil, vendar je vse to odvisno od gonilnika. Trenutno RS podpira sledeče tipe:

- ReportColumns,
- ReportSummaryGrid in
- ReportAttributes.

RS poleg običajnih konfiguracijskih nastavitev ne vsebuje nobenih specifičnih, ki bi veljale samo zanj.

3.2.8 SEP2 Prepayment Service

PS je namenjen predplačniškemu načinu kupovanja električne energije. Sprejema klice od PP, IS in POS ter drugih servisov ali aplikacij. Ob zagonu najprej vzpostavi povezavo s CS, da preveri veljavnost licenc, temu sledi povezovanje z EWS, kjer pridobi kupljeno kodo. Ko je koda pridobljena, lahko nadaljuje z delom. PS poleg običajnih nastavitev omogoča še:

ECWS URL	*ECWS URL določa URL naslov, na katerem se nahaja ECWS. V XML dokumentu je to niz znakov.*
Scheduler URL	*Naslednji ključ prav tako določa URL naslov, vendar se le-ta usmeri na tisti naslov, na katerem se nahaja SCH. Če naslova ni, potem se zahteva razpošlje na vse aktivne SCH servise. V XML dokumentu je to niz znakov.*
MakePurchase Job URL	*Dani parameter nam omogoča, da z njim določimo URL naslov na XML dokument, v katerem se nahaja MakePurchase definicija. V XML dokumentu je to niz znakov.*
GetCredit Job URL	*S tem parametrom določimo URL naslov na XML dokument, v katerem se nahaja GetCredit definicija. V XML dokumentu je to niz znakov.*
ResultType	*ResultType je ključ, s katerim določimo tipe rezultatov. V XML dokumentu je to niz znakov.*

3.2.9 SEP2 Validation Service

VS uporablja merilne naprave in validacijske dokumente, ki so shranjeni v podatkovni bazi, opravi preverjanje podatkov oziroma validacijo in jo shrani v bazo.
Validacija poteka v sledečih korakih:

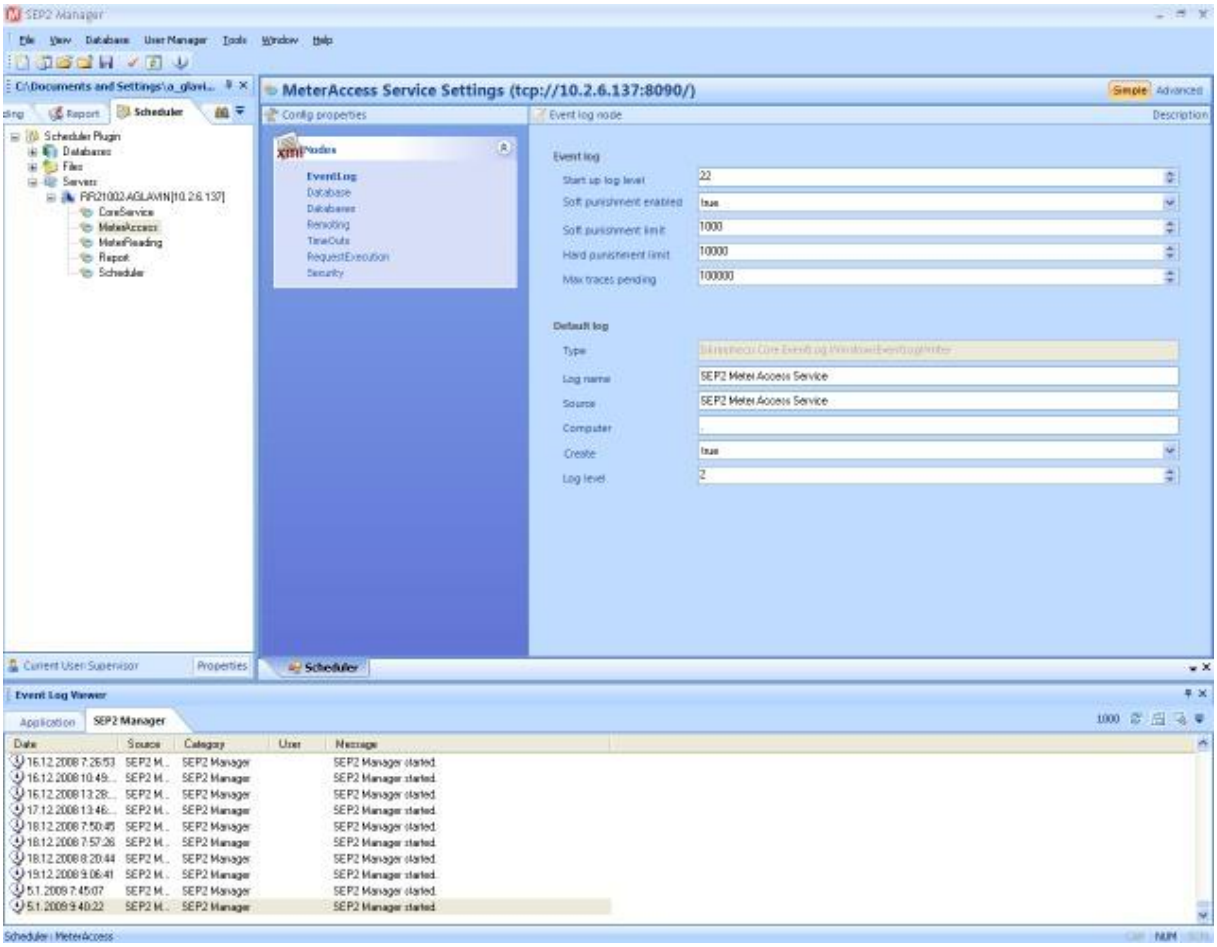
- VS postavi zahtevo v čakalno vrsto,
- izbere se prva zahteva v čakalni vrsti,
- poišče se ustrezen validacijski dokument in

- na podlagi zahtevanih parametrov ter validacijskega dokumenta se izvede validacija.

Poleg standardnih konfiguracijskih parametrov nima VS nobenih specifičnih.

