INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Evaldas Vaičiukynas

VĖJO ELEKTRINIŲ PARKO INFORMACINĖS SISTEMOS PROTOTIPAS

Magistro darbas

Recenzentas:
prof. habil. dr. Antanas Nemura

Vadovas:
doc. Lina Nemuraitė

KAUNAS, 2006

SUMMARY

Wind farm information system prototype

In the following Master's degree research main information and communication aspects in wind power plant farm analyzed and demonstrated by constructing basic prototype. The wind energy market in Europe is growing and needs have arisen to develop standard-based information systems to support future expansion of wind energy in Lithuania. Introduction and first chapter overview the background for modeling such system. Requirements for data structures representing main measurements and information exchange inside and outside of the system are defined on the base of IEC 61400-25 standard "Wind Turbine Generator Systems - Part 25: Communications for monitoring and control of wind power plants". Models, comprising such system are described and possible communication topologies enumerated. From several existing communication profiles, defined in above mentioned standard, SOAP / XML based web services for communication architecture are chosen and prototype built.

Article based on this work was written together with coauthor professor A. Nemura and presented in international KTU conference "Automation and Control technologies – 2005" and LEI conference "Application of information and management technologies in electricity energetics". The purpose of article is to form and refine the wind farm information system problem introducing IEC 61400-25 standard.

TURINYS

Lentelių sąrašas

Paveikslėlių sąrašas

ĮVADAS

Tobulinant Lietuvos energetikos sistemą ir derinantis prie Europos Sąjungos standartų Lietuva įsipareigojo dalį naudojamos energijos gauti iš atsinaujinančių energijos šaltinių. Pagal šiuo metu esančią situaciją, galia, gaunama iš tokių energijos šaltinių, turėtų būti padidinta dvigubai. Lietuvoje iki šiol pagrindiniai vietinio generavimo šaltiniai, naudojantys atsinaujinančius resursus, buvo nedidelės hidroelektrinės. Kitos atsinaujinančios energijos rūšys kaip vėjas ir saulė nebuvo ženkliau naudojamos.

Užsienio šalyse vėjo energija sėkmingai naudojama platesniu mastu jau keletą metų ir sulaukia vis didesnio pripažinimo, ypač kai valstybių vadovybės sudaro palankias sąlygas šiai energetikos šakai vystytis. Tuo pačiu auga ir vėjo jėgainių pasiūla ir įvairovė. Vėjo jėgainės tobulinamos, kad kuo labiau atitiktų naujausius standartus, būtų pritaikytos įvairiomis sąlygomis, nes kiekvienu konkrečiu atveju pasirenkamas tinkamiausias tai vietovei tiek techniniu, tiek ekonominiu požiūriu variantas.

Sutvarkius Lietuvos energetinių išteklių taupymo įstatyminę bazę ir pagal Europos Sąjungos direktyvas išsprendus finansavimo bei pagamintos elektros energijos supirkimo klausimus, Lietuvoje prasideda sparti vėjo elektrinių statyba. 2003 metų rudenį daugiau kaip 10 organizacijų paskelbė apie virš 700 MW suminės įrengtosios galios vėjo elektrinių statybos planus, ir šie skaičiai vis auga. Vėjo elektrinių statybos vietos dėl didesnių vėjo greičių telkiasi Lietuvos pajūryje ir netgi numatoma galimybė rengti jas pačioje jūroje.

Pagal vėjo elektrinių statybos planą, Lietuvoje numatoma iki 2010 m. įrengti keletą vėjo elektrinių parkų, kurių bendra įrengta vardinė galia sudarys 180 MW.

Pastatytam vėjo elektrinių parkui prižiūrėti ir valdyti neišvengiamai reikalingi kiekvienos jėgainės informacijos mainai su išore (pvz. parko dispečeriu). Šiam tikslui dabartinėse vėjo jėgainėse integruojamas valdiklis (angl. *IED - intelligent electronic device*), kuris yra atsakingas už vidinį komponentų stebėjimą ir gali siųsti duomenis išoriniams gavėjams bei gauti duomenis iš jų.

Komunikacijos su valdikliu terpę, naudojamo protokolo struktūrą, perduodamų duomenų formatą ir komandų semantiką apsprendžia gamintojai, o kai kurie teikia ne tik valdiklio sąsajos dokumentaciją bet ir savo programinę įrangą. Tačiau norint centralizuotai stebėti ir valdyti visas parko elektrines, susiduriama su sunkumais, nes nevienodus valdiklius turinčioms jėgainėms pasiekti tenka realizuoti atskirus komunikacijos būdus ir duomenų semantiką. Kuo parke daugiau skirtingų valdiklių, tuo sudėtingesnis darosi tokios informacinės sistemos uždavinys. Jau veikiančiame parke pastačius jėgainę, kurios valdiklio sistema nepalaiko, tektų keisti programinę įrangą. Nesuderinamumai iškiltų ir bandant jungti tokius vėjo elektrinių parkus į bendrą informacinį tinklą.

Kaip išeitis iš susidariusios situacijos 2001 m. gegužę suformuota IEC (International Electrotechnical Commision) TC (Technical Committee) 88 darbo grupė kurti IEC 61400-25 standarto, kuris pateiktų vieningą duomenų semantiką ir apibrėžtų komunikacijos aspektus. Užbaigtas standarto darbinis variantas 2003 m. rugpjūčio 8 d. pateiktas įvertinimui dalyvaujančiom jo kūrime šalim ir publikuojamas IEC tinklapyje. Tikimasi, kad jis bus priimtas ir taikomas pasauliniu mastu kaip tarptautinis standartas vėjo elektrinių srityje.

IEC 61400-25 standartas apibrėžia specifinę vėjo elektrinėms informaciją, jos semantinę struktūrą, komunikacijai naudojamus protokolus ir informacijos apsikeitimo mechanizmus. Šiuo atžvilgiu standartas detalizuoja pagrindinius ir bendrus visoms vėjo elektrinėms komponentus (nepriklausomai nuo gamintojo) ir komponento reprezentuojamos informacijos apsikeitimo aspektus.

Vėjo elektrinių parko informacinėje sistemoje naudojama lengvo kliento–serverio architektūra. Informacinę sistemą sudaro šie komponentai: 1) vėjo elektrinių parko informacijos modelis (statinė dalis – virtualus parko vaizdas); 2) informacijos mainų modelis (dinaminė dalis – informacijos apsikeitimo procesai); 3) informacijos ir jos mainų modelių priskyrimas komunikacijos variantui (komunikacinė dalis – duomenų perdavimo pasirinktu protokolu sąsaja).

Vėjo elektrinių parko komponentų informacijos modelis yra objektinis ir semantiškai standartizuotas. Komponentai modeliuojami kaip informaciniai objektai, identifikuojant visus jų atributus bei funkcionalumą ir sudarant medžio tipo hierarchiją. Kiekvienas atributas turi vardą ir paprastą arba sudėtingą tipą (klasę) ir aprašo konkrečią įrenginio informaciją, kurią galima nuskaityti arba pakeisti.

Informacijos mainų modelis įgalina standartizuotą informacijos apsikeitimą tarp vartotojų ir realaus įrenginio, kuris virtualiai atvaizduojamas informacijos modelyje. Informacijos mainų modelį sudaro visos serverio teikiamos paslaugos.

Informacijos ir jos mainų modeliai sudaro sąsają (angl. *interface*) tarp kliento ir serverio. Ši sąsaja leidžia serveriui bendrauti su skirtingais klientais vienu metu, nepriklausomai nuo jų architektūros ar naudojamos programinės įrangos tol, kol jie naudoja serverio palaikomą komunikacijos variantą ir formuoja teisingas užklausas ar valdymo komandas.

Prototipo kūrimui pasirenkamas SOAP / XML komunikacijos variantas, kuriame sąsaja specifikuojama WSDL (angl. *web services definition language*) dokumentu. IEC 61400-25 standartą atitinkančio WSDL dokumento sukūrimui skiriamas didžiausias dėmesys, nes toliau jis yra panaudojamas tiek serverio, tiek kliento skeletui generuoti.

Skeleto realizacijai pasirinkta Java programavimo kalba ir Eclipse platforma. Panaudojant *Apache XMLBeans* technologiją ir WS/XSUL2 bibliotekos įrankį *xwsdlc* iš WSDL dokumento sugeneruojamos duomenų klasės ir žiniatinklio paslaugų (angl. *web services*) sąsajos (interfeisai).

Šio darbo pagrindu parašyti ir pristatyti konferencijose du straipsniai (bendraautorius A. Nemura):

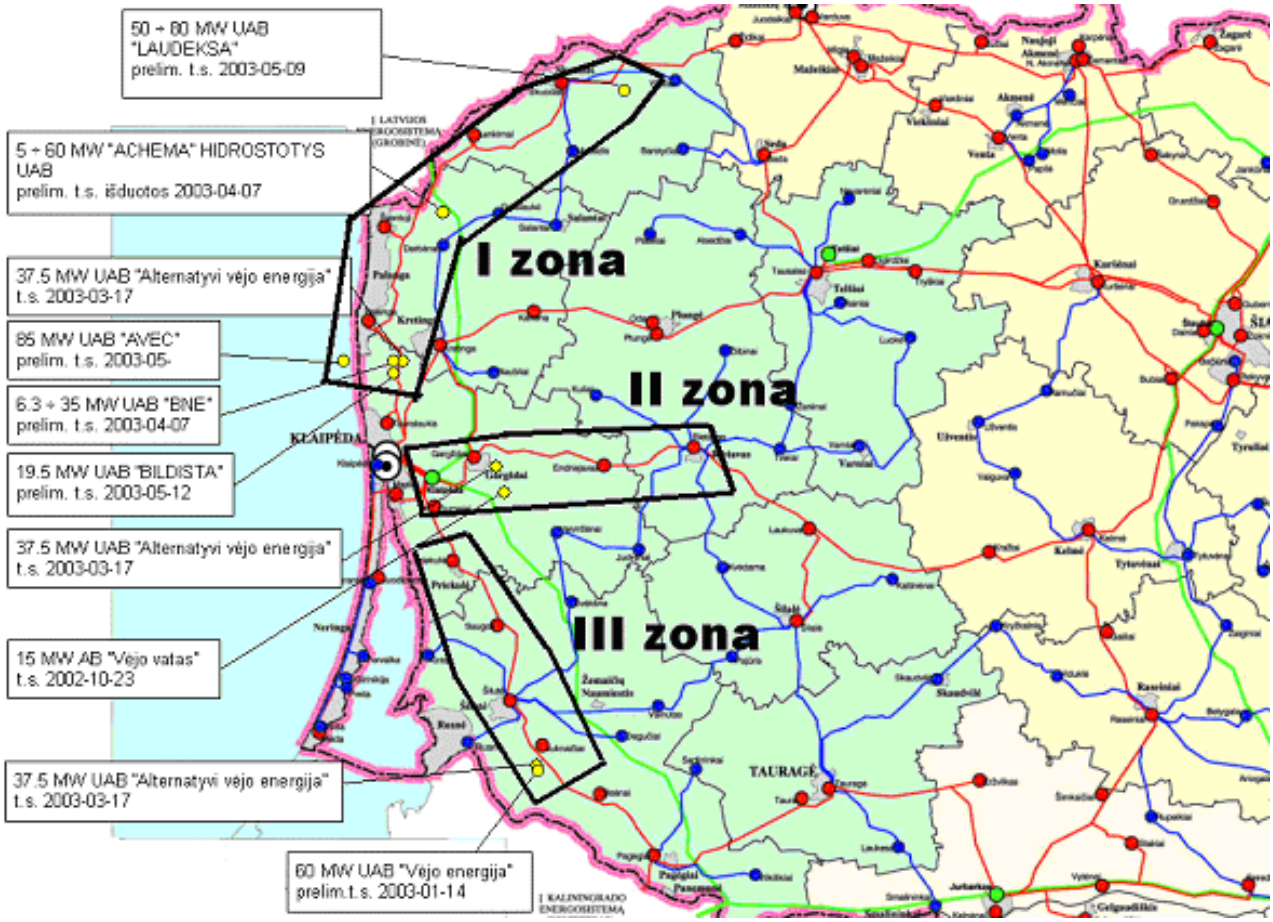
- ,,Vėjo elektrinių parko informacinės sistemos prototipas", tarptautinė konferencija ,,Automatika ir valdymo technologijos – 2005", KTU, 2005 m. gegužės 12 d.
- ,,Vėjo elektrinių parko informacinės sistemos architektūra ir komunikacijos aspektai", Lietuvos MA Technikos mokslų skyriaus susirinkimas – konferencija ,,Informacinių ir valdymo technologijų taikymas elektros energetikoje", LEI, 2005 m. birželio 10 d.

1. Vėjo elektrinių parko informacinės sistemos analizė

1.1. Vėjo elektrinių parko tyrimo sritis ir objektas

Vėjo elektrinės Lietuvoje numatomos statyti visame pajūryje, padalinant jį į zonas, matomas 1 pav.

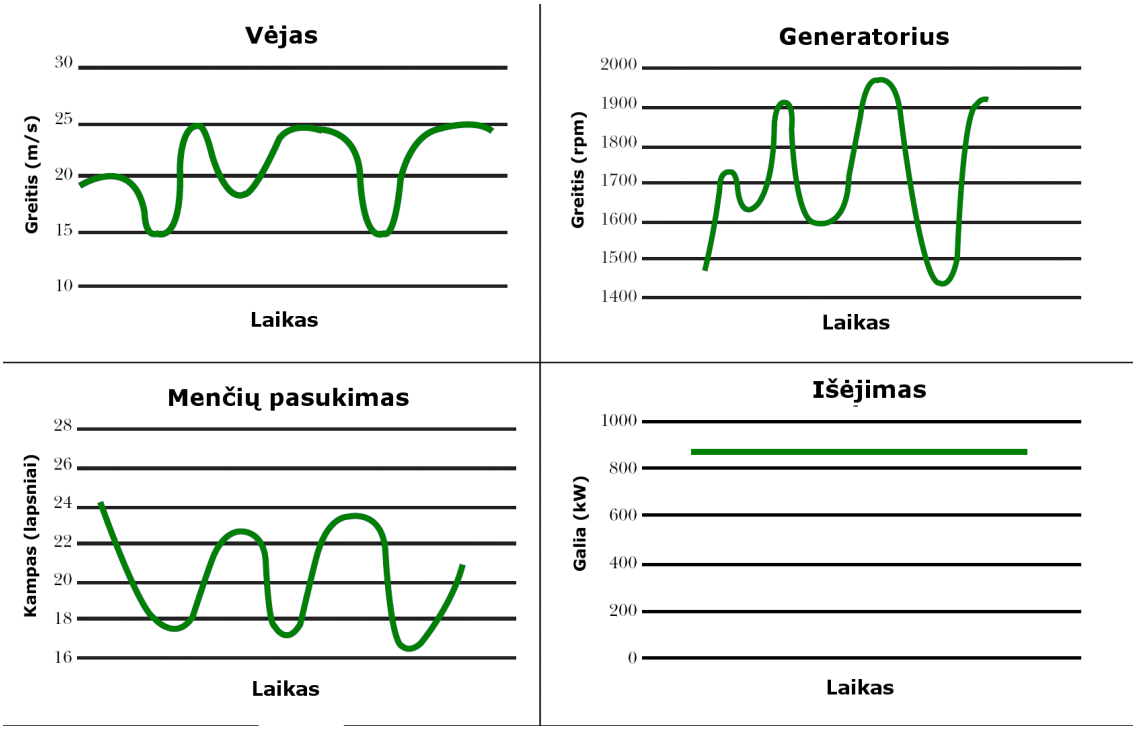
Šiame darbe kaip pavyzdys nagrinėjamas vėjo elektrinių parkas, kuris bus statomas 3-je zonoje (be tinklo plėtros), šalia Juknaičių, netoli 110 kV elektros perdavimo linijos. Darbus atliks UAB "Alternatyvi vėjo energija".



1 pav. Lietuvoje numatomų statyti vėjo elektrinių parkų zonos

Parke yra 44 vienodo tipo vėjo elektrinės, kurių kiekvienos vardinė galia yra lygi 0.85 MW. Bendra parko instaliuota vardinė galia yra 37.4 MW.

Kiti duomenys: 1) vėjo elektrinių tipas V52 – 0.85 MW (gamintojas - firma VESTAS); 2) vėjo turbinos diametras - 52 m.; 3) vėjo turbinos menčių skaičius – 3; 4) generuojamos galios reguliavimas Pitch/OptiSpeed sistema; 5) minimalus vėjo greitis - 4m/s; 6) vardinis vėjo greitis - 16 m/s; 7) maksimalus vėjo greitis - 25 m/s; 8) generatorius asinchroninis su OptiSpeed sistema; 9) vardinis dažnis - 50 Hz.; 10) vardinė įtampa - 690 V. [VESTAS]



2 pav. Vestas vėjo elektrinės V52-850 kW su OptiSpeed™ technologija laikinės charakteristikos

Generuojamos galios reguliavimo sistema pasižymi tuo, kad esant vėjo greičiui didesniam negu vardinis turbinos menčių pasukimo kampas reguliuojamas atsižvelgiant į vėjo greičio svyravimus ir taip generuojama nepertraukiama ir nesvyruojančios galios elektros energija. 2 pav. pateikiami tai iliustruojantys grafikai.

1.2. Tyrimo tikslai ir uždaviniai

Šio darbo tikslas yra išnagrinėti vėjo elektrinių veikimo bei prijungimo prie išorinių tinklų principus ir tokios informacinės sistemos pagrindinius reikalavimus, juos išanalizuoti, pasirinkti tinkamiausią tokios sistemos variantą ir sukurti vėjo elektrinių parko informacinės sistemos prototipą.

Analizė apima tyrimo srities, objekto ir problemos aprašymą, remiantis egzistuojančiais šioje srityje standartais.

Pagrindinis sprendžiamas uždavinys – vėjo elektrinių parko informacinės sistemos dalių (modelių) išskyrimas ir jų detalizavimas bei projektavimas.

1.3. Vėjo elektrinių prijungimo taisyklės

Vėjo elektrinių parko prijungimas turi būti atliktas atsižvelgiant į „Vėjo elektrinių prijungimo prie Lietuvos elektros energetikos sistemos technines taisykles“, kurių tikslas yra garantuoti, kad naujai prie elektros tinklo prijungiamos vėjo elektrinės turės tokias konstrukcijos, valdomumo ir veikos dinamines

savybes, kurios leistų vėjo elektrines prijungti prie Lietuvos elektros sistemos perdavimo ar skirstomųjų tinklų, o elektros vartotojams pirkti kokybės reikalavimus atitinkančią vėjo elektrinių gaminamą elektros energiją.