3.3 Grafični vmesnik in njegova izvedba

Kot sem že povedal, je grafični vmesnik namenjen lažjemu obdelovanju konfiguracijskih datotek. Tukaj mislim predvsem na daljinski dostop do datotek in lažjo čitljivost parametrov. Ni nujno, da za obdelovanje in spreminjanje samega dokumenta poznamo XML zapis, saj za to poskrbi program, mi pa mu moramo samo vnesti ustrezne parametre.
Vmesnik (glej sliko 6) ni samostojen program, ampak je del SEP2W Manager-ja. Trenutno se nahaja v SEP2 Scheduler pluginu, v bodoče pa ga bomo premestili v nov plugin, ki bo imel primernejše ime. Kje se bo vmesnik nahajal, zdaj ni tako pomembno. To je pomembnejše s tržnega vidika, jaz pa se bom zdaj poglobil v samo delovanje vmesnika in njegovo zgradbo.



Slika 6: SEP2 Manager in preprost pogled vmesnika za konfiguracijo.

3.3.1 Zgradba vmesnika

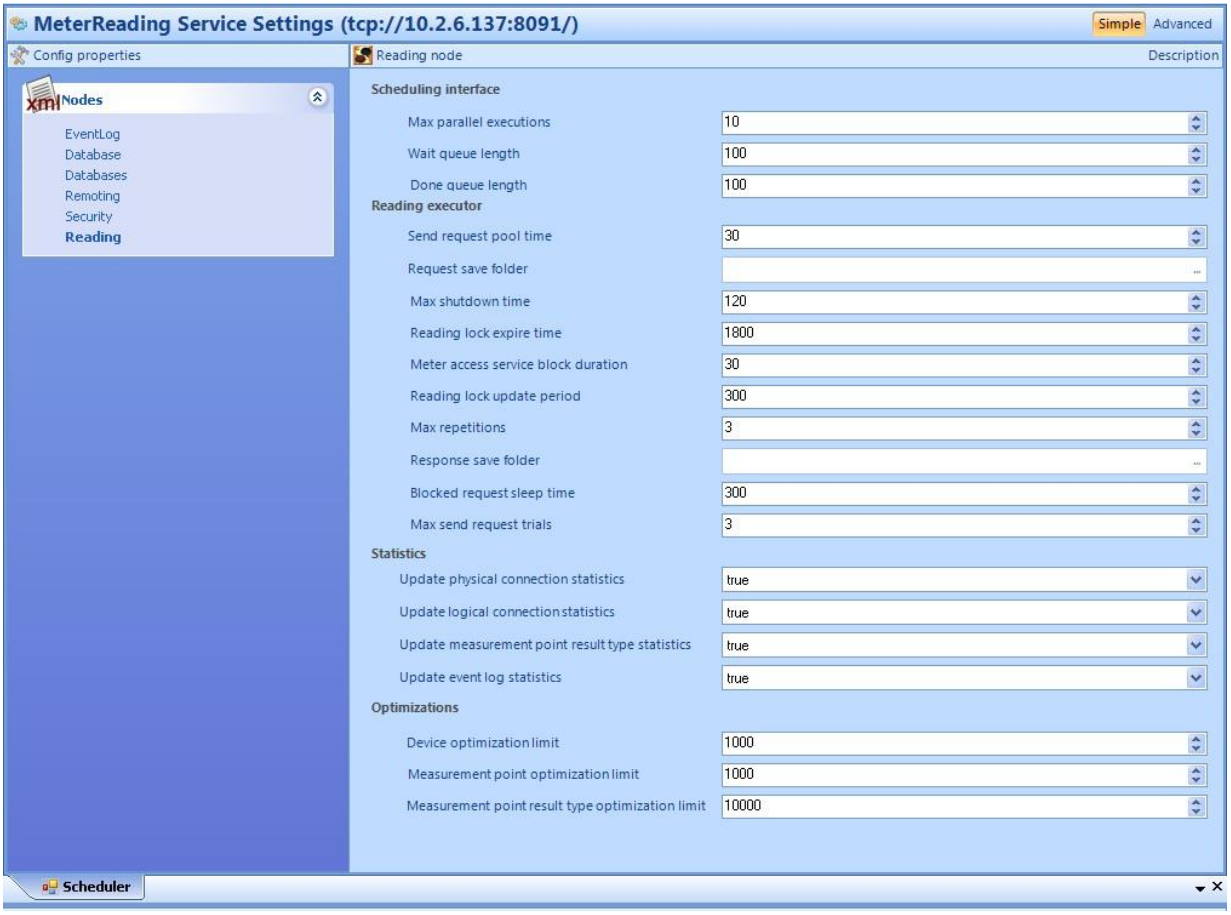
Vmesnik ima dva glavna dela, ki se ločita predvsem po funkcionalnosti. Sestavljen je iz preprostega (Simple view) in naprednega (Advanced view) pogleda. Razlika med njima je samo v funkcionalnosti oziroma v tem, kaj nam kateri omogoča:

- Preprosti pogled je namenjen manj veščim uporabnikom. Tu lahko spreminjamo samo predhodno določene parametre, ne moremo pa dodajati novih ali spreminjati njihovih imen. Torej spremenimo lahko le vrednost že obstoječega ključa. Poleg tega pa v tem pogledu niso zajeti vsi ključi, ampak samo najpogostejši.
- V naprednem pogledu pa je, ravno obratno, dovoljeno skoraj vse, zato mora uporabnik dobro vedeti, kaj želi spreminjati, dodajati, preimenovati, brisati, itd., kajti v nasprotnem primeru lahko parametre tudi »pokvari« in delovanje servisa spremeni tako, da le-ta ne deluje več pravilno oziramo drugače povedano servis deluje, kot smo mu ukazali, kar za nas lahko ni sprejemljivo.

Zgradba je zelo preprosta, če bo kdo na hitro pogledal, bo rekel: *»Ma, tukaj imamo nametanih malo label, malo tekstovnih polj, malo numeričnih polj, malo polj za izbiranje in to bo to.«* Pri naprednem pogledu pa bi se izjava glasila nekako takole: *»Imamo pogled na drevesno strukturo in en list view.«* Pripoved ne laže, natanko tako je, saj nam takšna zgradba omogoča najpreprostejše delo in največjo preglednost. Če se bomo poglobili v delo, bomo hitro opazili, da imamo v ozadju kar nekaj, na prvi pogled skritih stvari, osnova pa ostaja preprostost in preglednost.

3.3.1.1 Preprosti pogled

Kot sem že zgoraj zapisal, nam preprosti pogled omogoča spreminjanje samo predhodno definiranih parametrov. Zgrajen je tako, da ima za vsako sekcijo (Event Log, Database, Security, Time Outs, itd.) svojo kontrolo, ki je prikazana, ko neko sekcijo izberemo. Vsaka kontrola predstavlja svoj razred. Načeloma bi lahko napisali vse v eno kontrolo, vendar tako postane sama koda nepregledna, imena niso več reprezentativna, ker lahko eno polje uporabljamo za prikazovanje različnih ključev. Načelo, ki se ga moremo držati, pa je, da mora biti koda pregledna in berljiva. Polja so zasnovana tako, da ne dovolijo napak. Torej, če bi nekdo za časovno periodo vpisal niz, ga bi program opozoril in mu preprečil shranjevanje take vrednosti. Enako velja za vsa polja v tem pogledu. Uporabnik praktično drastične napake niti ne more narediti. Spodaj (glej sliko 7) podajam nekaj kontrol preprostega pogleda.



Slika 7: Primer kontrole za urejanje vrednosti.

Pri vseh kontrolah je uporabljen enak princip dela. Lahko bi rekel, da so po arhitekturi enake, razlikujejo se samo po podatkih, ki jih prikazujejo. Podrobneje si poglejmo glavni del kode:

```
public partial class RequestControl : UserControl
{
   bool change;
   bool makerequestexecution = true;
   ConfigNode node;
   ConfigNode requestNode;

   public event EventHandler Changed;

   public void OnChange()
   {
      if (Changed != null)
      {
         Changed(this, EventArgs.Empty);
      }
   }

   public RequestControl()
   {
      InitializeComponent();
   }
   public void SetMainNode(ConfigNode nodes)
   {
      node = nodes;
   }
```

```
    public void LoadData()
    {

        requestNode = node.FindSubNode(new ConfigPath("RequestExecution"));
        if (!requestNode.Name.Equals("<null node>"))
        {

                        MaxRequestsInExecutionud.Text = requestNode["MaxRequestsInExecution"].ToString();

        }
        else
        {
            Sep2MessageBox.ShowInfo(this, "This node does not exist!");
        }

    }

    public void SaveData()
    {
        try
        {
                        requestNode["MaxRequestsInExecution"] = Convert.ToInt32(MaxRequestsInExecutionud.Text,
                        CultureInfo.InvariantCulture);
        }
        catch (Exception e)
        {
            Sep2MessageBox.ShowInfo(this, e.Message);
        }
    }

    private void MaxRequestsInExecutionud_ValueChanged(object sender, EventArgs e)
    {
        if (change == true)
            OnChange();

        change = true;
    }
}
```

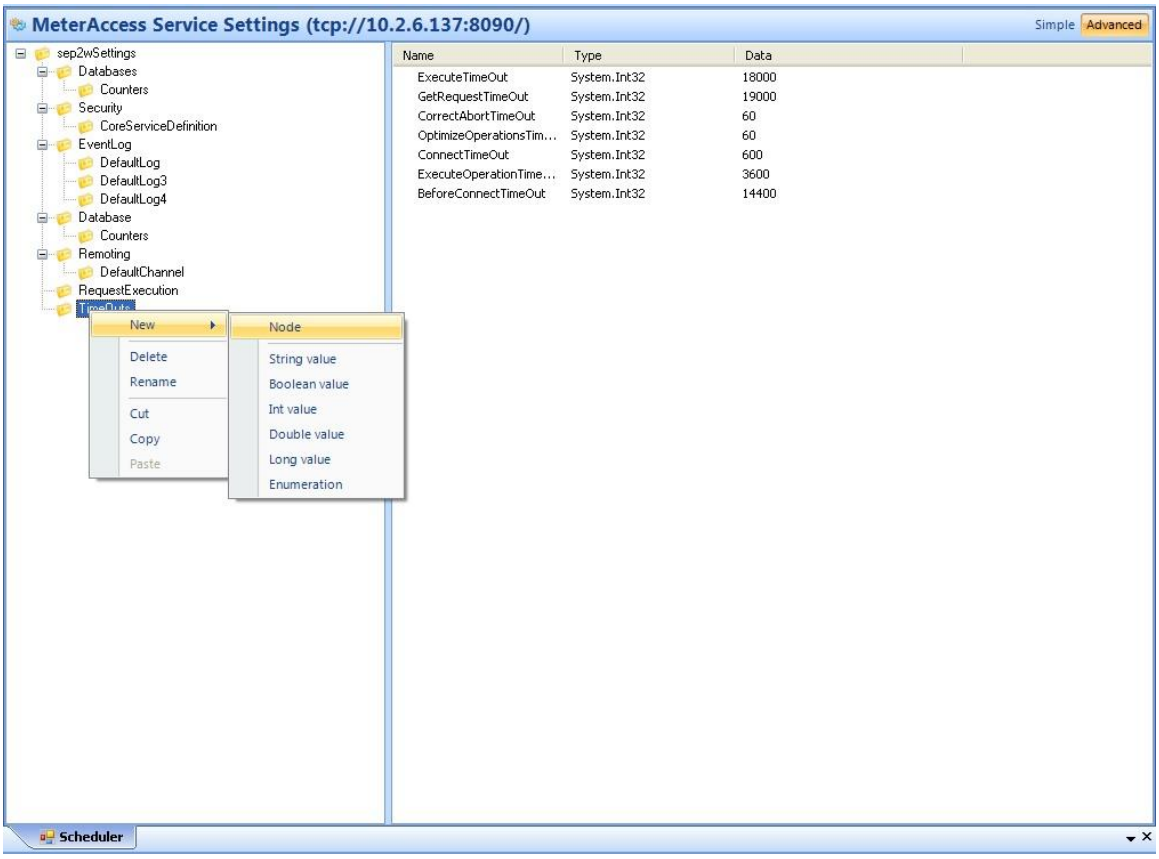
To je koda ene kontrole. Ko bom razložil njeno vsebino, bo hitro jasno, v čem se razlikujejo ostale kontrole. Tu imamo tri metode:

- SetMainNode: s to metodo dobimo potrebne podatke v obliki drevesne strukture, ni pa nam podano celotno drevo konfiguracijske datoteke, ampak samo tista veja, ki jo obdelujemo.
- LoadData: z dobljenega segmenta preberemo želene podatke. Ker gre za drevesno strukturo so le-ti shranjeni kot listi oziroma podveje tega drevesa.
- SaveData: ta metoda je poklicana, ko želimo spremenjeno vrednost shraniti v drevesno strukturo, ki nam vrednost shrani v točno določen list.