1.3.1. Apkrovų srautai

Vietinių vartotojų srautas, būdamas priešingos krypties, praplečia skirstomojo tinklo galimybes perduoti vėjo elektrinių galią:

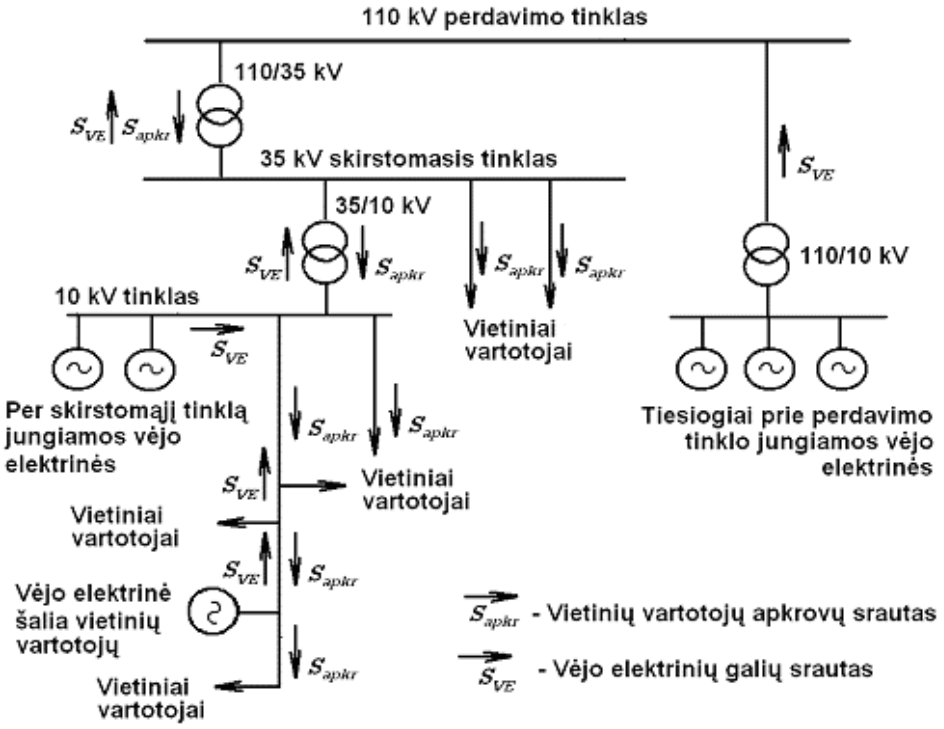
$$\left|S_{VE} - S_{i\ apkr}\right| \le S_{i\ leist}.$$

Čia:

$S_{i\ leist}$ – skirstomojo tinklo i–tojo elemento didžiausia leistinoji galia;

S_{VE} – vėjo elektrinių suminė pilnutinė galia;

$S_{i\ apkr}$ – tinklo i–tojo vietinio vartotojo apkrovos sutartinė galia.



3 pav. Vėjo elektrinių galių ir vietinių vartotojų apkrovų srautų kryptys

Pagal prijungimo taisykles, reikia numatyti atvejį, kai vietiniai vartotojai ima mažiausią galią, arba yra atsijungę (atjungti), o vėjo elektrinė generuoja savo didžiausią galią. Priimant, kad $S_{apkr} \approx 0$, gauname vietinio tinklo i-tojo elemento (linijos, transformatoriaus) pralaidumo sąlygą:

$$S_{VE} \le S_{i\ leist}.$$

Išplečiant pralaidumo sąlygą visiems skirstomojo tinklo elementams, per kuriuos tekės vėjo elektrinės galios srautas, reikia įvertinti tik mažiausio pralaidumo elementą:

$$S_{VE} \leq \min(S_{i\ leist}).$$

1.3.2. Aktyviosios galios valdymas

Kiekviena tiek atskira, tiek parko vėjo elektrinė turi būti valdoma atskirai ir jos generuojamą galią turi būti galima sumažinti žemiau jos didžiausios galios 20% vertės mažiau kaip per 2 sekundes.

Generuojamoji vėjo elektrinių parko galia, nustatyta kaip 1 minutės vidurkio vertė, bet kurį akimirksnį neturi viršyti generavimo leistinąją ribą daugiau kaip per 5% vėjo parko vardinės galios.

Turi būti numatyta galimybė kiekvienos vėjo elektrinės generuojamos galios leistinąją ribą ir galios kitimo greitį bet kada keisti iš operatoriaus dispečerinio valdymo punkto.

Turi būti galima mažinti generavimą valdant elektrinių parką iš atitinkamo tinklo operatoriaus dispečerinio valdymo punkto. Tiek mažinant generavimą, tiek grįžtant į įprastinį galios generavimo lygį, turi būti galima iš atitinkamo operatoriaus dispečerinio punkto valdyti reguliavimo greitį ir jį pasirinkti vardinės galios 10…100 % per minutę srityje.

Įprastai didinant vėjo elektrinių parko galią (tiek vėjui tik pradedant pūsti, tiek vėjo greičiui toliau didėjant), yra numatomas didžiausias leistinas MW/min didėjimo greitis.

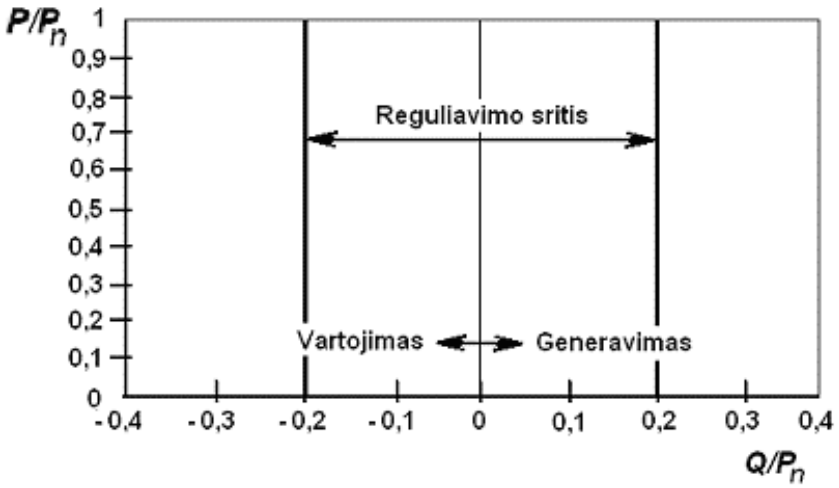
Turi būti galima valdyti vėjo elektrinių parko generavimą taip, kad ji neviršytų generavimo leistinosios ribos (MW). Vėjo elektrinių parko generavimo leistinąją ribą turi užduoti perdavimo tinklo operatorius, pasiųsdamas iš dispečerinio punkto atitinkamą signalą, arba ji turi būti nustatyta pagal elektrinės prijungimo taško vietinio dažnio ir (arba) įtampos vertes.

Turi būti galima mažinti kiekvienos vėjo elektrinės generuojamąją galią, esant bet kuriai užduotos galios vertei nuo 0% iki 100% vardinės galios srityje. Skirtumas tarp užduotos galios vertės ir išmatuotosios 1 minutės galios vidutinės vertės prijungimo taške neturi nukrypti daugiau kaip per vėjo elektrinės (parko) vardinės galios 5% vertę.

Galios ir dažnio valdymas turi būti toks, kad valdomas galios sumažinimas ar didinimas, kartu kontroliuojant dažnį, vėjo elektrinės valdikliui gavus atitinkamą komandą, būtų įvykdytas ne vėliau kaip per 30 sekundžių.

1.3.3. Reaktyviosios galios mainai

Reaktyviosios galios, kurią vėjo elektrinė (apimant ir aukštinantįjį transformatorių) ima arba atiduoda į perdavimo arba skirstomąjį tinklą, 5 minučių vidutinės vertės turi būti reguliavimo srities viduje, kaip parodyta 3 paveiksle. Šis reikalavimas, jei galios mainai mažesni kaip 25 kvar, netaikomas.



4 pav. Reaktyviosios galios mainų reikalavimai (P – aktyvioji, Q – reaktyvioji)

Reaktyviosios galios kiekis, kurį vėjo elektrinė gali suvartoti ar pateikti į tinklą virš 3 paveiksle nurodytų ribų, turi būti nustatytas prijungiant vėjo elektrinę prie prijungimo taško ir pagal susitarimą su perdavimo ar skirstomojo tinklo operatoriumi gali būti panaudotas elektros sistemoje reguliuojant reaktyviąją galią. Tai taip pat taikoma vėjo elektrinių parkui.

1.3.4. Apkrovų ir generuojamų galių matavimai

Atsižvelgiant į apkrovų ir generuojamų galių srautų analizės tikslą, gali būti svarbu vėjo vėjo elektrinių parko galią išreikšti vardinėmis P_n ir Q_n, didžiausiomis leistinosiomis $P_{max\ leist}$ ir $Q_{max\ leist}$, 60 sekundžių vidurkio P_{60} ir Q_{60} arba 0,2 sekundės intervalo (10 periodų) vidurkio $P_{0,2}$ ir $Q_{0,2}$ galiomis.

Kai prie prijungimo taško yra prijungiama ne viena, o kelios vėjo elektrinės, jų vardinės, didžiausios leistinosios ir 60 sekundžių intervalų galios yra įprastai susumuojamos. Tačiau skaičiuojant bendrą kelių elektrinių 0,2 sekundės trukmės intervalų galių sumą pasireiškia tikimybinis generuojamosios galios pobūdis, ir jį reikia įvertinti:

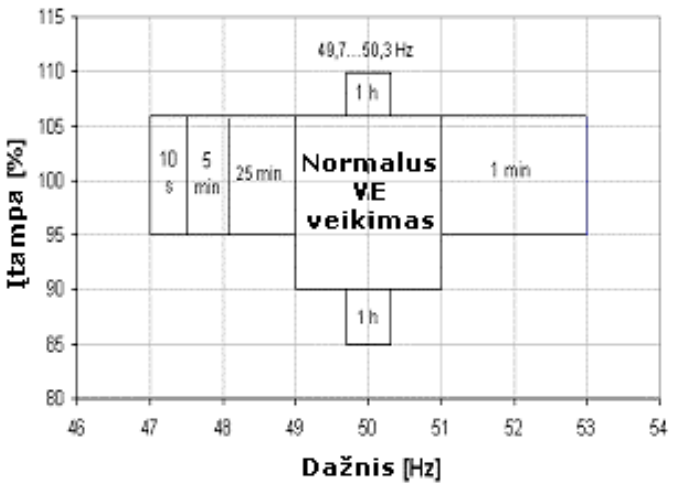
$$P_{0,2\Sigma} = \sum_{i=1}^{N_{VE}} P_{n,i} + \sqrt{\sum_{i=1}^{N_{VE}} \left(P_{0,2i} - P_{n,i}\right)^2};$$

$$Q_{0,2\Sigma} = \sum_{i=1}^{N_{VE}} Q_{n,i} + \sqrt{\sum_{i=1}^{N_{VE}} \left(Q_{0,2i} - Q_{n,i}\right)^2}.$$

1.3.5. Vėjo elektrinės paleidimas ir stabdymas

Vėjo elektrinę įjungti ir išjungti turi būti galima taip pat valdant iš tinklo operatoriaus dispečerinio punkto. Elektrinė turi būti atjungiama pagal nuokrypų nuo įprastinės veikos trukmes, pavaizduotas 1 lentelėje.

1 lentelė. Atjungimo kriterijai.

Atjungimo kriterijus	Statos vertė		Leistinoji trukmė		Konstrukcijoje numatytos įtampos ir dažnio ribos
Per žema įtampa	0,9 U_n	V	10..60	s	
Per daug žema įtampa	0,85 U_n	V	≤ 10	s	
Per aukšta įtampa	1,06 U_n	V	60	s	
Viršįtampis	1,1 U_n	V	200	ms	
Per daug aukštas dažnis	50,5 su dažnio valdymu 51	Hz Hz	200 200	ms ms	
Per daug žemas dažnis	47	Hz	200	ms	

Esant dažniu ar (ir) įtampai leistinose ribose, tačiau susiklosčius avarinei situacijai perdavimo ar skirstomajam tinkle, atitinkamo tinklo operatorius gali atjungti vėjo elektrinę, pasiunčiant atjungimo signalą iš dispečerinio valdymo punkto.

Jei per ankstesnį perdavimo ar skirstomojo tinklo gedimą vėjo elektrinė buvo atjungta valdant iš operatoriaus dispečerinio valdymo punkto ar automatiškai gavusi atjungimo komandą dėl neleistinų nukrypimų nuo įprastinės veikos sąlygų perdavimo ar skirstomajame tinkle, veikos sąlygoms tapus įprastinėms (žr. lentelės paveikslą) ji neturi būti vėl įjungiama tol, kol negaus analogiško įjungimo komandos signalo.

Tam, kad vėjo elektrines stabdant per audrą būtų išvengta staigaus elektros generavimo kryčio, vėjo parko elektrinių stabdymo greičių statos turi tarpusavyje skirtis.

1.3.6. Metrologiniai reikalavimai

Vėjo elektrinių parko prijungimo taške turi būti matuojami mažiausiai šie dydžiai:

– tinklo įtampa;

– vėjo elektrinės (elektrinių parko) srovė;

– vėjo elektrinės (elektrinių parko) pateikiama į tinklą aktyvioji galia;

– vėjo elektrinės (elektrinių parko) vartojama iš tinklo reaktyvioji galia;

– vėjo elektrinės (elektrinių parko) pateikiama į tinklą reaktyvioji galia;

– vėjo elektrinės (elektrinių parko) pateikiama į tinklą (parduodama) aktyvioji energija;

– vėjo elektrinės (elektrinių parko) vartojama iš tinklo (perkamoji) aktyvioji energija,

– vėjo elektrinės (elektrinių parko) pateikiama į tinklą reaktyvioji energija;

– vėjo elektrinės (elektrinių parko) vartojama iš tinklo reaktyvioji energija;

– netiesinių iškreipių suminis faktorius prijungimo taške;

– įtampos mirgėjimo aštrumo trumpalaikis (10 min. intervalo vidutinis) rodiklis prijungimo taške.

1.3.7. Informacijos mainai

Duomenys, kurie turi būti perduodami tarp vėjo elektrinių parko prijungimo taško ir tinklo operatoriaus dispečerinio valdymo punkto pateikiami 2 lentelėje, o tarp vėjo elektrinių parko ir vietinio operatoriaus dispečerinio valdymo punkto pateikiami 3 lentelėje.

2 lentelė. Duomenys tarp vėjo elektrinių parko prijungimo taško ir tinklo operatoriaus

Duomenys	Kiekis ir tipas
Komercinės apskaitos duomenys	
Per prijungimo tašką į tinklą parduodama aktyvioji energija, kWh	M
Per prijungimo tašką iš tinklo vartojama aktyvioji energija, kWh	M
Per prijungimo tašką į tinklą pateikiama reaktyvioji energija, kvarh	M
Per prijungimo tašką iš tinklo vartojama reaktyvioji energija, kvarh	M
Matavimų duomenys	
Per prijungimo tašką į tinklą parduodama aktyvioji galia, kW	M
Per prijungimo tašką iš tinklo vartojama aktyvioji galia, kW	M
Per prijungimo tašką į tinklą pateikiama reaktyvioji galia, kvar	M
Per prijungimo tašką iš tinklo vartojama reaktyvioji galia, kvar	M
Prijungimo taško įtampa, kV	M
Per prijungimo tašką tekanti srovė, A	M
Netiesinių iškreipių suminis faktorius prijungimo taške, %	M
Įtampos mirgėjimo aštrumo trumpalaikis (10 min. intervalo vidutinis) rodiklis	M

Rekomenduojama numatyti galimybę 2 lentelėje parodytus duomenis į operatoriaus dispečerinio valdymo punktą perduoti pagal informacinių sistemų pareikalavimą, o operatoriaus užduoto intervalo ribas viršijančius duomenų pokyčius būtina perduoti iš karto.

Galios skaičiaus ženklas turi atitikti generatoriams taikomą tvarką. Reaktyvioji galia turi būti neigiama, jei generatorius ją vartoja, ir teigiama, jei generatorius reaktyviąją galią generuoja.

3 lentelė. Duomenys tarp vėjo elektrinės ir vietinio operatoriaus

Duomenys	**Kiekis ir tipas**
Didžiausia galimos generuoti galios riba, iki (nuo)	DO ir DT
Generuojamoji arba vartojamoji aktyvioji galia, ±kW	K, B ir M
Reguliavimo greitis, kW/min	B
Vidutinis 60 sek intervalo vėjo greitis vėjaračio veleno aukštyje, m/s	M
Dažnio valdymas, leidžiamas (neleidžiamas)	DO ir DT
Reguliavimo srities viršutinė ir apatinė dažnio ribos, Hz	2 B
Nukrypusių nuo vardinio dažnių nejautrumo sritis, Hz	2 B
Reaktyviosios galios valdymas, leidžiamas (neleidžiamas)	DO ir DT
Vartojamoji arba generuojamoji reaktyvioji galia, ±kvar	2 K, B ir M
Vėjo turbinos įjungimas (išjungimas)	DO ir DT
Vėjo elektrinė įjungta (išjungta)	DT
Išjungta dėl per didelio vėjo greičio	T
Išjunta perdavimo ar skirstomojo tinklo operatoriaus	T
Neveikia dėl pažaidos (apžiūros ir aptarnavimo darbų)	T
Žemosios pusės įtampa, V	M
Vėjo elektrinės atpažinimo numeris (Valstybinio sertifikato registracijos numeris)	V

4 lentelė. 2 ir 3 lentelėse naudojami sutrumpinimai.

O:	Skiltis	M:	Matuojamoji vertė
T:	Būklė	DO:	Dviejų skilčių
K:	Dabartinė didžiausia leistinoji vertė	DT:	Keturių būklių indikacija
B:	Statos vertė	V:	(18 skaitmenų) vertė

Visos didžiausios leistinosios ir visos išmatuotos vertės turi būti tos akimirkos vertės ir turi būti perduodamos mažiausiai 11 bitų skyra plius ženklas ir 1.0 tikslumo klasės.

Didžiausios leistinosios generuoti galios ir išmatuotoji aktyviosios galios vertės (žr. 3 lentelę) turi būti automatiškai perduodamos į operatoriaus dispečerinio valdymo punktą kai tik nauja vertė pakeičia anksčiau perduotąją vertę. Turi būti galima kiekvieną matavimą filtruoti. Rekomenduojama taikyti „Įvykis – pokytis“ metodą, kuris reiškia, kad matuojamosios vertės pokyčiai laike yra kaupiami, suteikta teisė operatoriaus užduoto intervalo ribas viršijančius duomenų pokyčius perduoti iš karto, o kitus – po tam tikro laiko.

Vėjo elektrinė 3 lentelėje parodytus duomenis į operatoriaus dispečerinio valdymo punktą turi perduoti pagal pareikalavimą.

Jei dispečerinio valdymo punktai perskaičiuoja ir pakeičia statų ir delsų vertes, jos vėjo elektrinei turi būti siunčiamos automatiškai.