Poleg teh treh metod imamo še druge, ki pa so namenjene predvsem temu, da glavna aplikacija (SEP2 Manager) ve, ali so se podatki spremenili in potem vpraša, če jih želimo shraniti oziroma ovreči.

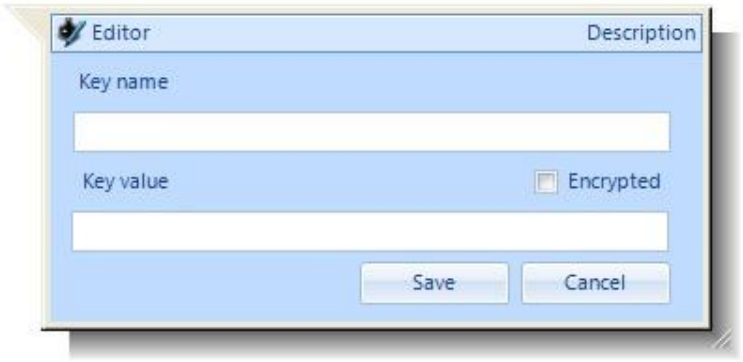
3.3.1.2 Napredni pogled

Ta pogled (glej sliko 8) ima zelo preprosto zgradbo. Sestavljen je iz dveh elementov: drevesne strukture in polja za prikazovanje parametrov.



Slika 8: Napredni pogled.

Ko kliknemo na želeno vejo, se v polju prikažejo njeni parametri, ki jih lahko spreminjamo, brišemo, dodajamo. Dodajamo in brišemo lahko tudi veje, program pa nam omogoča tudi kopiranje posamezne veje in njeno lepljenje na drugo mesto. Če ga primerjam s kakšnim programom, je zelo podoben urejevalniku registra v Windowsu. Za dodajanje parametrov imamo poseben urejevalnik (glej sliko 9), ki nam pomaga pri določanju tipov in nam prepreči vnašanje nesmiselnih vrednosti.



Slika 9: Urejevalnik parametrov, ki je obenem namenjen dodajanju novih in spreminjanju obstoječih parametrov.

Urejevalnik je zasnovan tako, da sam poskrbi za pretvorbo podatka v pravilni podatkovni tip oziroma, ko v meniju izberemo na primer dodajanje niza, nam bo urejevalnik ponudil ustrezna polja, v katera bomo lahko vnašali samo nize ali, če želimo, števila, itd. Za parametre, ki so predhodno določeni, že vnaprej vemo, katere vrednosti lahko zasedejo. Naj bodo te vrednosti ponujene ali ne, ni potrebno, da jih bi vedeli na pamet.
Zdaj pa si poglejmo nekoliko podrobneje zgradbo navedenih izsekov**.** Podal bom najpomembnejše dele kode in na kratko opisal njihovo delovanje.

```
private void Add_key(string type)
{
            ListViewItem item = null;
    int n_items;

            using (SEP2PopupControl popup = new SEP2PopupControl())
            {
     popup.Frozen = true;
     popup.ShowInTaskbar = false;

     using (Editor edit_text = new Editor(type, g_node.Keys))
     {
                        if (SEP2PopupControl.DisplayPopup(popup, position_x, position_y, edit_text, false) == DialogResult.OK)
       {

          if (type.Equals("System.Int32"))
          {
           listView.Items.Add(edit_text.KeyName);
           n_items = listView.Items.Count;
           item = listView.Items[n_items - 1];
           item.SubItems.Add(type);
           item.SubItems.Add(edit_text.Data);
           g_node[edit_text.KeyName] =       Convert.ToInt32(edit_text.Data, CultureInfo.InvariantCulture);
```

V tem izseku vidimo metodo za dodajanje novega parametra. Kaj torej naredimo? Najprej moramo prikazati Popup zgolj zaradi oblike našega glavnega programa Manager, sicer bi lahko urejevalnik prikazali kar kot svoje okno. V urejevalnik vpišemo želene podatke in ko jih potrdimo, se Popup zapre, mi pa moramo podatke vpisati na pravo mesto. Preverimo, katerega tipa je novi parameter. Ko to naredimo, ga moremo najprej vpisat v naš prikazovalnik parametrov, torej v »listView«, nato pa še v samo drevo »g_node«. Novi parameter naredimo tako, da drevesu dodamo nov list. To napravi sledeči stavek:

```
g_node[edit_text.KeyName]=Convert.ToInt32(edit_text.Data, CultureInfo.InvariantCulture);
```

g_node je veja, kateri želimo dodati nov list. V oglatih oklepajih podamo ime lista, na koncu pa mu dodelimo še vrednost in nov parameter je vpisan v drevesno strukturo, to pa ne pomeni, da je vpisan tudi v konfiguracijsko datoteko. Šele, ko shranimo spremembe, se nova drevesna struktura prepiše v konfiguracijsko datoteko.

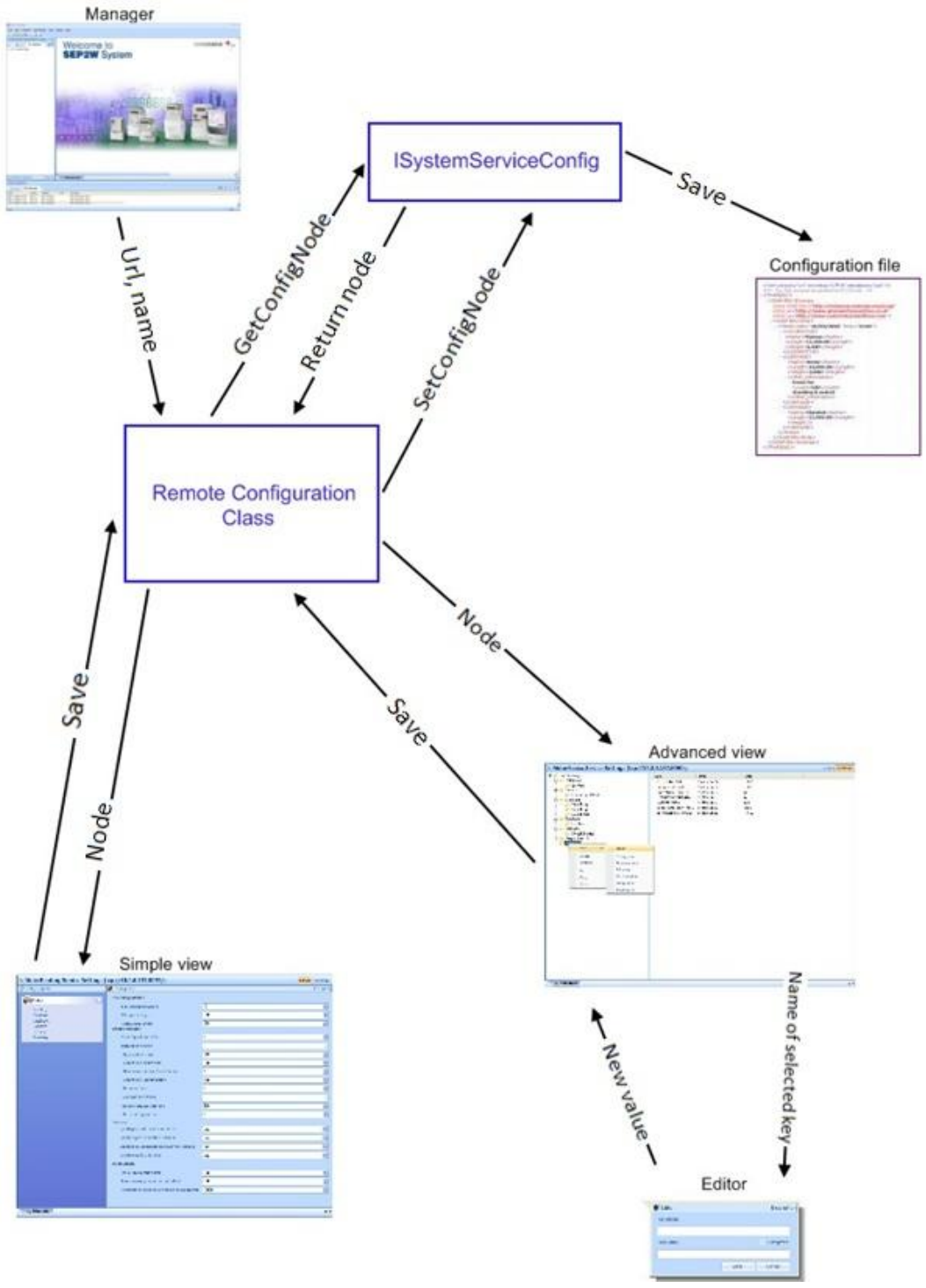
Razred vsebuje še druge metode, ki so namenjene spreminjanju vrednosti ključa, dodajanju nove veje, brisanju veje, brisanju ključa, kopiranju veje, itd. Nesmiselno bi bilo opisovati vsako posebej, kajti pri vseh delamo z drevesom, kjer je princip dela popolnoma enak, razlika

je samo v temu, da pri nekaterih ključe in elemente dodajamo, pri drugih jih brišemo, pri tretjih pa samo preverjamo.

Naj navedem še podatek, ki je pomemben pri kopiranju in lepljenju. Imena vej v drevesu morajo biti unikatna, torej eno ime se ne sme dvakrat pojaviti v isti veji, zato imam še metodo, ki preverja ali veja, ki jo želimo prilepiti, že obstaja. Če je tako, potem njeno ime spremenimo.

3.4 Shema delovanja in kratek opis

Delovanje oddaljene konfiguracije nazorno prikazuje shema (na sliki 10). Mogoče je razbrati povezave in podatke, ki se prenašajo med posameznimi razredi, kljub temu pa bom podal malenkost podrobnejši opis za lažjo predstavo nad celotnim stanjem.



Slika 10: Shema delovanja.