1.4. Vėjo elektrinių parko analizė

Vėjo elektrinės stebimos ir valdomos įvairių išorinių aktorių, pavyzdžiui, lokalių ar nutolusių SCADA (angl. *Supervisory Control And Data Acquisition*) sistemų (pagal informacinių sistemų projektavimo standartus aktoriais vadinami išoriniai naudotojai ar sistemos, sąveikaujančios su nagrinėjama sistema). Šiame kontekste, stebėjimas ir valdymas yra supaprastinamas iki dinaminių komunikavimo procesų, kuriuos charakterizuoja informacijos apsikeitimas tarp aktorių ir konkrečios vėjo elektrinės.

Vėjo elektrinių stebėjimo tikslas yra aprūpinti aktorius informacija apie visą sistema ir jos komponentus. Ši informacija tampa svarbiomis žiniomis vėjo elektrinių valdymui. Pavyzdžiui, SCADA sistema, kuri nori sustabdyti tam tikros vėjo jėgainės darbą, visų pirma turi žinoti kaip ta vėjo jėgainė yra identifikuojama vėjo elektrinių parke ir kokia jos būsena. Turint tokią informaciją jau galima kreiptis į norimą pasiekti komponentą. Taip pat reikia atsakomojo ryšio, kad SCADA sistema žinotų ar valdymo komanda buvo atlikta.

Taigi vėjo elektrinės ir išoriniai aktoriai, norėdami keistis informacija, reikalinga stebėjimui ir valdymui, turi atlikti tai pagal aiškiai apibrėžtus komunikacinius reikalavimus.

1.4.1. Informacijos tipai

Stebėjimo ir valdymo kontekste vykstančios komunikacijos pagrindas yra informacija. Materialus pagrindas yra neapdoroti duomenys, kurie turi būti paverčiami reikšminga informacija. Išskiriami 5 pagrindiniai informacijos tipai, kurie gali būti toliau detalizuojami: 1) proceso informacija; 2) auganti informacija; 3) istorinė informacija; 4) nustatymų informacija; 5) metainformacija.

Proceso, auganti ir istorinė informacija yra reikalinga vėjo elektrinių stebėjimui. Šią informaciją turi teikti vėjo elektrinė. Proceso informacija dažniausiai yra analoginė ir apima informaciją apie bendrą vėjo elektrinės ir jos atskirų komponentų veiklą ir būseną. Auganti informacija yra įvykių skaitliukai, būsenų trukmės ir charakteringi dydžiai (min, max vidurkis, nuokrypis ir pan), padedantys įvertinti vėjo elektrinės veiklą statistiškai. Naudojant istorinę informaciją turi būti įmanoma atsekti elektrinės darbą norimu laikotarpiu, tam panaudojant įvykių žurnalus (angl. *log*) ir ataskaitas (angl. *report*).

Nustatymų informacija naudojama perduoti valdymui reikalingą turinį, tokį kaip nustatytos ribos, parametrai ir komandos. Šią informaciją inicijuoja aktoriai. Vėjo elektrinės tokią informaciją turi saugoti ir pateikti pagal pareikalavimą vėlesniuose komunikavimo procesuose.

Metainformacija padeda orientuotis perduodamuose dydžiuose ir papildo visas aukščiau aptartas informacijos rūšis, priskiriant ir gaunant jų turinį. Metainformacijos pagalba galima sužinoti iš kur kilus informacija, kokiais matavimo vienetais ir kokiu tikslumu ji išreikšta, taip pat laiko žymę bei duomenų tipą ir kokybės deskriptorių.

Pagal tiesioginį atitikimą realiems įrenginio duomenims informaciją galima skirstyti į: šaltinio (proceso ir būsenos) ir išvestą (auganti ir istorinė).

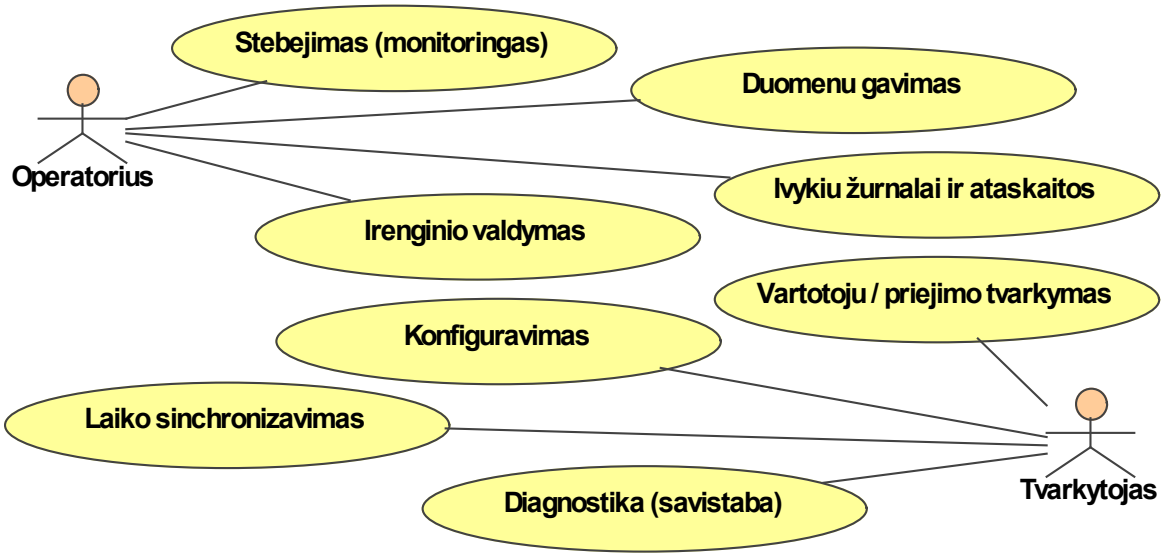
Vėjo elektrinės komponentai su vidine stebėjimo ir valdymo sistema (integruotu valdikliu) pastoviai renka ir saugo duomenis, tokius kaip išmatuoti ir dvejetainiai dydžiai, skaitliukų, taimerių parodymai, įvykių žurnalai, aliarmo signalai, įvykiai, nustatytos ribos ir komandos. Vėjo elektrinių gamintojai teikia priėjimą prie šių duomenų, tačiau jų teikiamos informacijos elementų pavadinimai skirtingi ir operatoriai, stebintys šiuos įrenginius susiduria su semantine problema. Kitas nepatogumas yra tai, kad ši informacija paprastai saugoma ir perduodama linijine struktūra, pvz. lentelių forma, dėl ko atsiranda funkciškai orientuotas priėjimas prie informacijos.

Kaip paminėta įžangoje, tai yra viena pagrindinių priežasčių dėl kurios buvo ieškoma sprendimo, semantiškai standartizuojančio prieinamus duomenis ir sukuriančio komponentišką sistemos vaizdą.

1.4.2. Sistemos vartotojai ir jų funkcijos

Operatorius yra pagrindinis sistemos vartotojas, prižiūrintis vėjo elektrinių parko darbą, valdantis elektrines ir palaikantis išsaugotos informacijos kaupimą.

Tvarkytojas parengia vėjo elektrinę darbui, administruoja jos vartotojus, gauna ataskaitas apie įvykusius nesklandumus.



5 pav. Vėjo elektrinių parko aktoriai ir panaudos atvejai

Aktoriams, prižiūrintiems ir valdantiems vėjo elektrines, reikalingos specialios funkcijos, skirtos konfigūruoti, vykdyti ir stebėti informacijos apsikeitimą su vėjo elektrinėm. Šias funkcijas galima suskirstyti į dvi kategorijas: operacinės funkcijos ir valdymo funkcijos.

Operacinės funkcijos yra reikalingos paprastoms, kasdienėms operacijoms su vėjo elektrinėmis. Jos naudojamos gauti konkrečią vėjo elektrinės informaciją (duomenis) ir nusiųsti atitinkamas valdymo instrukcijas (komandas) vėjo elektrinei.

5 lentelė. Operacinės funkcijos.

Operacinė funkcija	Taikymo sritis (praktinis panaudojimas)
stebėjimas (monitoringas)	vietinis ar nuotolinis būsenų ir būsenų pasikeitimų (indikacijų) stebėjimas
įvykių žurnalas ir ataskaitos	analizavimas, ataskaitų formavimas, įvertinimas
duomenų gavimas	archyvavimas, eksportavimas, duomenų atstatymas
valdymas	pakeitimas ir modifikavimas, įsiterpimas, išjungimas/įjungimas, valdymas, parametrizavimas, optimizavimas

Tvarkymo funkcijos reikalingos aukštesniu lygiu valdyti informacijos apsikeitimą, ir naudojamos apibrėžti bendras stebėjimo ir kontrolės taisykles bei stebėti jų laikymąsi.

6 lentelė. Tvarkymo funkcijos.

Tvarkymo funkcija	Taikymo sritis (praktinis panaudojimas)
vartotojų / priėjimo tvarkymas	vartotojų nustatymas, modifikavimas, naikinimas, priėjimo teisių priskyrimas, prisijungimų stebėjimas
laiko sinchronizavimas	įrenginių sinchronizavimas komunikacijos sistemoje
diagnostika (savistaba)	komunikacinės sistemos diagnostikos nustatymas ir teikimas
konfigūravimas	informacijos apsikeitimo konfigūracijos nustatymas, pakeitimas ir gavimas

1.4.3. Vėjo elektrinių parko ir jo komponentų funkcijos

Žemiau pateikiamos pagrindinės vėjo elektrinių parko funkcijos ir duomenys.

7 lentelė. Vėjo elektrinių parko standartinės funkcijos

Funkcija	Paslaugos ir duomenys
Kontrolės (control)	Sustabdyti vėjo elektrinių parką Paleisti vėjo elektrinių parką Nustatyti galios pareikalavimą Nustatyti galios koeficientą
Stebėjimo ir priežiūros	Aktyvi galia

(monitoring / supervision)	Reaktyvi galia Vidutinis vėjo greitis Vidutinė vėjo kryptis Oro slėgis Oro temperatūra Bendras vėjo elektrinių skaičius Įjungtų vėjo elektrinių skaičius Skaičius dispečerio sustabdytų elektrinių Skaičius operatoriaus sustabdytų elektrinių Nežinomų elektrinių skaičius Sugedusių elektrinių skaičius
Duomenų surinkimas ir gavimas (data collection and retrieval)	10 minučių statistikos saugojimas Skaitliukų parodymų saugojimas Priėjimas prie senų matavimų Gauti būsenos stebėjimo duomenis Gauti laiko ašies duomenis
Veikimas ir raportavimas (performance and reporting)	Prieinamumas Produkcija Prognozė Gauti VE parko nustatymus
Aliarmas ir įvykių valdymas (alarm and event managing)	Nuotolinis įspėjimas apie parko lygio nesklandumus Įvykusių nesklandumų saugojimas
Konfiguracija (configuration)	Pakeisti nustatytus duomenis

Kiekviena vėjo elektrinė galėtų teikti žemiau pateiktas paslaugas ir duomenis.

8 lentelė. Vėjo elektrinės (turbinos) funkcijos

Funkcija	**Paslaugos ir duomenys**
Kontrolės (control)	Paleisti Sustabdyti (dispečerio) Sustabdyti (operatoriaus) Atitaisyti (išvalyti nesklandumus bet nepaleisti) Nustatyti galios pareikalavimą Nustatyti galios koeficientą
Stebėjimo ir priežiūros (monitoring / supervision)	Aktyvi galia Reaktyvi galia RMS įtampa (3 fazės) Rotoriaus greitis Neleidžiamas nuotolinis valdymas (crew present) Vėjo greitis turbinos kaušo aukštyje Turbinos kaušo kryptis
Duomenų surinkimas ir gavimas (data collection and retrieval)	Sukurti ir laikinai saugoti 10 min statistiką Siųsti išsaugotą statistiką
Veikimas ir raportavimas (performance and reporting)	Prieinamumas Produkcijos skaitliukas Platformos pasisukimų skaičius Atsipalaidavimų skaičius Stabdymų skaičius

	Paleidimų skaičius Produkcija Prognozė Gauti VE parko nustatymus
Aliarmas ir įvykių valdymas (alarm and event managing)	Nuotolinis įspėjimas apie parko lygio nesklandumus Įvykusių nesklandumų saugojimas

9 lentelė. Elektros sistemos standartinės funkcijos

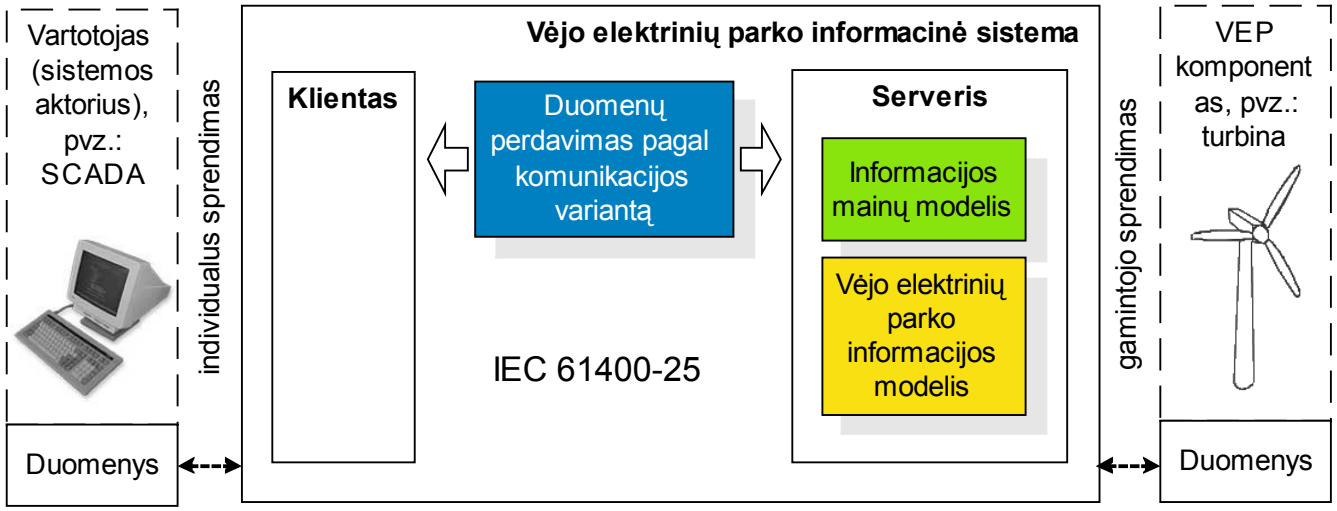
Funkcija	**Paslaugos ir duomenys**
Kontrolės (control)	Atidaryti / uždaryti srovės pertraukiklius
Stebėjimo ir priežiūros (monitoring / supervision)	Fazinės įtampos Fazinės srovės Dažnis Galia Reaktyvi galia
Duomenų surinkimas ir gavimas (data collection and retrieval)	Sukurti ir laikinai saugoti 10 min statistiką Siųsti išsaugotą statistiką Būsenos stebėjimų archyvas
Veikimas ir raportavimas (performance and reporting)	
Aliarmas ir įvykių valdymas (alarm and event managing)	Nuotolinis įspėjimas apie nesklandumus Įvykusių nesklandumų saugojimas Trigeriavimas ir gavimas laiko ašies duomenų įvykus įvykiui
Konfiguracija (configuration)	Pakeisti nustatytus duomenis

10 lentelė. Meteorologinės funkcijos

Funkcija	**Paslaugos ir duomenys**
Kontrolės (control)	Šildymo elementas automatinis / įjungtas / išjungtas
Stebėjimo ir priežiūros (monitoring / supervision)	Vėjo greitis Vėjo kryptis Temperatūra Oro slėgis Lietaus sensoriai Ledo sensoriai Drėgnumo lygis
Duomenų surinkimas ir gavimas (data collection and retrieval)	Sukurti ir laikinai saugoti 10 min statistiką Siųsti išsaugotą statistiką
Veikimas ir raportavimas (performance and reporting)	
Aliarmas ir įvykių valdymas (alarm and event managing)	Nuotolinis įspėjimas apie nesklandumus Įvykusių nesklandumų saugojimas Trigeriavimas ir gavimas laiko ašies duomenų įvykus įvykiui
Konfiguracija (configuration)	Pakeisti nustatytus duomenis

2. Vėjo elektrinių parko informacinės sistemos modeliai

Vėjo elektrinių parko informacinei sistemai patogiausia naudoti kliento–serverio architektūrą, kurios esmė – duomenų mainai komunikaciniu kanalu tarp dviejų nutolusių esybių – serverio ir kliento. IEC 64100-25 standarte siūloma lengvo kliento-serverio architektūra (angl. *thin client*): serveris apdoroja ir saugo visą informaciją, teikia paslaugas ir gali aptarnauti visus į jį besikreipiančius vartotojus, turinčius atitinkamas teises naudotis serveriu.



6 pav. Vėjo elektrinių parko informacinės sistemos koncepcija

Vėjo elektrinių parko komponentų informacijos modelis yra objektinis ir semantiškai standartizuotas. Komponentai modeliuojami kaip informaciniai objektai, identifikuojant visus jų atributus bei funkcionalumą ir sudarant medžio tipo hierarchiją. Kiekvienas atributas turi vardą ir paprastą arba sudėtingą tipą (klasę) ir aprašo konkrečią įrenginio informaciją, kurią galima nuskaityti arba pakeisti.

Informacijos mainų modelis įgalina standartizuotą informacijos apsikeitimą tarp vartotojų ir realaus įrenginio, kurį virtualiai atvaizduoja aukščiau aptartas informacijos modelis. Informacijos mainų modelį sudaro visos serverio teikiamos paslaugos.

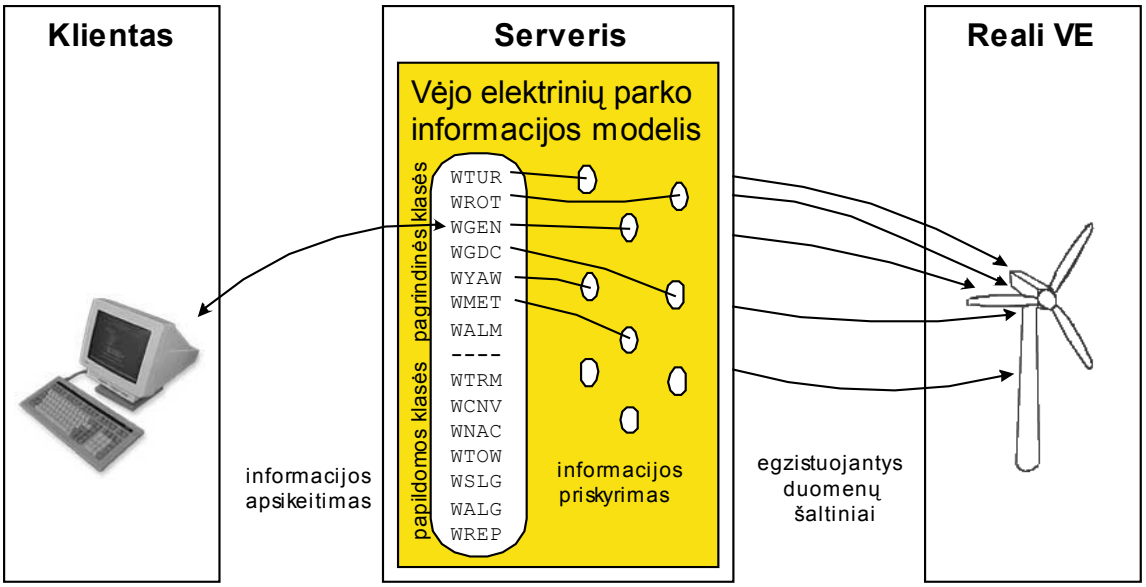
Informacijos ir jos mainų modeliai sudaro sąsają (angl. *interface*) tarp kliento ir serverio. Ši sąsaja leidžia serveriui bendrauti su skirtingais klientais vienu metu, nepriklausomai nuo jų architektūros ar naudojamos programinės įrangos tol, kol jie naudoja serverio palaikomą komunikacijos variantą ir formuoja teisingas užklausas ar valdymo komandas.