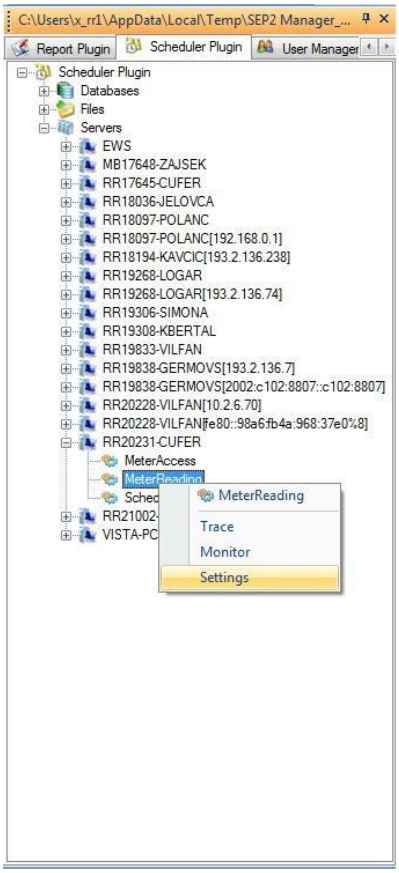
3.4 Shema delovanja in kratek opis

V *Managerju (na sliki 10 v zgornjem levem kotu)* imamo prikazane vse računalnike, ki so povezani na izbrani CS. Ko kliknemo na želeni računalnik, se nam izpišejo vsi servisi, ki tečejo na njem. Po izbiri želenega servisa *Manager* pokliče *oddaljeno konfiguracijo (na sliki 10 Remote Configuration Class)* tako, da ji poda Url naslov servisa in njegovo ime. Na podlagi teh podatkov *oddaljena konfiguracija* ustvari objekt tipa *ISystemServiceConfig*. Sledi klic metode *GetConfigNode*, ki nam vrne v izbrano spremenljivko tipa *ConfigNode* celotno konfiguracijsko datoteko. Tu imamo zdaj vse konfiguracijske parametre. Ko imamo konfiguracijsko datoteko v obliki drevesne strukture, lahko nadaljujemo z delom. Delo sedaj postane dokaj preprosto. Sprehajamo se po drevesni strukturi in popravimo, kar uporabnik želi. Torej, če uporabnik izbere *preprost pogled (na sliki 10 v spodnjem levem kotu – Simple view), oddaljena konfiguracija* pošlje drevesno strukturo *preprostemu pogledu*. Ta nam prikaže elemente v grafični obliki, ki jih potem lahko spremenimo ali samo pogledamo. Če parametre spremenimo, *preprosti pogled* pokliče metodo za shranjevanje in *oddaljena konfiguracija* te spremembe shrani tako, da pokliče metodo *SetConfigNode*. Ta poskrbi, da se spremembe shranijo v konfiguracijsko datoteko. V primeru naprednega pogleda *(na sliki 10 v spodnjem desnem kotu – Advanced view)* pa je vsa komunikacija enaka, vzpostavi se le še komunikacija z *urejevalnikom*, ki posreduje spremenjene podatke.

4 Analiza realizacije izvedbe oddaljene konfiguracije

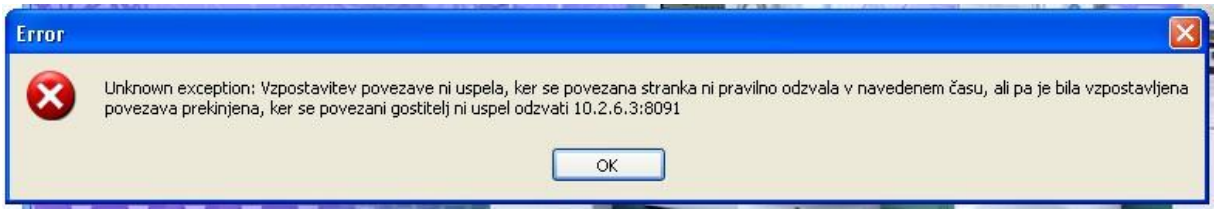
Kot vam je zdaj verjetno že jasno, sistem SEP že deluje in bo kmalu tudi uradno izdan, vendar to ne pomeni, da je razvoj s tem končan. Ravno nasprotno, vedno je potrebno kaj dodati, izboljšati, optimizirati, ipd. Tudi oddaljena konfiguracija ni idealna in bo prav tako doživela izboljšave, ki so se, po opravljenem testiranju, pokazale za potrebne.
V nadaljevanju bom podal nekaj »slabosti«, ki so se prikazale in se jih bo v prihodnje tudi popravilo. Opisal bom dva največja problema, ki se mi zdita tudi najbolj pereča. Sicer ne vplivata na samo delovanje, ampak bolj na praktičnost in lažjo uporabo. Podal pa bom tudi nekaj podatkov o zmogljivosti oziroma o obremenjevanju sistema.

Prva težava, ki se je pojavila, je ta, da se stanje strežnikov ne osvežuje. Kaj to pomeni? V *Managerju* imamo prikazovalnik strežnikov (glej sliko 11). Ta prikazovalnik nam pokaže vse strežnike, ki so se kadarkoli povezali na izbrani CS.



Slika 11: Prikaz strežnikov v Managerju.

In ravno v tem je problem, saj CS ne osvežuje stanja strežnikov. Torej, če se je strežnik prijavil na CS, četudi samo za nekaj sekund, bo tam za vedno zapisan oziroma dokler ga ne bomo ročno izbrisali iz baze. Ko bomo kliknili nanj, se bodo prikazali servisi, ki so na tem strežniku tekli. Tu se seveda ponovi ista zgodba: tudi če servis ne deluje več, bo prikazan. Če kliknemo nanj, bomo kar nekaj časa čakali, da se izteče časovna omejitev, nato se bo prikazalo obvestilo (glej sliko 12), da servis ni dosegljiv.



Slika 12: Obvestilo o nedosegljivosti servisa.

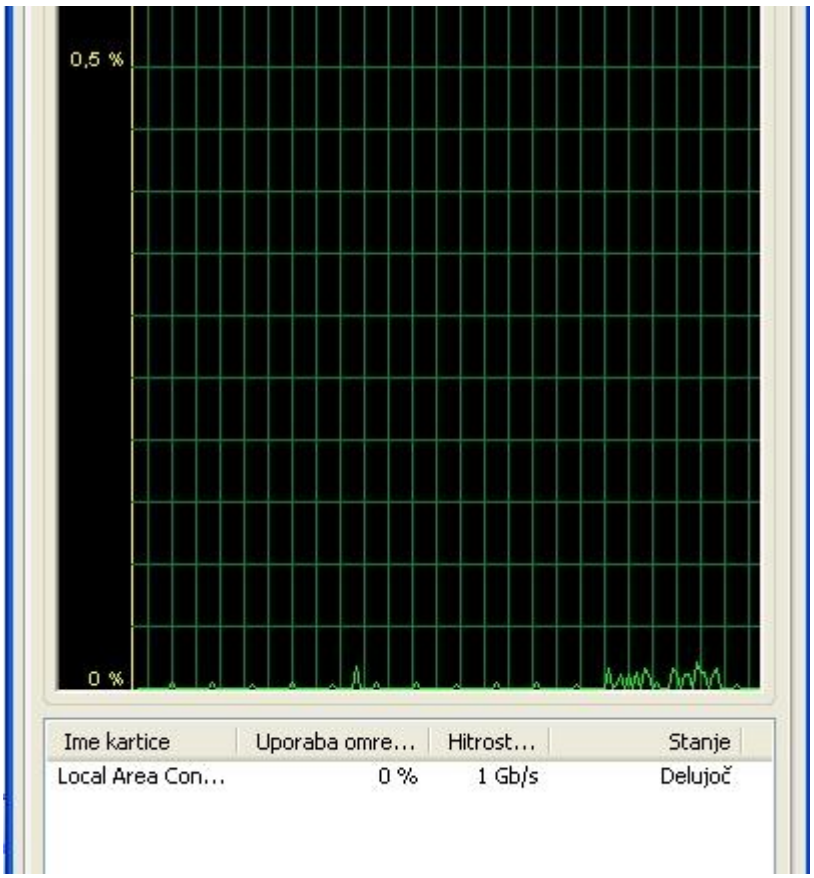
Z mojega vidika je takšno odzivanje zelo moteče. Popraviti ga bomo morali tako, da bo CS na določeno periodo preverjal, ali so servisi še vedno aktivni. Torej CS bo potrebno spremeniti tako, da bo servise, ki niso več aktivni, izbrisal s seznama. Ob vnovični povezavi nanj pa se bi ponovno vpisali. S tem bi dosegli, da imamo na listi dejansko servise, ki so trenutno aktivni in še kup neaktivnih. Tako ne bi več izgubljali časa z ugotavljanjem, če je servis sploh aktiven. Rešitev ni ravno trivialna. Potrebno se bo posvetiti problemu in CS spremeniti tako, da bo to omogočal, obenem pa moramo paziti, da se preveč ne spremeni, ker to posledično pomeni popravljanje vseh modulov.