Pagal informacijos modelį serverį sudaro vienas arba daugiau loginių įrenginių. Loginiai įrenginiai virtualiai vaizduoja vėjo elektrinę. Loginiai mazgai atitinka sudedamąsias įrenginio dalis, kurios vykdo konkrečią įrenginio funkciją.

2.1. Statinė struktūra - informacijos modelis

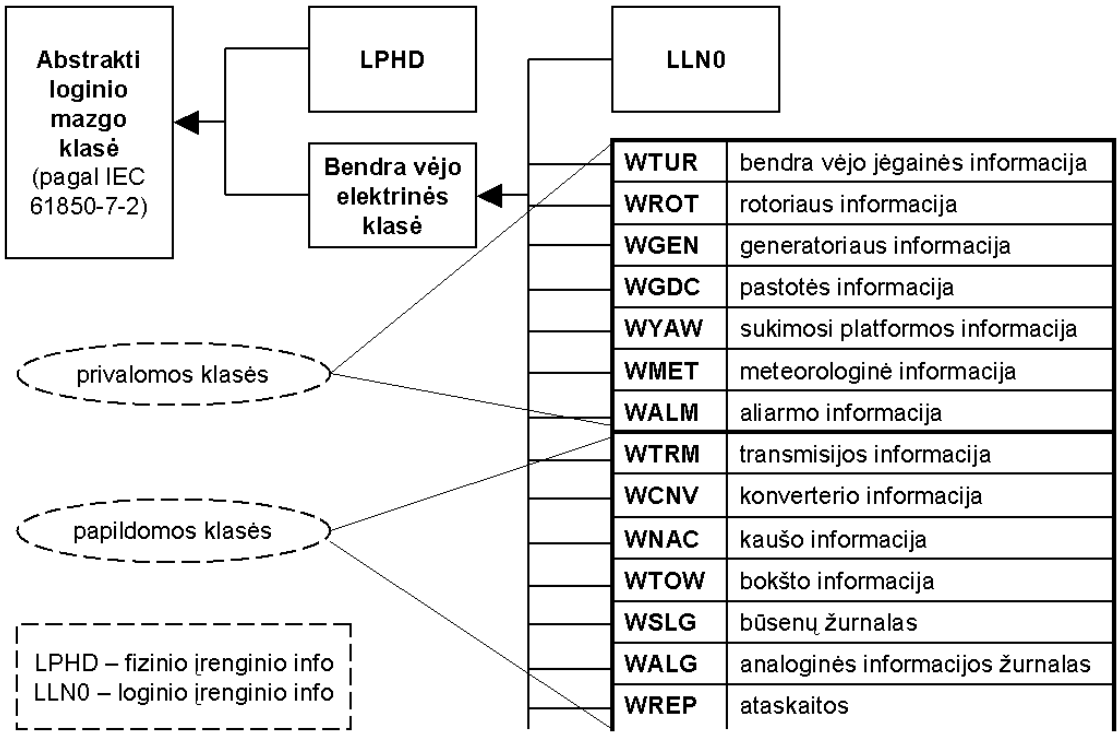
Pagrindinės (privalomos) vėjo elektrinės loginių mazgų klasės: bendra vėjo jėgainės informacija (WTUR); rotoriaus informacija (WROT); generatoriaus informacija (WGEN); pastotės informacija (WGDC); sukimosi platformos informacija (WYAW); meteorologinė informacija (WMET); aliarmo informacija (WALM).

Papildomos (pasirenkamos) vėjo elektrinės loginių mazgų klasės: transmisijos informacija (WTRM); konverterio informacija (WCNV); kaušo informacija (WNAC); bokšto informacija (WTOW); būsenų žurnalas (WSLG); analoginės informacijos žurnalas (WALG); ataskaitos (WREP).



7 pav. Vėjo elektrinėje egzistuojantys loginiai mazgai

Be vėjo elektrinėms specifinių yra dvi privalomos kiekvienam loginiam įrenginiui loginių mazgų klasės: nulinis loginis mazgas (LLN0), teikiantis loginio įrenginio informaciją (vardą ir stovį) ir fizinio įrenginio loginis mazgas (LPHD), teikiantis fizinio įrenginio informaciją (vardą ir stovį).

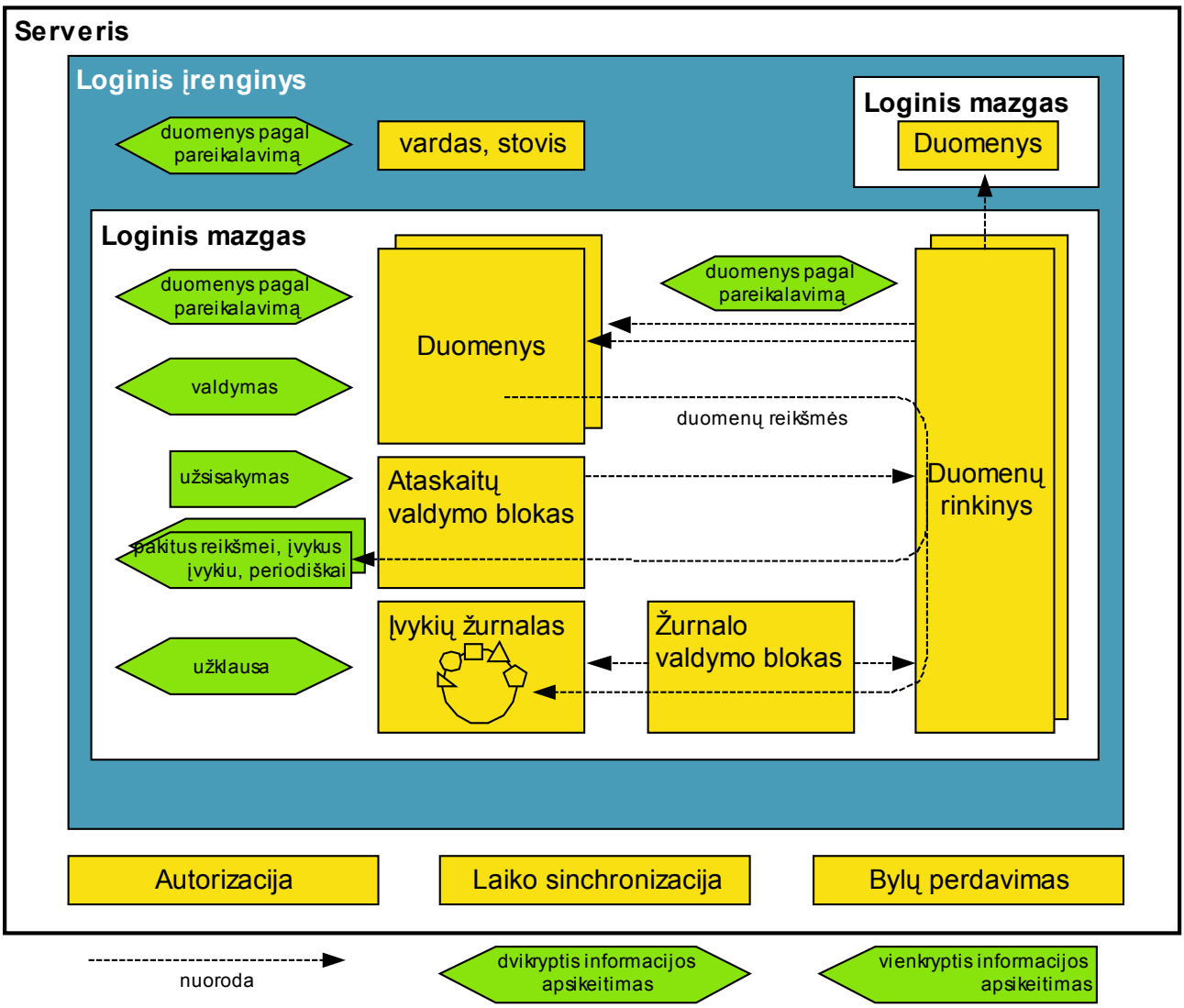


8 pav. Vėjo elektrinės loginių mazgų klasės

2.2. Sistemos elgsena - informacijos mainų modelis

Kiekviename loginiame mazge egzistuoja duomenys (loginio mazgo klasės atributai), kurie gali būti nuskaitomi ir įrašomi individualiai arba grupėmis (duomenų rinkiniais). Be operacijų su duomenimis loginis mazgas reaguoja į valdymo komandas, teikia pavienes arba periodiškas ataskaitas ir pildo įvykių žurnalą, kuris vėliau gali būti peržiūrėtas.

Kiekviename loginiame mazge egzistuoja duomenys (loginio mazgo klasės atributai), kurie gali būti nuskaitomi ir įrašomi individualiai arba grupėmis (duomenų rinkiniais). Be operacijų su duomenimis loginis mazgas reaguoja į valdymo komandas, teikia pavienes arba periodiškas ataskaitas ir pildo įvykių žurnalą, kuris vėliau gali būti peržiūrėtas.



9 pav. Serverio veikimo schema

Proceso (analoginių matavimų) ir būsenos informacija paprastai yra tik nuskaitoma, o valdymo ir konfigūravimo informacija dar ir įrašoma. Dažniausiai naudojamam duomenų rinkiniui klientas gali suteikti savo vardą ir paskui tuo vardu juos nesunkiai pasiekti. Be šios patogios savybės duomenų rinkiniai yra naudojami įvykių žurnalui ir ataskaitoms, kurių formavimo taisyklės realizuotos valdymo blokuose.

Įvykių žurnalai – tai laiko ašyje išrikiuoti atitinkamų duomenų rinkiniai. Jiems pasiekti klientas formuoja filtruotą užklausą, kurioje nurodo norimą laiko periodą ir pageidaujamus duomenis.

Įrenginys gali teikti ataskaitas, pasikeitus reikšmei, įvykus atitinkamam įvykiui arba periodiškai. Nutrūkus ryšiui šios ataskaitos gali būti kaupiamos serveryje ir pateikiamos vėliau.

2.2.1. Serverio teikiamos paslaugos

Operacijos, vykdomos serveryje, turi operacijos užklausimą ir operacijos atsakymą, kuriuos atitinka atskiri pranešimai. Žemiau aptariamos visos serverio operacijos, sudarančios serverio teikiamas paslaugas.

Login. Tai prisijungimo operacija, kurią reikia atlikti pirmiausia, norint gauti duomenis ir naudotis kitomis serverio paslaugomis.

Logoff. Tai atsijungimo operacija, kuria pabaigiama kliento – serverio komunikacijos sesija.

GetServerDirectory. Ši operacija naudojama gauti serverį sudarančių loginių įrenginių vaizdą. Jos atsakymas gražina visų serveryje matomų įrenginių vardus.

2.2.2. Loginio įrenginio ir loginio mazgo paslaugos

Loginis įrenginys – tai loginių mazgų rinkinys. Loginį įrenginį sudarančių loginių mazgų sąrašą teikia **GetLogicalDeviceDirectory** paslauga.

GetLogicalNodeDirectory. Klientas naudoja šią operaciją norėdamas sužinoti kokie loginiai mazgai sudaro nurodytą loginį įrenginį.

2.2.3. Duomenų ir jų rinkinių paslaugos

GetDataDirectory operacija padeda sužinoti kokie atributai matomi duomenų (DATA) elemente.

GetDataDefinition. Duomenų (DATA) elemento struktūrą ir atributų formatą apibūdinanti operacija.

GetDataValues naudojama norint gauti nurodytų duomenų atributų reikšmes.

SetDataValues nustato nurodytų duomenų atributų reikšmes.

CreateDataSet naudojama sukurti duomenų rinkinį (DATA-SET).

DeleteDataSet panaikina nurodyto vardo duomenų rinkinį, jeigu jis buvo sukurtas.

GetDataSetValues ir **SetDataSetValues** veikia analogiškai duomenų nuskaitymui ir nustatymui.

2.2.4. Ataskaitų paslaugos

Apibrėžus duomenų ataskaitų struktūrą, kuri aptariama standarte, su ja susijusios yra šios ataskaitų formavimo operacijos:

- **GetBRCBValues** (angl. *buffered report control block*);
- **SetBRCBValues** (angl. *buffered report control block*);
- **GetURCBValues** (angl. *unbuffered report control block*);
- **SetURCBValues** (angl. *unbuffered report control block*).

Pastaba: ataskaitų siuntimui, serveris naudoja ne SOAP, o HTTP (POST) protokolą, nes taškas per kurį bendrauja klientas yra tik į vieną pusę, arba tam gali būti taikomi asinchroninės komunikacijos metodai.

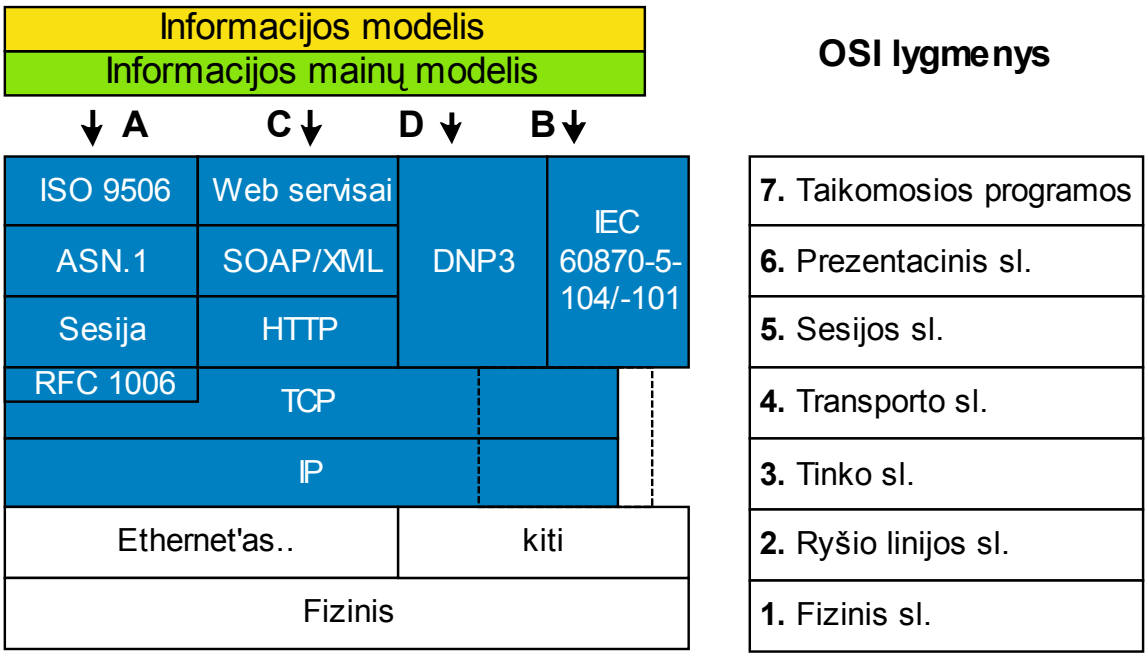
2.2.5. Įvykių žurnalo paslaugos

Įvykių žurnalo bloko (angl. *log control block*) operacijos:

- **GetLCBValues** – gauti įvykių žurnalo bloko fiksuojamus duomenis;
- **GetLCBValues** – nustatyti įvykių žurnalo bloko fiksuojamus duomenis;
- **GetLogStatusValues** – gauti duomenų reikšmes;
- **QueryLogByTime** – gauti reikšmes pagal laiką;QueryLogAfter – gauti reikšmes po nurodyto laiko.

2.3. Sistemos sąsaja - komunikacijos variantai

Informacijos apsikeitimas tarp serverio ir kliento reikalauja vienaprasmiško komunikacinio protokolo abiejose pusėse. Specifinio komunikacijos varianto (šablono) priskyrimas apibrėžia, kaip bus realizuojami vėjo elektrinių parko informacijos modelio objektai, o taip pat funkcijos ir servisai, naudojami informacijos mainų modelyje.



10 pav. Galimi komunikacijos variantai

Priskyrimas komunikacijos variantui remiasi OSI (angl. *Open System Interconnection*) modeliu, kuriame komunikacija tarp kliento ir serverio yra padalinta į 7 sluoksnius, kur 7, 6, ir 5 sluoksniai yra susiję su taikomąja programa, o žemesnieji su transportavimu.

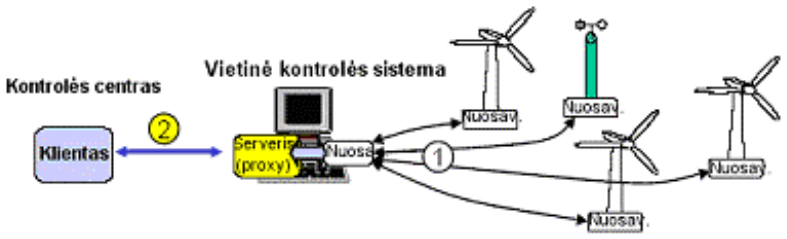
11 lentelė. Komunikacijos variantų palyginimas.

Komunikacijos variantas	Priskyrimo dokumentacija	Pagrindinės charakteristikos
ISO 9506 (MMS)	IEC 61850-8-1 IEC 61400-25 (priedas A)	• visas paslaugas (servisus) susieja su dvejetainiais pranešimais (labai efektyvus kodavimas) • teikia spontaniškas ataskaitas (SoE) • teikia įvykių žurnalo galimybę
IEC 60870-5-101/104	IEC 61400-25 (priedas B)	• įgalina proceso duomenų reikšmių mainus (panaudojant signalus) • įvykių žurnalą galima siųsti panaudojant bylas
Žiniatinklio paslaugos (angl. *web servisai*)	IEC 61400-25 (priedas C) OPC XML-DA	• teikia paslaugų (servisų) poaibį (proceso duomenų reikšmių mainams) naudojant tekstinius (ASCII) pranešimus
	IEC 61400-25 (priedas C) patobulinti web servisai	• teikia papildomas paslaugas (servisus) naudojant SOAP/ASCII • teikia spontaniškas ataskaitas (SoE) • veikia įvykių žurnalas (duomenys kaupiami ir neprisijungus, o vėliau perduodami pagal užklausą)
DNP3	IEC 61400-25 (priedas D)	• įgalina proceso duomenų reikšmių mainus (panaudojant signalus)

Palyginus komunikacijos variantus matome, kad išsamiausiai standartą palaiko žiniatinklio paslaugos, perdavimui naudojant HTTP protokolą.