Drugi problem, ki s stališča delovanja programa ni tako pereč, se zrcali v XML dokumentih. Vsa konfiguracija posameznega servisa je shranjena v XML dokumentih. V konfiguracijski datoteki so shranjeni vsi nastavljivi parametri, različni servisi pa imajo tudi nekaj skupnih nastavitev, ki so v bistvu pri vseh enake, vendar se morajo zaradi trenutne zasnove pojaviti v vseh konfiguracijskih datotekah. Tu se ne pojavi problem preobsežnih XML datotek, kot se lahko recimo zgodi, ko beremo podatke s števcev. Tu je predvsem stvar centralizacije, da bi imeli vse konfiguracijske parametre na enem mestu, poleg tega pa ne bi bilo več podvajanj. Kaj torej narediti? Zanimivo bi bilo poskusiti servise zastaviti tako, da imajo v XML datotekah samo najnujnejše parametre oziroma parameter. Dovolj bi bilo že to, da mu povemo, kje se nahaja CS, vse ostale nastavitve pa bi pridobil iz baze. Tako bi se lahko hitro znebili nepotrebnega podvajanja in tudi sama konfiguracija bi bila veliko bolj preprosta. Vse bi imeli na enem mestu.

Slika 13: Kaj nam prinese več prednosti?

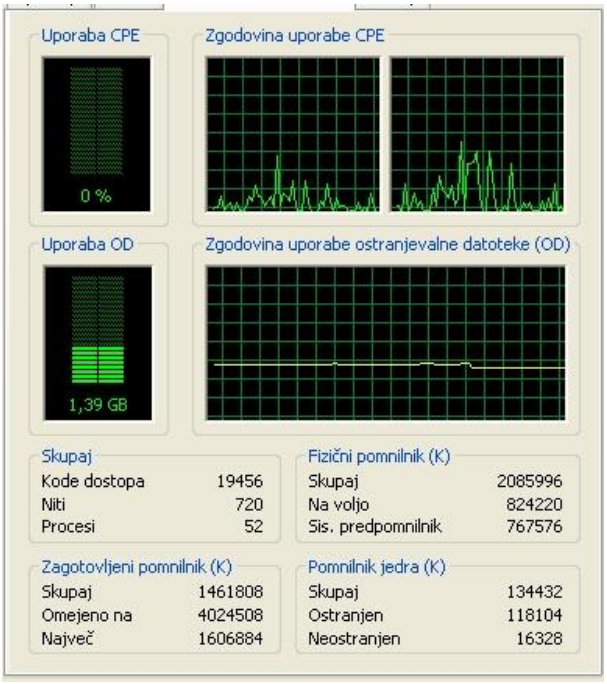
Vendar ta rešitev ne prinese prav veliko prednosti, prinese nam le veliko sprememb za njeno uresničitev. Problem se torej ne nahaja na prioritetni listi, kajti sistem deluje dobro, tudi če za konfiguracijo uporabljamo XML datoteke. Preden se bomo lotili te uresničitve, bo potrebno pretehtati, (glej sliko 13) ali spremembe prinesejo dovolj prednosti in ali so resnično potrebne.

Oddaljena konfiguracija sama po sebi ne predstavlja nobene obremenitve za sistem. XML dokumenti (konfiguracijske datoteke), s katerimi upravljamo, so majhne datoteke. Velikost se giblje od 11 KB pa tam do 20 KB, to pa danes ne predstavlja nobenega problema. V delovni pomnilnik nalagamo po eno datoteko, in če pomislimo, kako velike delovne pomnilnike premoremo danes, hitro ugotovimo, da te datoteke ne predstavljajo niti promila našega delovnega pomnilnika.

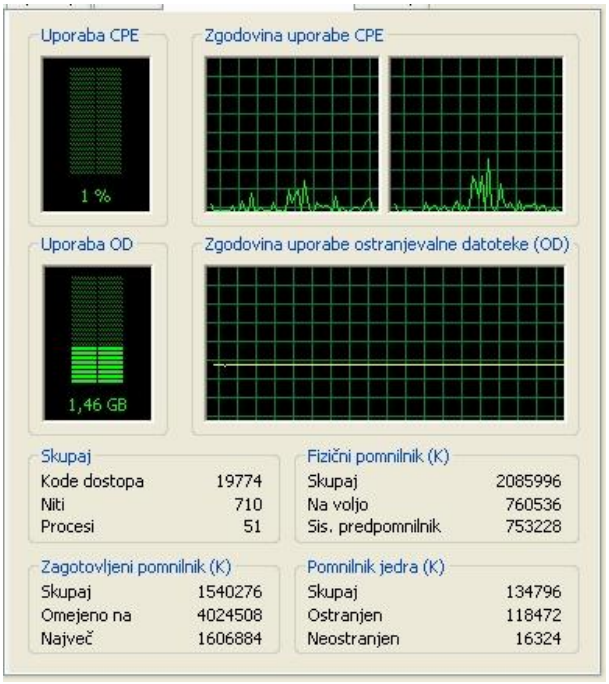


Slika 14: Večkratno hitro menjevanje servisov povzroči nekoliko aktivnosti na mreži. Če pa odpremo samo en servis, vidimo komaj opazno konico.