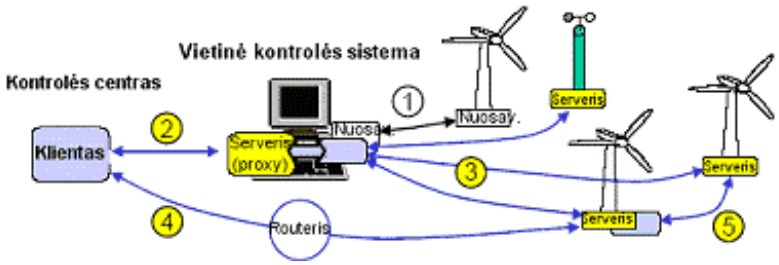
2.4. Komunikacinė architektūra

IEC 64100-25 standarte tikimasi, kad naujos kartos vėjo elektrinių valdikliai jau bus gaminami su standartą palaikančiomis komunikacinėmis sąsajomis ir tokios elektrinės galės nesunkiai būti apjungiamos į bendrą tinklą tinklo šakotuvų (angl. *router*) pagalba. Tačiau šiuo metu egzistuojančioms elektrinėms prie tokio tinklo prijungti teks naudoti jas pagal standartą reprezentuojančius serverius (angl. *proxy, gateway*). Šie serveriai bus tarpininkai tarp standartizuoto tinklo ir nuosavą komunikavimo protokolą turinčių vėjo elektrinių. Pagal serverio pritaikymą numatomi trys pagrindiniai komunikacinės architektūros variantai iliustruojami ir aptariami žemiau.



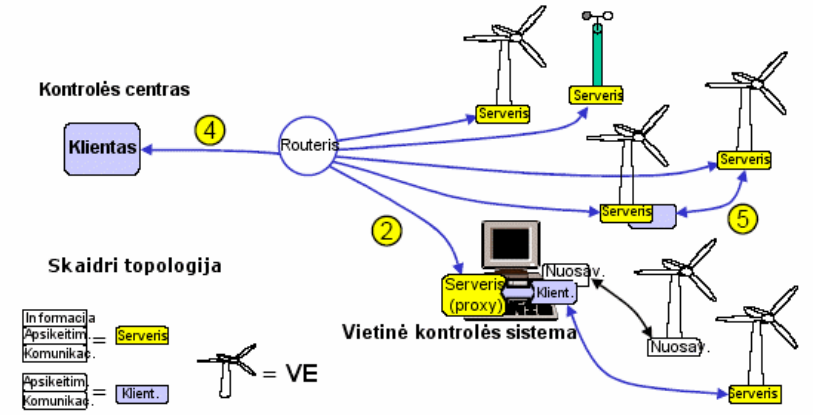
11 pav. Centralizuota topologija

Centralizuoto parko komponentai pasiekiami tik per vietinį valdymo serverį, kai nutolęs klientas komunikuoja (2) su juo. Tolesnė komunikacija tarp serverio ir atskirų parko komponentų (1) realizuojama serveryje nuosavu komunikaciniu protokolu, priklausomai nuo įrenginių valdiklių sąsajos.



12 pav. Maišyta topologija

Kai serveris naudojamas pasiekti įrenginius tiek jų nuosavu protokolu (1), tiek standartiškai (3), gauname maišytą topologiją. Turinčius standartą palaikančią komunikacinę sąsają įrenginius galima pasiekti ir tiesiogiai per šakotuvą (4). Tokie įrenginiai gali komunikuoti tarpusavyje (5).



13 pav. Skaidri topologija

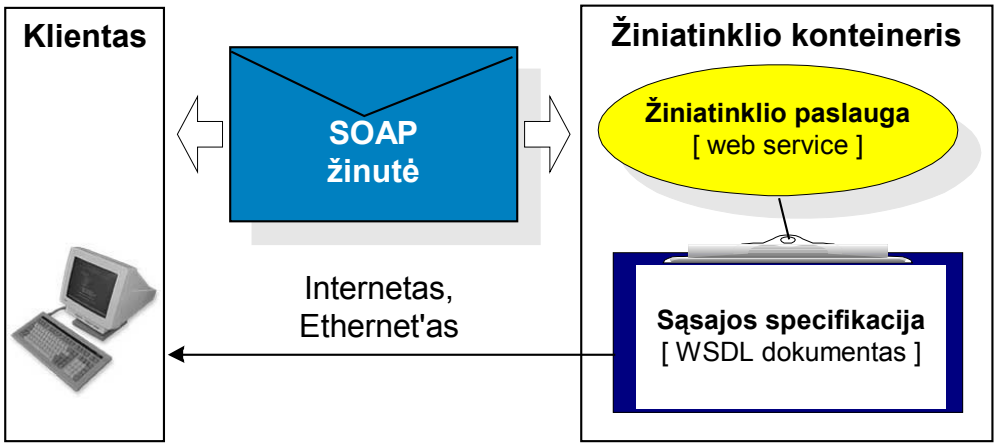
Skaidrioje tinklo topologijoje vėjo elektrinių parko komponentus nutolęs klientas pasiekia per tinklo šakotuvą (4). Kiekvienas atskiras įrenginys turi standartizuotą serverį, prijungtą (2) prie šakotuvo. Jeigu toks parkas jau turėjo savo kontrolės sistemą ir nepalaikančių standarto įrenginių, tai jie tokiu atveju pasiekiami įdiegiant serverį (angl. *proxy*), padedantį komunikuoti su ta elektrine jos nuosavu protokolu. Prie šio serverio galima būtų jungti ir standartizuotus įrenginius.

Visų iki šiol pagamintų vėjo elektrinių duomenis tenka pasiekti pagal jos gamintojo sprendimą. Neišvengiamai tokiems parkams bus reikalingas serveris, kuris tarp išorės ir vėjo elektrinių parko komponentų sudarytų komunikacijos sąsają, atitinkančią standartą. Tokiu atveju tinkamiausia yra centralizuota topologija.

3. Eksperimentinė vėjo elektrinių parko informacinės sistema

3.1. Modelių realizacija ir jai naudojamos technologijos

Realizacijoje vėjo elektrinių parką atitinka serveris, teikiantis žiniatinklio paslaugas (angl. *web services*). Tam pasirinkta SOAP / XML technologija, kurios pagrindas – žiniatinklio paslauga, aprašyta WSDL dokumentu, ir su klientu bendraujanti SOAP pranešimais. Tiek WSDL dokumentai, tiek SOAP pranešimai naudoja XML standartą. Turint išbaigtą WSDL dokumentą, Apache XMLBeans technologijos pagalba generuojamos Java klasės, kurios toliau įgyvendinamos programiškai kuriant servisą arba klientą.



14 pav. Realizacijos koncepcija

Kaip matome 9 pav., žiniatinklio paslaugos sąsaja aprašoma WSDL (angl. *web service definition language*) dokumentu, kurį atitinkamu adresu gali pasiekti kiekvienas, norintis naudotis žiniatinklio paslauga klientas. WSDL dokumentas pilnai specifikuoja informacinės sistemos sąsają: egzistuojančius duomenų tipus, visus įmanomus pranešimus ir atsakymus į juos, bei priskyrimą norimam komunikacijos būdui (HTTP, SOAP, SMTP). Mūsų serveris su klientu bendraus naudodamas SOAP pranešimus (angl. *SOAP envelopes*).

Išbaigtą WSDL nesunkiai galima paversti Java klase, kuri serveryje tampa žiniatinklio paslaugą realizuojančia sąsajos (interfeiso) klase, o kliente – su žiniatinklio paslauga bendraujančio tarpininko (angl. *proxy*) klase, paslepenčia komunikacines subtilybes. Tai esminis realizavimo aspektas.

Žiniatinklio paslaugos sąsają (interface) realizuojanti Java klasė talpinama žiniatinklio konteineryje pasinaudojant WS/XSUL 2.0 (Web and XML Services Utility Library Version 2) biblioteka, o konkrečiau šiais jos moduliais:

- xsul.xservo.XService;
- xsul.xservices_xbeans.XmlBeansBasedService;

- xsul.xservo_soap_http.HttpBasedServices.

Tai vienas paprasčiausių žiniatinklio paslaugų kūrimui skirtų įrankių, panaudojantis XMLBeans pagalba sugeneruotas Java klases ir artimiausia WSDL dokumentui sąsają. WS/XSUL biblioteka vystoma ir sėkmingai naudojama Indianos Universiteto „Extreme! Computing" laboratorijoje [XSUL].

Pradžioje žiniatinklio paslaugas buvo bandoma kurti „Systinet Server for Java" priemonėmis, tačiau buvo susidurta su neadekvačiu WSDL dokumentui duomenų atvaizdavimu ir galimybės valdyti perduodamo SOAP pranešimo XML turinį stoka.

Kiti tyrinėti žiniatinklio konteineriai (angl. *web konteineriai*) Tomcat, JBoss, Apache Axis turi būti instaliuojami atskirai ir tik tada į juos galima įkelti sukurtą žiniatinklio paslaugą. Pasirinktos technologijos leidžią tiek web konteinerio sukūrimą, tiek žiniatinklio paslaugos patalpinimą į jį atlikti programiškai, tuo tarpu minėtiem žiniatinklio konteineriam tą darbą atlieka papildomai sukuriamas įdiegimo aprašymas, kuris kiekvienu atskiru atveju turi būti suformuojamas ir įvykdomas norint įkelti į žiniatinklio paslaugos klases į konteinerį.

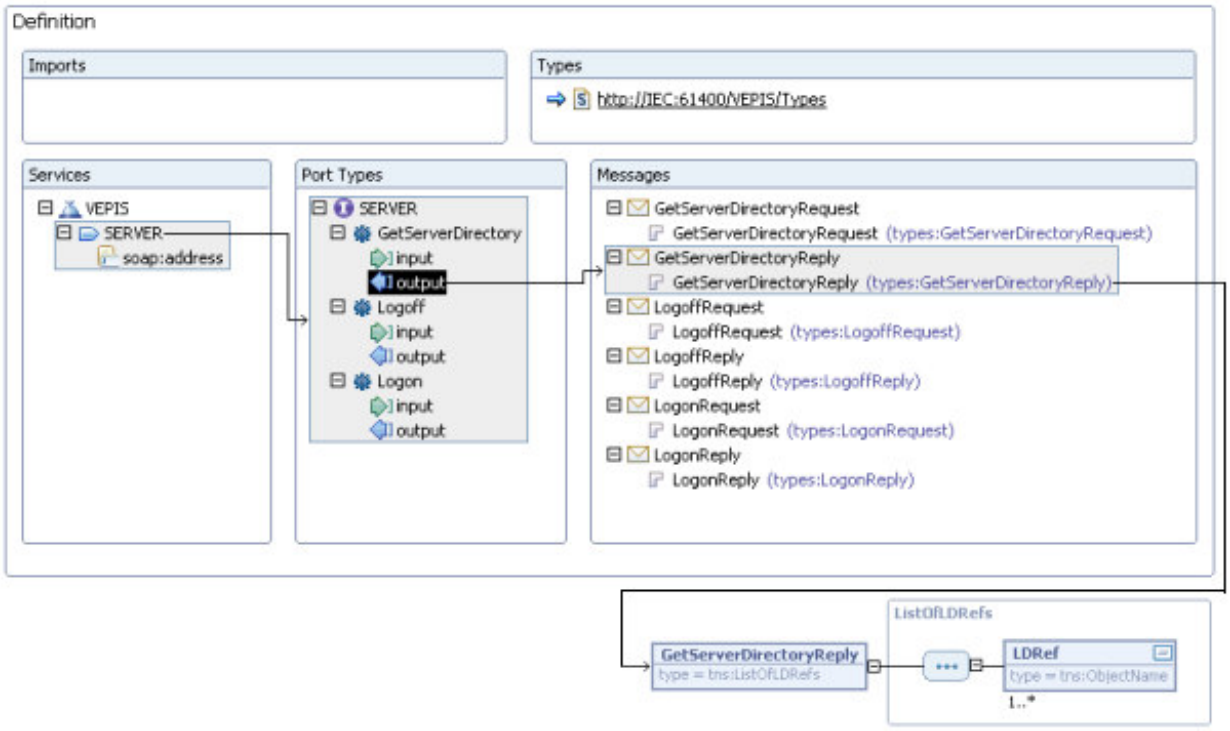
3.2. Žiniatinklio paslaugų sąsajos specifikacija (WSDL)

Žiniatinklio paslaugos sąsają aprašo WSDL dokumentas, kurį sudaro šios dalys:

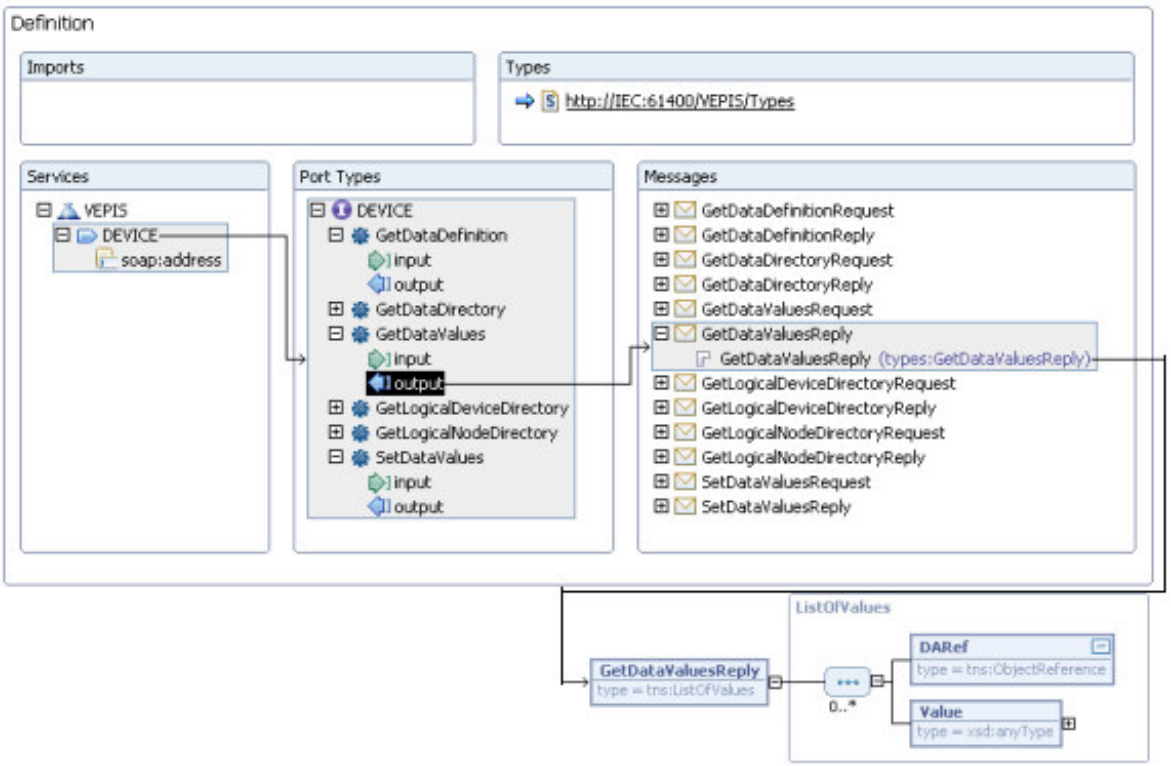
- duomenų tipai (angl. *types*) – pranešimo duomenų įpakavimas, aprašomas XML tipo schema;
- pranešimai (angl. *messages*) – abstraktus, nurodantis naudojamus tipus ir aprašantis perduodamus duomenis pranešimo formatas;
- operacijos (angl. *operations*) – serverio teikiamos paslaugos ir jose naudojami pranešimai;
- taško tipas (angl. *port type*) – galutiniai taškai ir jų palaikomos operacijos, taip pat naudojama pranešimų kryptis;
- priskyrimas (angl. *binding*) – galutiniuose taškuose naudojamas protokolas ir duomenų formatas;
- taškas (angl. *port*) – pagrindinis prieigos taškas ir jame naudojamas priskyrimas;
- servisai (angl. *services*) – susijusių taškų rinkiniai.

Mūsų kuriamam prototipui naudosime du atskirus WSDL dokumentus – serveriui ir loginiam įrenginiui. Šie dokumentai grafiškai atvaizduoti 13 ir 14 paveikslėliuose. Taip pat juose detaliau parodyta po vieną pavyzdinę žinutę: *GetServerDirectoryReply* (*GetServerDirectory* paslaugos atsakymo struktūra) ir *GetDataValuesReply* (*GetDataValues* paslaugos atsakymo struktūra).

WSDL dokumentai originaliu (tekstiniu) formatu yra pateikiami šio darbo 1 ir 2 prieduose.



15 pav. Serverio WSDL dokumento struktūra



16 pav. Loginio įrenginio WSDL dokumento struktūra

3.3. Egzistuojantys tipai

SOAP pranešimuose naudojamų duomenų tipų XML schemos pateikiamos 1 ir 2 prieduose.

Serverio, loginio įrenginio, loginio mazgo ir jo duomenų tipai yra pagrindiniai, statinę vėjo elektrinės dalį vaizduojantys tipai.

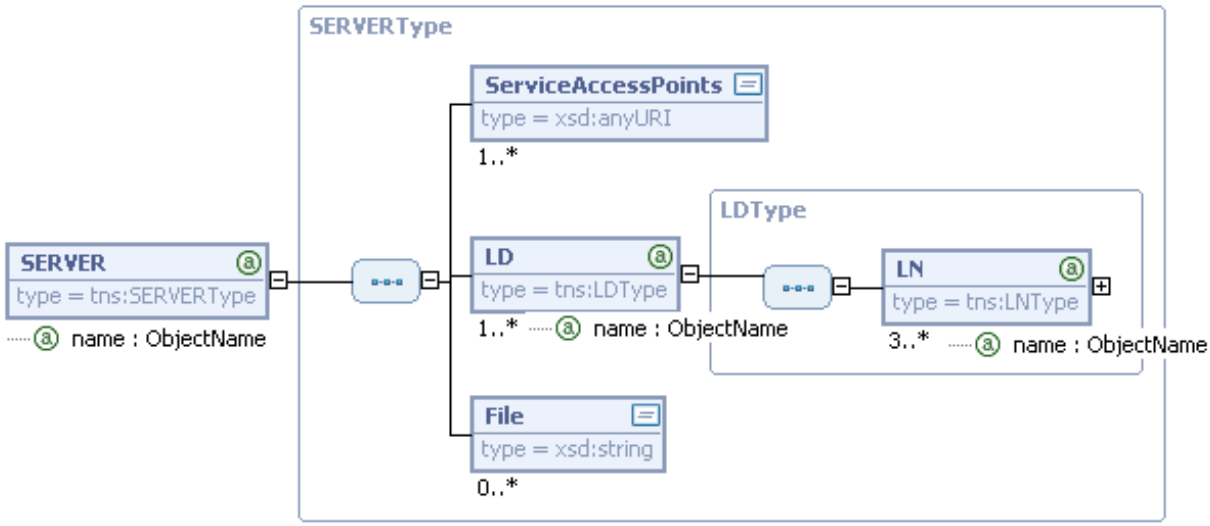
Pranešimuose be pagrindinių egzistuojančių tipų panaudojami įvairūs sąrašai.

ObjectName tipas yra objekto vardas iki 32 simbolių ilgio. *ObjectReference* tipas yra nuoroda į objektą loginiame įrenginyje ir reprezentuoja pilną kelią iki objekto (objektų vardai atskiriami taškais).

3.3.1. Serverio ir loginio įrenginio tipai

Serverio elementas (*SERVER*) turi vardą (*ObjectName* tipo), vieną ar kelis prieigos taškus (*ServerAccessPoints*), vieną arba daugiau loginių įrenginių (*LD*) ir gali turėti parsisiuntimui skirtus failus.

Loginis įrenginys LD yra loginio įrenginio tipo (*LDType*) ir turi vardą (*ObjectName* tipo) bei daugiau nei tris loginius mazgus LN, kurie yra loginio mazgo tipo (*LNType*).



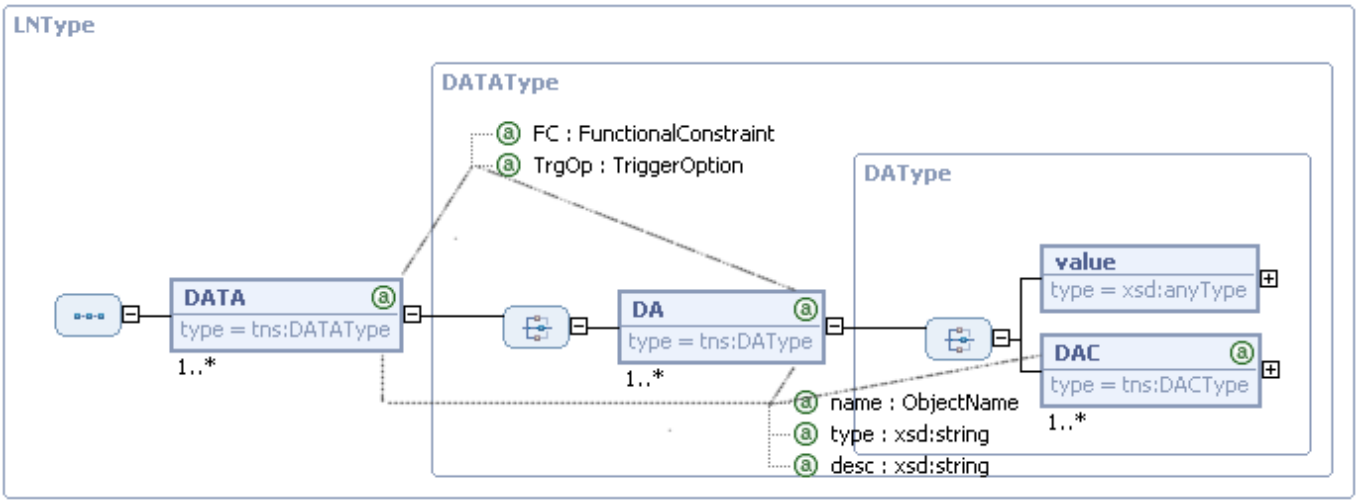
17 pav. Serverio ir loginio įrenginio tipai

3.3.2. Loginis mazgas ir jį sudarantys duomenų tipai

Loginio mazgo tipą sudaro viena arba daugiau duomenų klasių (*DATA*), kurios yra duomenų klasės tipo (*DATAType*). Duomenų klasės tipas (*DATAType*) yra medžio tipo struktūra iš duomenų atributų DA (*DAType* tipo), kurie arba iš karto gali turėti konkrečią bet kokio tipo (*anyType*) reikšmę (*value*), arba apjungti keletą atributo komponentų (*DAC*).

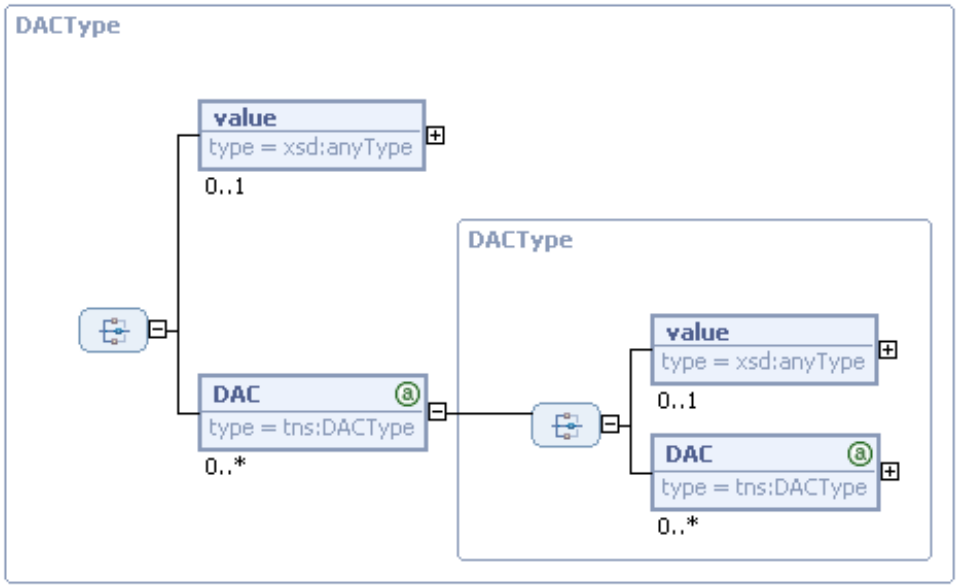
Visos duomenų struktūros (*DATA*, *DA* ir *DAC*) turi vardą (*ObjectName* tipo), tipo pavadinimą (*type*) ir trumpą aprašymą (*desc*). Aukštesnės duomenų struktūros – duomenų klasė (*DATA*) ir duomenų atributas

(*DA*) – papildomai turi funkcionalumą nusakančią žymę (*FC*), pagal kurią galime žinoti kokios paskirties yra duomenys ir trigerio nustatymą (*TrgOp*), panaudojamą įvykių žurnale ir ataskaitose.



18 pav. Loginį mazgą sudarantys duomenų tipai

Jeigu duomenų atributą sudaro keletas semantiškai susijusių duomenų atributo komponentų (*DAC*), tai jie, savo ruožtu, gali įgyti konkrečią bet kokio tipo (*anyType*) reikšmę (*value*) arba toliau šakotis į gylį.



19 pav. Duomenų komponento (DAC) tipas

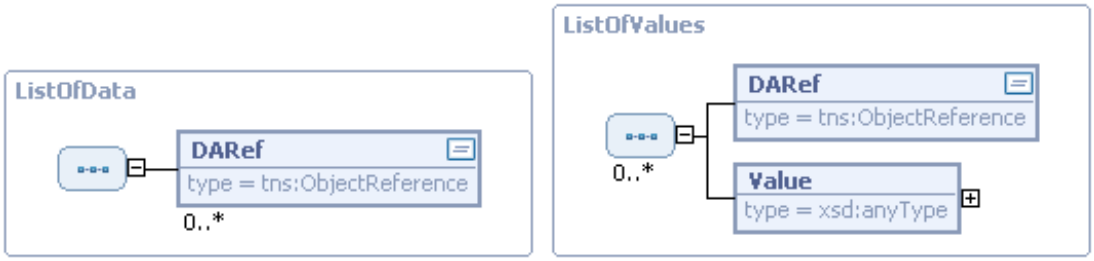
3.3.3. Papildomi (įvairių sąrašų) tipai

Serverio ir loginio įrenginio paslaugos teikiamos konkrečiomis operacijomis, kurios turi apibrėžtus pranešimų tipus. Pranešimuose, be aukščiau aptartų pagrindinių tipų, gali egzistuoti pagalbiniai tipai. Šiuo atveju tai įvairūs nuorodų sąrašai.

Perduodant loginių įrenginių pavadinimų sąrašą naudojamas *ListOfLDRefs* tipas, o perduodant loginių mazgų pavadinimų sąrašą – *ListOfLNRefs* tipas.

20 pav. Loginių įrenginių ir loginių mazgų sąrašai

Duomenų perdavime naudojami duomenų pavadinimų sąrašas *ListOfData* ir duomenų reikšmių sąrašas *ListOfValues*.



21 pav. Duomenų pavadinimų ir duomenų reikšmių sąrašai

3.4. Serverio operacijos ir žinutės

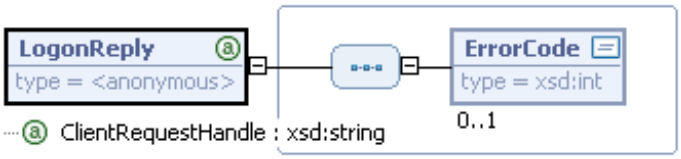
Operacijos, vykdomos serveryje, turi operacijos užklausimą ir operacijos atsakymą, kuriuos atitinka atskiri pranešimai. Žemiau detalizuojamos visos serverio operacijos, sudarančios serverio teikiamas paslaugas.

3.4.1. Logon

Tai prisijungimo prie serverio operacija, naudojama pradėti darbą su serveriu.

22 pav. LogonRequest žinutės struktūra

Užklausos žinutę sudaro šie privalomi atributai: vartotojo vardas (*UserName*) ir slaptažodis (*Password*). Neprivalomi atributai: kalba (Language) ir kliento rodyklė (*ClientRequestHandle*).



23 pav. LogonReply žinutės struktūra

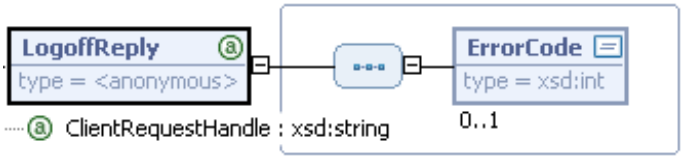
Atsakymo žinutė nepavykus prisijungti grąžina klaidos kodą (*ErrorCode*). Jeigu užklausoje buvo perduota kliento rodyklė, ji grąžinama tokia pat. Kliento rodyklė gali būti panaudota iš to paties adreso kreipiantis keliems skirtingiems klientams, kad konkretus klientas galėtų žinoti ar atsakymas yra jam.

3.4.2. Logoff

Tai atsijungimo nuo serverio operacija, naudojama baigti darbą su serveriu.

24 pav. LogoffRequest žinutės struktūra

Čia vėl, kaip ir prisijungimo (*Logon*) operacijoje, gali būti panaudojama kliento rodyklė (*ClientRequestHandle*) ir grąžinamas klaidos kodas (*ErrorCode*) nesėkmės atveju.



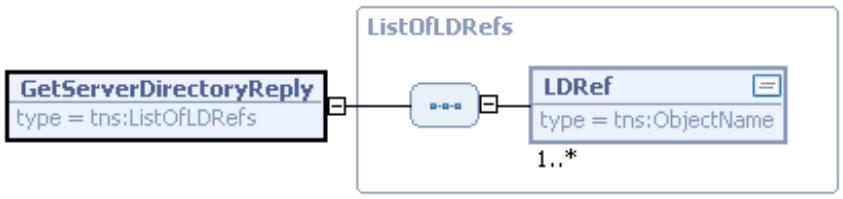
25 pav. LogoffReply žinutės struktūra

3.4.3. GetServerDirectory

Ši operacija naudojama gauti serverį sudarančių loginių įrenginių pavadinimų sąrašą.

26 pav. GetServerDirectoryRequest žinutės struktūra

Užklausos žinutė paprastai siunčiama tuščia. Atsakymo žinutė grąžina loginių įrenginių sąrašą, kurį sudaro vienas arba daugiau loginių įrenginių pavadinimų (*LDRef*).



27 pav. GetServerDirectoryReply žinutės struktūra

3.5. Loginio įrenginio ir loginio mazgo operacijos ir žinutės

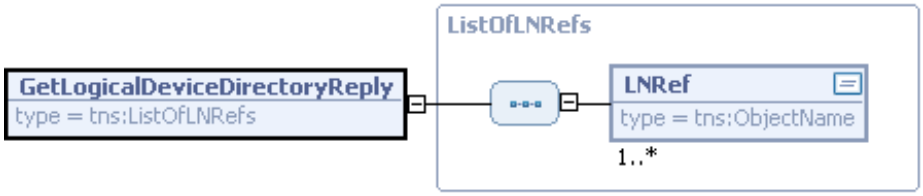
Operacijos, vykdomos loginiame įrenginyje, turi operacijos užklausimą ir operacijos atsakymą, kuriuos atitinka atskiri pranešimai. Žemiau detalizuojamos visos loginio įrenginio operacijos, sudarančios loginio įrenginio teikiamas paslaugas.

3.5.1. GetLogicalDeviceDirectory

Ši operacija naudojama sužinoti, kokie loginiai mazgai sudaro loginį įrenginį.

28 pav. GetLogicalDeviceDirectoryRequest žinutės struktūra

Užklausos žinutė paprastai siunčiama tuščia. Atsakymo žinutė grąžina loginių mazgų sąrašą, kurį sudaro vienas arba daugiau loginių mazgų pavadinimų (*LNRef*).



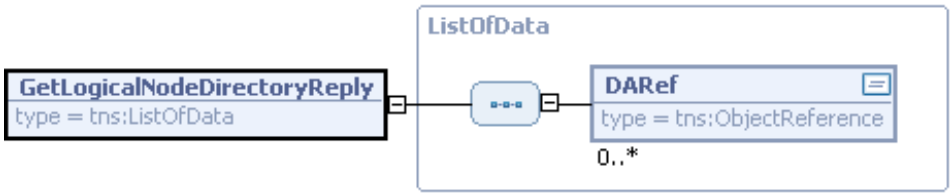
29 pav. GetLogicalDeviceDirectoryReply žinutės struktūra

3.5.2. GetLogicalNodeDirectory

Tai operacija, skirta sužinoti, kokios duomenų klasės (*DATA*) sudaro loginį mazgą.

30 pav. GetLogicalNodeDirectoryRequest žinutės struktūra

Užklausos žinute nurodomas mus dominančio loginio mazgo pavadinimas. Atsakymo žinutė grąžina loginį mazgą sudarančių duomenų klasių nuorodas (pilnus kelius iki objekto) *DARef*.



31 pav. GetLogicalNodeDirectoryReply žinutės struktūra

3.6. Duomenų operacijos ir žinutės

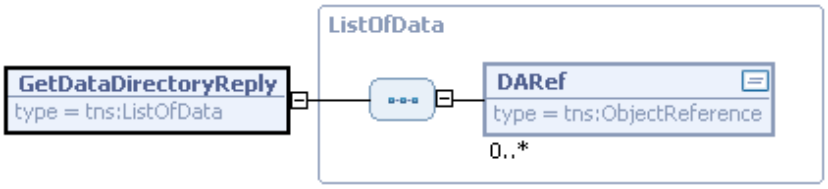
Operacijos su duomenimis priskiriamos prie loginio įrenginio teikiamų paslaugų.

Kiekvienai iš jų formuojant užklausos žinutę reikia *ObjectReference* tipo nuorodos (pilno kelio iki duomenų objekto). Atsakymo žinutėmis grąžinamos taip pat *ObjectReference* tipo nuorodos.

3.6.1. GetDataDirectory

Ši operacija naudojama sužinoti konkrečią duomenų klasę sudarančius atributus.

32 pav. GetDataDirectoryRequest žinutės struktūra

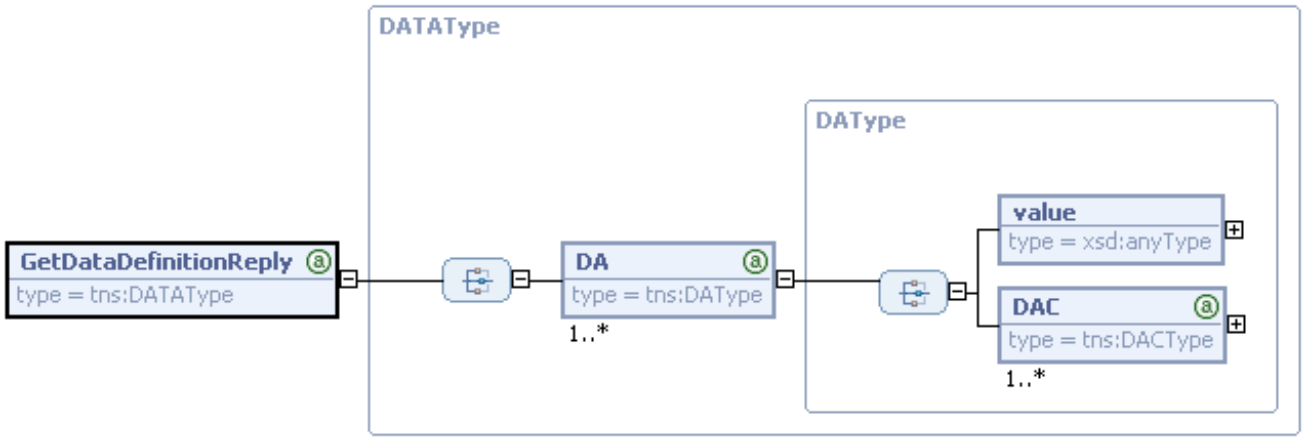


33 pav. GetDataDirectoryReply žinutės struktūra

3.6.2. GetDataDefinition

Tai nurodytos duomenų klasės struktūrą nusakanti operacija. Atsakymo žinute grąžinamas visas duomenų klasės atributų ir jų komponentų medis su reikšmių (value) tipais.

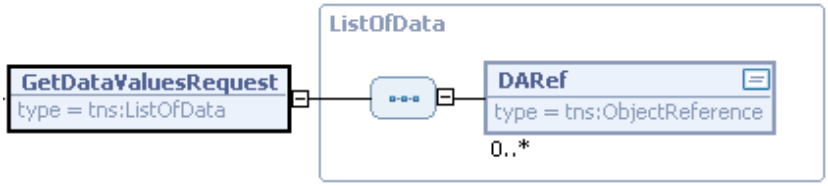
34 pav. GetDataDefinitionRequest žinutės struktūra



35 pav. GetDataDefinitionReply žinutės struktūra

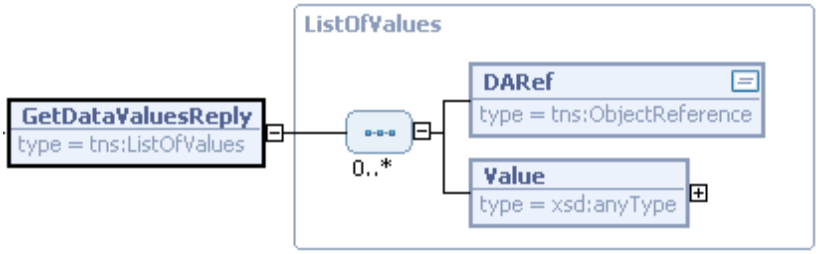
3.6.3. GetDataValues

Operacija, skirta gauti užklausoje nurodyto duomenų atributų sąrašo *DARef* reikšmes.