Poglejmo si še malo, kako je s hitrostjo. Ta je pogojena predvsem z infrastrukturo, torej z LAN in WAN povezavo (glej sliko 14), kajti ko želimo nastavljati parametre določenega servisa, moremo najprej priti do tega servisa, da preberemo konfiguracijsko datoteko v naš pomnilnik. Enkrat, ko jo imamo v naši drevesni strukturi, poteka vse lokalno. Program zahteva zelo malo procesorske moči in delovnega pomnilnika (glej sliki 15 in 16), tako da aplikacija ni kritična s stališča zmogljivosti. Kakor prikazujeta tudi sliki, obremenitev računalnika in omrežja skorajda ni relevantna, tako da ta del programa tehnikom ne bo povzročal večjih preglavic. Odzivnost in hitrost delovanja pa se lahko hitro poslabša, če so povezave do računalnikov, na katerih se nahajajo servisi, zelo slabe in nestabilne (kar pomeni, da večkrat pade povezava). V tem primeru bodo servisi večkrat nedostopni in naj tu ponovno izpostavim, da CS ne osvežuje stanja servisov, zato bomo kaj kmalu naleteli na le-te in čakali, da se iztečejo časovne omejitve, ki pa delovanje zelo upočasnijo. Vendar to so ekstremni primeri, ki se v praksi ne bodo pojavili, ker servisi morajo imeti stabilne WAN in LAN povezave za komunikacijo z bazo.



Slika 15: Stanje sistema pred uporabo programa.



Slika 16: Stanje sistema med uporabo programa. Opazno je manjše povečanje aktivnosti na procesorju in večja poraba pomnilnika.

Nekdo v svetu računalništva je predstavil svoje nove programe na prav poseben način in sicer, prišel je na oder in rekel: *»Naredili smo to in to /.../ program ne zna delati tega, tu je malo okoren, itd.«* Kaj sem hotel s tem povedati? Gospod je program predstavil tako, da je podal vse njegove slabosti in niti ene prednosti. Ne vem, če je to ravno najboljši sistem, strinjam pa se, da je potrebno jasno povedati, česa program ne zmore, zato sem podal probleme, ki še čakajo na odpravo. Da ne omenjam same slabosti, naj podam še nekaj prednosti. Program izpolnjuje zahteve, saj je olajšal konfiguriranje servisov in omogočil hitrejše ter preglednejše delo. Zdaj lahko prosto rečem, da so bili vsi zadani cilji izpolnjeni.

5 Zaključek

Cilje pričujoče diplomske naloge je bil narediti grafični vmesnik, s katerim bomo konfigurirali servise, ki tečejo v sistemu SEP, obenem pa je bil moj tihi cilj spoznati se s sistemom, ki me bo spremljal na delu in ga bom še nadaljnje razvijal.

Na samem začetku sem si postavil sledeča načela: program mora biti preprost, uporabniku prijazen in razumljiv, omogočati mora spreminjanje konfiguracijskih datotek in s tem posledično delovanja servisov. Načel sem se poskušal držati, vendar pa se med programiranjem kaj hitro pokaže, da smo z določenimi stvarmi omejeni, predvsem zaradi zahtev. Šele z delom se pokaže, da nekatere stvari ne bomo mogli narediti, kot smo si jih zamislili, ker smo ubrali napačen pristop. Podobno se je zgodilo tudi meni, tako da sem moral napraviti korak nazaj in začeti vse znova, vendar se s tem nisem preveč obremenjeval. Vse skupaj sem vzel kot dobro šolo in ravno te napake so mi pomagale spoznati sistem, ki se na začetku zdi zelo preprost in samoumeven, vendar ko sem dobil nalogo in spoznal, da sploh ne vem, kaj delam, se je kaj hitro pokazalo, kako zelo je sistem kompleksen in velik.

Počasi in z zanimanjem sem poprijel za delo, veliko stvari sem vprašal sodelavce, ki so mi pomagali pri spoznavanju, dobil sem dokumentacijo in se lotil branja. Šele, ko sem sistem spoznal v grobem, se mi je začelo svitati, kaj moram narediti, zakaj je to sploh smiselno in uporabno. Začel sem programirati in tu so se pojavile nove težave, ki pa sem jih s spoznavanjem sistema sproti odpravljal. Najprej sem naredil en del programa, potem se lotil drugega, vmes pa sem spoznal, da nekatere, že narejene rešitve, niso najbolj idealne in jih bo potrebno v bližnji prihodnosti popraviti.

Menim, da sem s programom dosegel želene cilje, saj program poenostavlja delo s konfiguracijskimi datotekami, je preprost, preprečuje nam vnašanje napak oziroma, da ne bom zvenel preveč naivno, v večji meri nam prepreči, da naredimo napako, še vedno pa se lahko kašna pojavi, predvsem pa sem spoznal sistem, ki se mi je na začetku zdel tako nedostopen. Nestrpno pričakujem pohod sistema na tržišče in odzive uporabnikov.

Seznam slik

Literatura

[1] Wikipedia, PLC comunication . Dostopno na:
http://en.wikipedia.org/wiki/Power_line_communication

[2] Wikipedia, GSM. Dostopno na:
http://en.wikipedia.org/wiki/Global_System_for_Mobile_Communications

[3] Wikipedia, GPRS. Dostopno na:
http://sl.wikipedia.org/wiki/GPRS

[4] Wikipedia, Automatic Meter Reading. Dostopno na:
http://en.wikipedia.org/wiki/Automatic_meter_reading

[5] SWS_SEP2WSystem.doc, interna specifikacija podjetja Iskraemeco d.d.

[6] SWS_SEP2CoreService.doc, interna specifikacija podjetja Iskraemeco d.d.

[7] SWS_SEP2Manager.doc, interna specifikacija podjetja Iskraemeco d.d.

[8] SWS_SEP2MeterReading.doc, interna specifikacija podjetja Iskraemeco d.d.

[9] SWS_SEP2ReportService.doc, interna specifikacija podjetja Iskraemeco d.d.

[10] SWS_SEP2Scheduler.doc, interna specifikacija podjetja Iskraemeco d.d.

[11] SWS_SEP2MeterAccess.doc, interna specifikacija podjetja Iskraemeco d.d.

[12] Microsoft, MSDN Library. Dostopno na:
http://msdn.microsoft.com/library

[13] Wikipedia, XML. Dostopno na:
http://en.wikipedia.org/wiki/XML