36 pav. GetDataValuesRequest žinutės struktūra

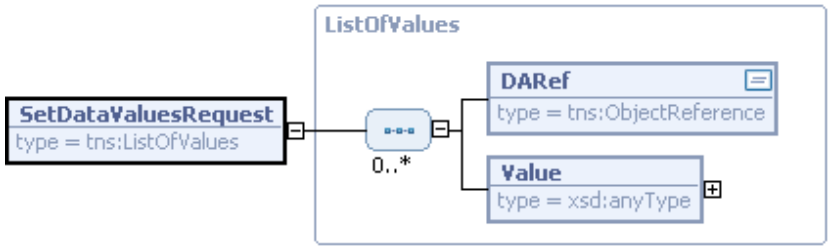
Atsakymo žinute grąžinamas užklausoje nurodytas *DARef* duomenų atributų sąrašas su įterpta po kiekvienu *DARef* atributu jo reikšme (*Value*).



37 pav. GetDataValuesReply žinutės struktūra

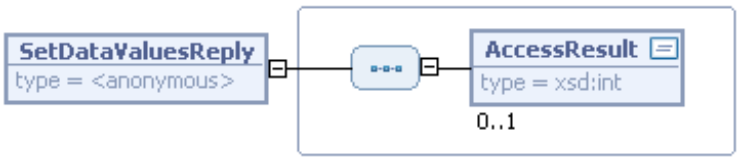
3.6.4. SetDataValues

Operacija, skirta nustatyti duomenų atributų sąrašo *DARef* reikšmes. Užklausos žinutėje iš kart po kiekvieno norimo nustatyti duomenų atributo elementu *Value* nurodoma norima nustatyti reikšmė.



38 pav. SetDataValuesRequest žinutės struktūra

Atsakymo žinutė grąžina skaitmeninį rezultatą (*AccessResult*) apie operacijos sėkmingumą, kurio skaičius parodo kiek duomenų atributų nustatyta sėkmingai. Operacijai nustatinėjant duomenų atributus, jeigu atributas randamas ir reikšmę pavyksta pakeisti, rezultato skaičius padidinamas vienetu, jeigu ne – sumažinamas.



39 pav. SetDataValuesReply žinutės struktūra

3.7. VEPIS prototipas ir darbas su juo

Apache XMLBeans generuoja Java klases, paveldinčias patogias darbui su XML *org.apache.xmlbeans.XmlObject* savybes ir enkapsuliuojančias kiekvieną XML schemos elementą arba tipą. Realizacijoje XMLBeans klasių pagalba užkraunami loginių įrenginių duomenys iš .xml failų.

Pagal serverio ir loginio įrenginio WSDL dokumentus (pateiktus 1 ir 2 prieduose) Apache XMLBeans pagalba sugeneravus visus duomenų tipus ir elementus atitinkančias Java klases, o panaudojant WS/XSUL2 bibliotekos *xwsdlc* įrankį – Java sąsajos (angl. *interface*) klases, belieka jas realizuoti ir išmėginti.

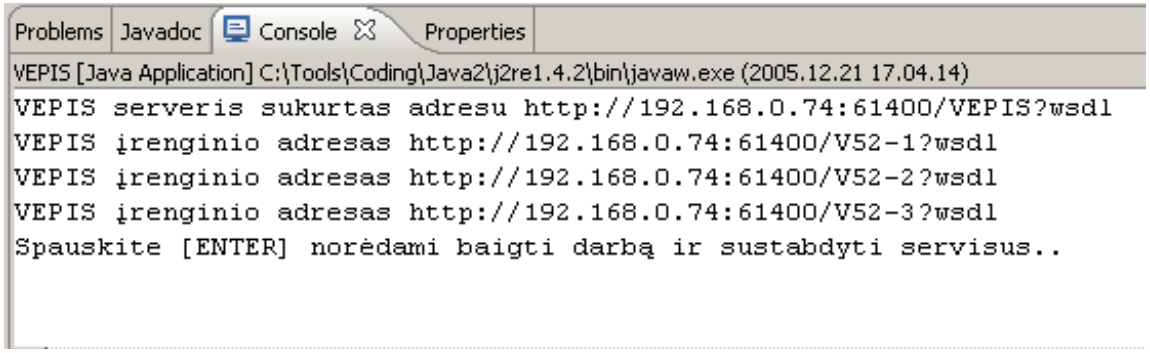
Realizacijai naudojama **Eclipse 3.1** programavimo aplinka (angl. *IDE – integrated development enviroment*) – daugiafunkcinė Java kūrimo platforma. Serveris kuriamas programiškai realizuojant sugeneruotas serverio ir loginio įrenginio Java sąsajos klases. Klientas realizuojamas pasinaudojant Eclipse žiniatinklių paslaugų testavimo įrankiu – Web Services Explorer.

Serverio realizacijoje pažymėtinas XPath užklausų panaudojimas duomenų atributų paieškai. XPath (W3C rekomendacija nuo 1999 lapkričio) – tai paprasta užklausų / išraiškų kalba XML'ui, palengvinanti semantinę navigaciją XML struktūrose.

Serveris veikia ir atskirai nuo Eclipse aplinkos, tačiau klientas yra mūsų prototipui pritaikytas (modifikuotas) Eclipse įrankis, todėl kolkas žemiau aprašomui darbui su serveriu neišvengiamai reikalingas Eclipse programų paketas su įdiegtu WTP (*Web Tools Platform*) įskiepiu.

3.7.1. Serverio ir kliento paleidimas

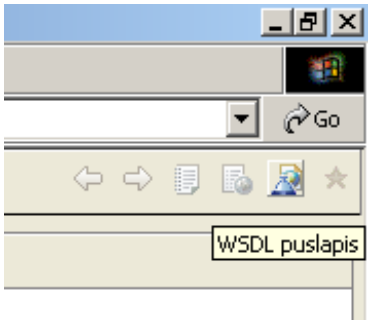
Eclipse aplinkoje atidarome projektą VEPISProject ir jame surandame pagrindinį viso serverio failą VEPIS.java (*iec61400.vepis* pakete) ir jį paleidžiame meniu komanda *Run -> Run As -> Java Application*.



40 pav. Sėkmingai paleisto serverio konsolė

Klientas paleidžiamas meniu komanda *Run -> Launch the Web Services Explorer*.

Atsidaro tuščias kliento langas, kurio viršutiniame dešiniame kampe esančiu priešpaskutiniu mygtuku perjungiam *WSDL puslapio* režimą.

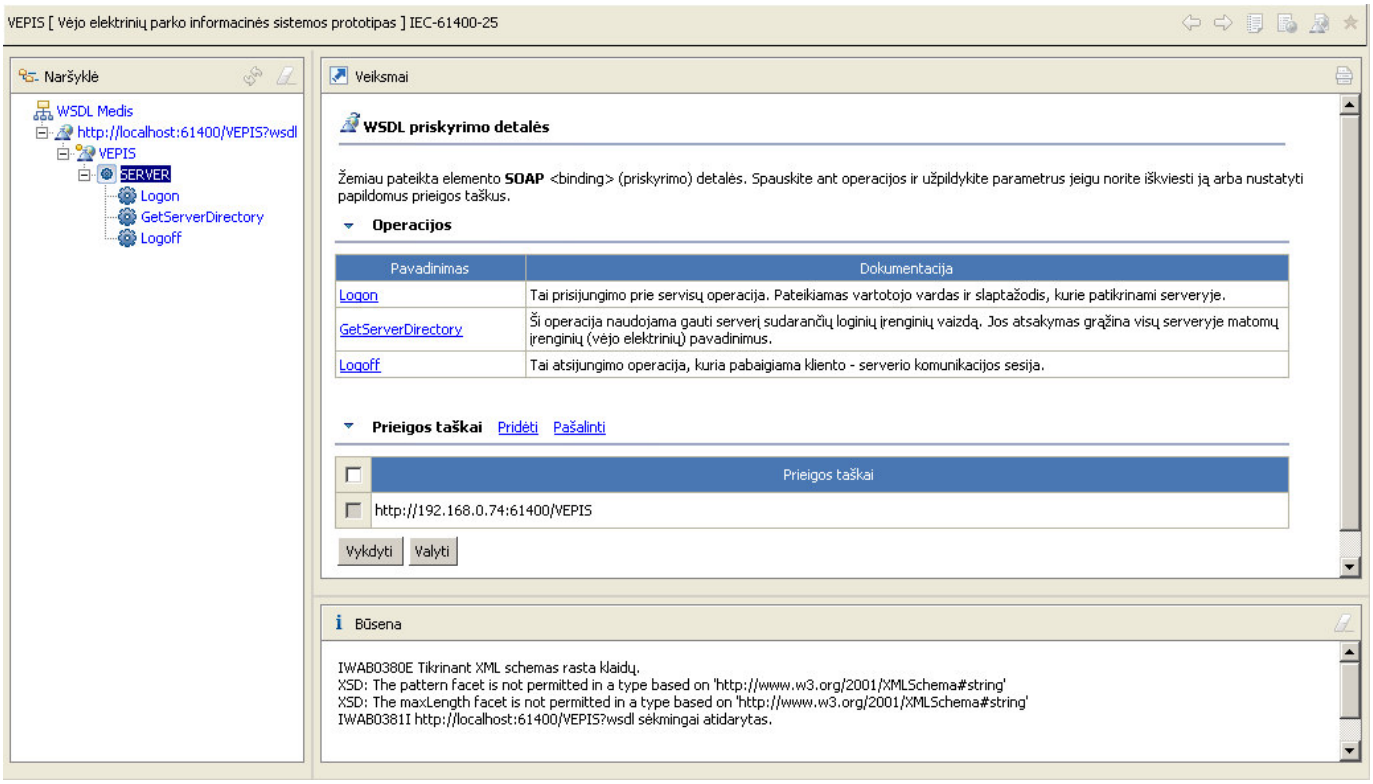


41 pav. WSDL puslapio režimo pasirinkimas kliente

Kairėje esančiame lange *Naršyklė* paspaudžiame ant *WSDL medis* užrašo ir lange *Veiksmai* suvedame serverio pateiktą *WSDL URL*: http://192.168.0.74:61400/VEPIS?wsdl ir spaudžiam *Vykdyti*.

Kadangi IP adresai kompiuteriuose skiriasi, o klientas greičiausiai bus paleidžiamas tame pačiame kompiuteryje kaip ir serveris, tai nurodant *WSDL URL* adresą vietoje čia naudojamo ‘192.168.0.74’ gali būti patogiau naudoti ‘localhost’ arba jo skaitmeninį atitinkenį ‘127.0.0.1’.

Sėkmingai pasiekus serverio žiniatinklio paslaugą, adresas įrašomas į *Naršyklės* lango *WSDL medį*, o lange *Veiksmai* pamatome *WSDL priskyrimo detales*.



42 pav. Sėkmingas prisijungimas prie VEPIS serverio

3.7.2. Serveryje esančių loginių įrenginių radimas

Norėdami toliau dirbti su serveriu, turime išsiaiškinti kokie loginiai įrenginiai (vėjo elektrinės) egzistuoja jame. Tam turime įvykdyti jo *GetServerDirectory* operaciją, kurios rezultatas duoda loginių įrenginių pavadinimus.

Operacijos vykdymo langas atidaromas paspaudus ant jos lange *Naršyklė* arba lange *Veiksmai*. Kadangi ši operacija nereikalauja jokių parametrų (užklausa gali būti tuščia), tai tiesiog spaudžiame mygtuką *Vykdyti* ir apačioje (žemiau veiksmų langa) esančiame lange *Būsena* gauname atsakymą.

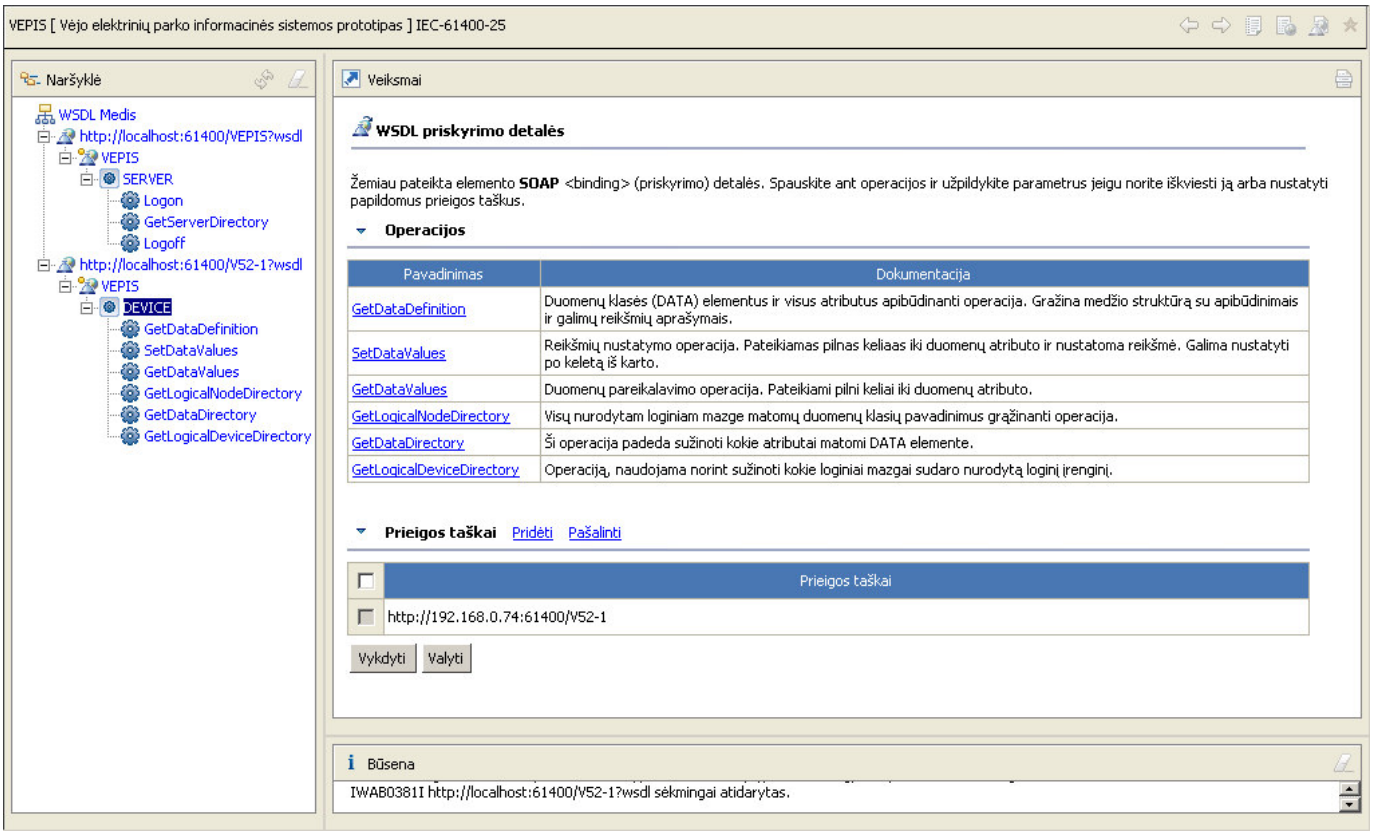
43 pav. Serveryje egzistuojantys loginiai įrenginiai

Serveryje veikiančius loginius įrenginius ir jų adresus taip pat galime pažiūrėti serverio konsolėje.

3.7.3. Prisijungimas prie loginio įrenginio

Norėdami dirbti su konkrečia vėjo elektrine, turime prie jos prisijungti tokiu pat būdu kaip prisijungėme prie serverio, tik vietoje serverio vardo VEPIS turime įrašyti vėjo elektrinės vardą.

Kairėje esančiame lange *Naršyklė* paspaudžiame ant *WSDL medis* užrašo ir lange *Veiksmai* suvedame norimo pasiekti loginio įrenginio *WSDL URL*: http://localhost:61400/V52-1?wsdl ir įvykdome.



44 pav. Sėkmingas prisijungimas prie V52-1 loginio įrenginio

3.7.4. Loginiame įrenginyje esančių loginių mazgų radimas

Taigi prisijungę prie pasirinktos vėjo elektrinės (mūsų atveju V52-1) pirmiausia mes norime sužinoti kokie mazgai jame egzistuoja. Tam atidarome loginio įrenginio paslaugą *GetLogicalDeviceDirectory*.

Operacijos *GetLogicalDeviceDirectory* vykdymo langas atidaromas paspaudus ant jos lange *Naršyklė* arba lange *Veiksmai*. Kadangi ši operacija nereikalauja jokių parametrų (užklausa gali būti tuščia), tai tiesiog spaudžiame mygtuką *Vykdyti* ir žiūrime į apačioje (žemiau veiksmų lango) esantį langą *Būsena*.

45 pav. Loginiame įrenginyje egzistuojantys loginiai mazgai

Matome, kad vėjo elektrinę "V52-1" sudaro šie loginiai mazgai: LLN0, LPHD, WTUR, WROT, WGEN.

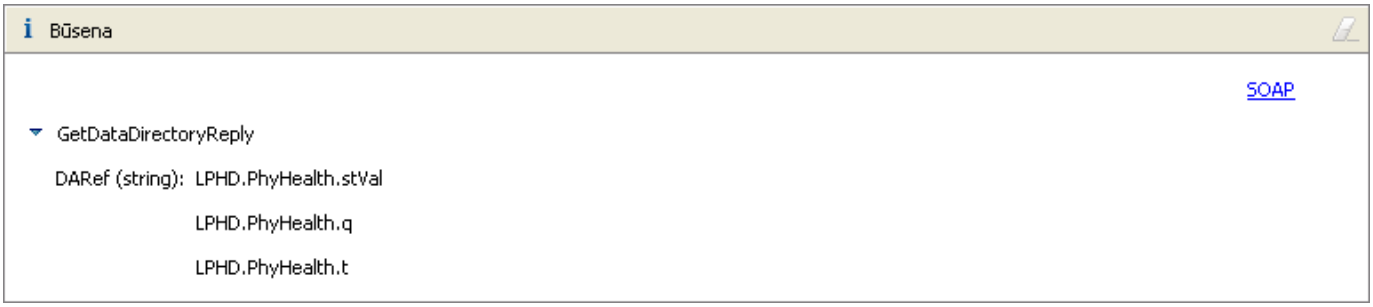
3.7.5. Loginį mazgą sudarančių duomenų klasių radimas

Pasirenkame mus dominantį vėjo elektrinės 'V52-1' loginį mazgą 'LPHD', kuriame, kaip žinome iš standarto, saugoma loginio įrenginio info. Norėdami sužinoti šiame loginiame mazge egzistuojančias duomenų klases, naudojamės *GetLogicalNodeDirectory* paslauga, kuriai kaip parametrą perduodame loginio mazgo pavadinimą 'LPHD'. Rezultate matome visas šio mazgo duomenų klases.

46 pav. Vėjo elektrinės V52-1 loginiame mazge LPHD egzistuojančios duomenų klasės

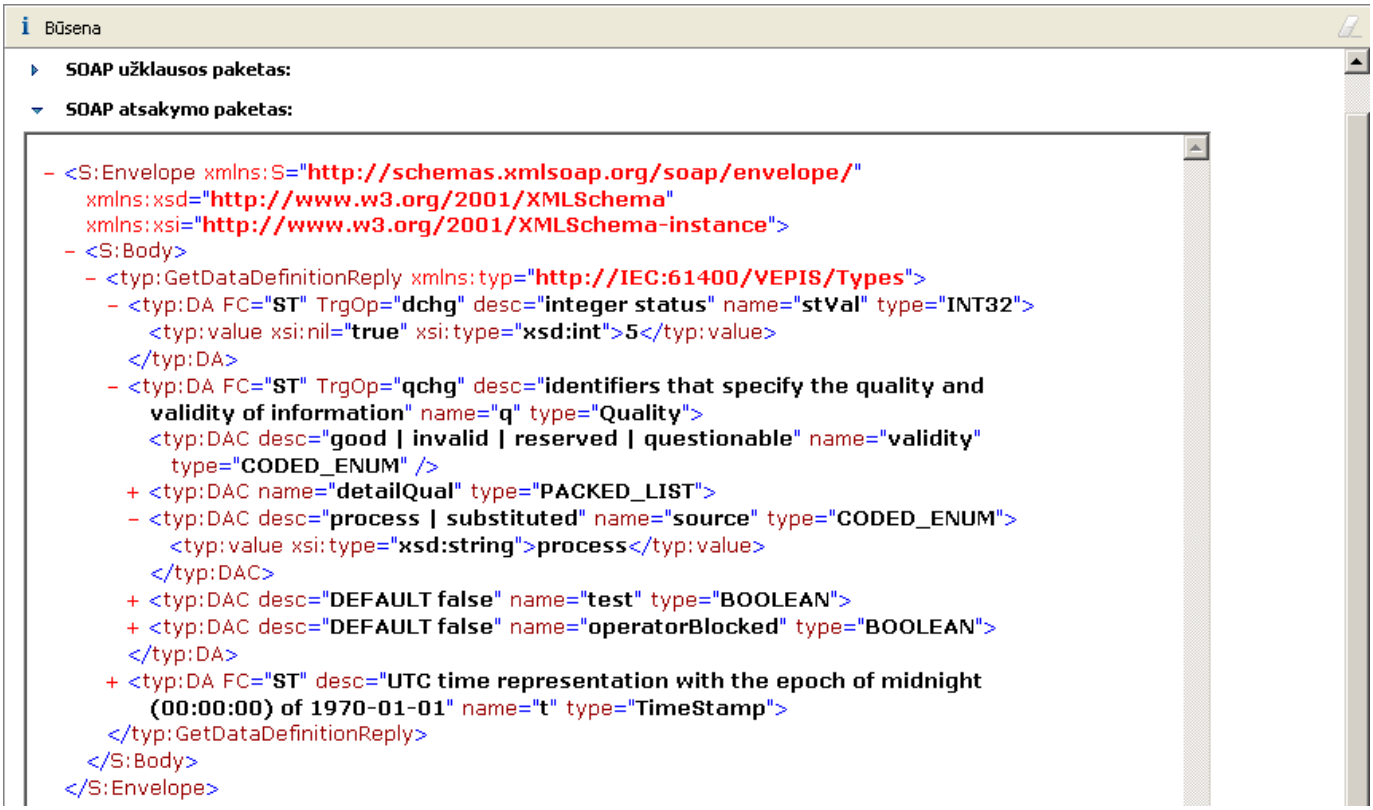
3.7.6. Duomenų klasės atributų sąrašo ir struktūros radimas

Dominančios duomenų klasės atributus randame paslaugos *GetDataDirectory* paslaugos pagalba. Šios paslaugos užklausoje įrašome mus dominančią duomenų klasę (pilną kelią iki duomenų klasės) *LPHD.PhyHealth* ir gauname ją sudarančių atributų sąrašą (irgi su pilnais keliais iki atributo).



47 pav. Vėjo elektrinės V52-1 duomenų klasėje LPHD.PhyHealth egzistuojančios duomenų klasės

Sudėtingesnį rezultatą grąžina paslauga *GetDataDefinition*, kuriai taip pat nurodoma tik pilnas kelias iki mus dominančios duomenų klasės (šiuo atveju *LPHD.PhyHealth*). Norėdami pamatyti rezultatą XML formate lango *Būsena* dešiniajame viršutiniame kampe spaudžiame SOAP.

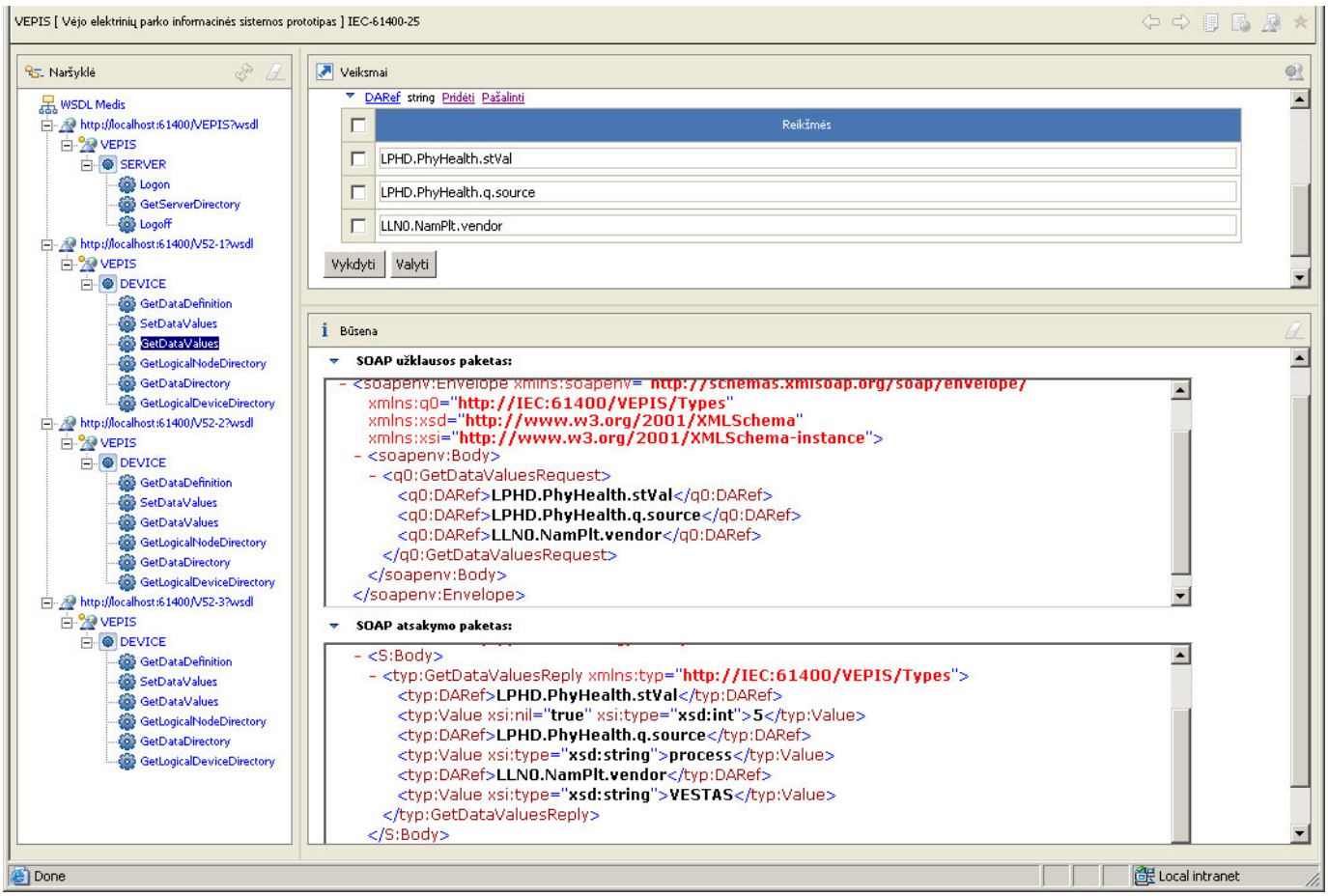


48 pav. Vėjo elektrinės V52-1 duomenų klasės LPHD.PhyHealth struktūra

3.7.7. Duomenų gavimas ir nustatymas

Duomenys gaunami *GetDataValues* užklausoje duomenų atributus, kurių reikšmę norime sužinoti surašant į sąrašą, kurio elementai *DARef*.

GetDataValues atsakymo žinutė grąžina tokį pat sąrašą tik su po kiekvienu DARef įterpta duomenų atributo reikšme (*Value*).



49 pav. Vėjo elektrinės V52-1 duomenų atributų gavimas

Duomenys nustatomi *SetDataValues* paslaugai užklausą suformuojant taip pat kaip *GetDataValues* atsakymo žinutės. *SetDataValues* grąžina skaitmeninį rezultatą, pagal kurį galime žinoti kiek duomenų atributų reikšmių pavyko sėkmingai nustatyti.

IŠVADOS

IEC 64100-25 yra tarptautinis standartas tiriamoje srityje ir neprieštarauja vėjo elektrinių prijungimo prie Lietuvos elektros energetikos sistemos techninėms taisyklėms. Atsižvelgiant į tai ir numatant tolimesnį vėjo energetikos augimą, kuris ateityje gali pareikalauti atitikimo Europos standartams, šis tiriamasis darbas atliktas orientuojantis į IEC 64100-25.

IEC 64100-25 standartas apibrėžia kaip modeliuoti vėjo elektrinės duomenis, paslaugas ir kokiu formatu perduoti informaciją, naudojant pasirinktą komunikacijos variantą. Standartas buvo kuriamas numatant, kad serveris, suformuojantis komunikavimo sąsają, galėtų būti pagamintas kartu su vėjo elektrine.

Visų iki šiol pagamintų vėjo elektrinių duomenis tenka pasiekti pagal jos gamintojo sprendimą. Neišvengiamai tokiems parkams eksploatuoti bus reikalingas serveris, kuris tarp išorės ir vėjo elektrinių parko komponentų sudarytų komunikacijos sąsają, atitinkančią standartą. Tokiu atveju tinkamiausia komunikacinė architektūra yra centralizuotos topologijos.

Palyginus egzistuojančius komunikacinius variantus matome, kad išsamiausiai standartą palaiko ir daugiausia galimybių teikia žiniatinklio paslaugų (angl. *web services)* SOAP / XML pagrindu technologija, perdavimui naudojant HTTP protokolą.

Aptarus pagrindinius sistemos modelius ir pasirinkus komunikacinį variantą, WSDL dokumentais specifikuojama sąsaja tarp serverio ir kliento. Šios sąsajos programiška realizacija ir sudaro darbe kuriamą eksperimentinį prototipą.

Vėjo elektrinių parko komponentų informacijos modelis projektuojamas pasirenkant standartizuotas duomenų klases ir nesunkiai gali būti išplėstas naujomis duomenų klasėmis (aprašant jas XML dokumentu, sukurtu pagal WSDL dokumente pateiktą schemą ir duomenų atributams suteikiant pavadinimus pagal standarte naudojamą sintaksę ir semantiką).

Prototipas gali būti toliau vystomas ir panaudojamas kaip pagrindas kuriant IEC 64100-25 standartą atitinkančias informacines sistemas. Autorius tikisi, kad šis darbas ateityje pravers tokių sistemų kūrėjams.

LITERATŪRA

1. [VEPT] AB “Lietuvos energija”. Vėjo elektrinių prijungimo prie Lietuvos elektros energetikos sistemos techninės taisyklės, 2003 10 16, elektroninis variantas.

2. [VESTAS] Firmos “VESTAS” gaminių techniniai duomenys, V52-850 kW Pitch regulated wind turbine with OptiTip® and OptiSpeed™, 2003 01 04.

3. [IEC25]. IEC. IEC 61400-25, Wind Turbine Generator Systems - Part 25: Communications for monitoring and control of wind power plants, 1st committee draft (final to IEC), 2003 07 04.

4. [ANEM] Habil. dr. A. Nemura, Vėjo elektrinių modeliavimo ir jų įtakos elektros sistemos darbui studija, Autorinis darbas (sutartis Nr. A21-124.3.4..), 2004 01 30.

5. [ANDR] Lietuvos elektros energetikos sistemos papildomų išlaidų, prijungus vėjo jėgaines, įvertinimas. Galutinė ataskaita . Dr. R. Andruškevičius. Sistemų valdymo ir automatizavimo laboratorija

6. [IEEE] Automatic Generation Control of a Wind Farm With Variable Speed Wind Turbines, IEEE TRANSACTIONS ON ENERGY CONVERSION, VOL. 17, NO. 2, JUNE 2002

7. [CONT] A comparison of control concepts for wind turbines in terms of energy capture, Rolf Hofjmann, Technischen Universitat Darmstadt, 1970 July 31

8. [1000MW] Experiences with over 1000 MW wind power installed in Germany, elektroninis variantas (pdf formatas)

9. [3000MW] Practical Experience with 3000 MW Wind Power Installed in Germany, elektroninis variantas (pdf formatas)

10. [XSUL] WS/XSUL2: Web and XML Services Utility Library (Version 2), Aleksander Slominski, Extreme! Computing Lab, High Performance Distributed & Parallel Systems Research, Department of Computer Science, Indiana University, www.extreme.indiana.edu/xgws/xsul/, 2006 01 07

11. [NEVAR] ,,Vėjo energetikos įtaka elektros sistemų režimams ir jų valdymui”, E. V. Nevardauskas, D. Šulga, Lietuvos MA Technikos mokslų skyriaus susirinkimas – konferencija ,,Informacinių ir valdymo technologijų taikymas elektros energetikoje”, LEI, 2005 m. birželio 10 d.

12. [VEPIS1] ,,Vėjo elektrinių parko informacinės sistemos prototipas”, E. Vaičiukynas, A. Nemura, tarptautinė konferencija ,,Automatika ir valdymo technologijos – 2005”, KTU, 2005 m. gegužės 12 d.

13. [VEPIS2] ,,Vėjo elektrinių parko informacinės sistemos architektūra ir komunikacijos aspektai”, E. Vaičiukynas, A. Nemura, Lietuvos MA Technikos mokslų skyriaus susirinkimas – konferencija ,,Informacinių ir valdymo technologijų taikymas elektros energetikoje”, LEI, 2005 m. birželio 10 d.

1 PRIEDAS. Serverio WSDL dokumentas (SERVER.wsdl)

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="VEPIS" targetNamespace="http://IEC:61400/VEPIS"
xmlns:types="http://IEC:61400/VEPIS/Types"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:vepis="http://IEC:61400/VEPIS">

  <wsdl:types>
    <xsd:schema targetNamespace="http://IEC:61400/VEPIS/Types"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:tns="http://IEC:61400/VEPIS/Types" elementFormDefault="qualified">

  <xsd:complexType name="ListOfLDRefs">
    <xsd:sequence>
      <xsd:element name="LDRef" type="tns:ObjectName" minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:simpleType name="ObjectName">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[\w]*" />
      <xsd:maxLength value="32" />
    </xsd:restriction>
  </xsd:simpleType>

      <xsd:element name="GetServerDirectoryRequest" type="xsd:string" />

      <xsd:element name="GetServerDirectoryReply" type="tns:ListOfLDRefs" />

      <xsd:element name="LogonRequest">
        <xsd:complexType>
          <xsd:attribute name="UserName" type="xsd:string" use="required" />
          <xsd:attribute name="Password" type="xsd:string" use="required" />
          <xsd:attribute name="Language" type="xsd:string" use="optional" />
          <xsd:attribute name="ClientRequestHandle" type="xsd:string" use="optional" />
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="LogonReply">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="ErrorCode" type="xsd:int" minOccurs="0" maxOccurs="1" />
          </xsd:sequence>
          <xsd:attribute name="ClientRequestHandle" type="xsd:string" use="optional" />
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="LogoffRequest">
        <xsd:complexType>
          <xsd:attribute name="ClientRequestHandle" type="xsd:string" use="optional" />
        </xsd:complexType>
      </xsd:element>

      <xsd:element name="LogoffReply">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="ErrorCode" type="xsd:int" minOccurs="0" maxOccurs="1" />
          </xsd:sequence>
          <xsd:attribute name="ClientRequestHandle" type="xsd:string" use="optional" />
        </xsd:complexType>
      </xsd:element>

    </xsd:schema>
  </wsdl:types>

  <wsdl:message name="GetServerDirectoryReply">
    <wsdl:part name="GetServerDirectoryReply" element="types:GetServerDirectoryReply" />
  </wsdl:message>
  <wsdl:message name="GetServerDirectoryRequest">
    <wsdl:part name="GetServerDirectoryRequest" element="types:GetServerDirectoryRequest" />
  </wsdl:message>
```

```
  <wsdl:message name="LogoffReply">
    <wsdl:part name="LogoffReply" element="types:LogoffReply" />
  </wsdl:message>
  <wsdl:message name="LogoffRequest">
    <wsdl:part name="LogoffRequest" element="types:LogoffRequest" />
  </wsdl:message>
  <wsdl:message name="LogonReply">
    <wsdl:part name="LogonReply" element="types:LogonReply" />
  </wsdl:message>
  <wsdl:message name="LogonRequest">
    <wsdl:part name="LogonRequest" element="types:LogonRequest" />
  </wsdl:message>

  <wsdl:portType name="SERVER">
    <wsdl:operation name="GetServerDirectory">
      <wsdl:documentation>Ši operacija naudojama gauti serverį sudarančių loginių įrenginių vaizdą. Jos
atsakymas grąžina visų serveryje matomų įrenginių (vėjo elektrinių) pavadinimus.</wsdl:documentation>
      <wsdl:input message="vepis:GetServerDirectoryRequest" />
      <wsdl:output message="vepis:GetServerDirectoryReply" />
    </wsdl:operation>
    <wsdl:operation name="Logoff">
      <wsdl:documentation> Tai atsijungimo operacija, kuria pabaigiama kliento – serverio komunikacijos
sesija.</wsdl:documentation>
      <wsdl:input message="vepis:LogoffRequest" />
      <wsdl:output message="vepis:LogoffReply" />
    </wsdl:operation>
    <wsdl:operation name="Logon">
      <wsdl:documentation> Tai prisijungimo prie servisų operacija. Pateikiamas vartotojo vardas ir
slaptažodis, kurie patikrinami serveryje.</wsdl:documentation>
      <wsdl:input message="vepis:LogonRequest" />
      <wsdl:output message="vepis:LogonReply" />
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="SERVER" type="vepis:SERVER">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="Logon">
      <soap:operation soapAction="http://localhost:61400/VEPIS/Logon" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetServerDirectory">
      <soap:operation soapAction="http://localhost:61400/VEPIS/GetServerDirectory" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="Logoff">
      <soap:operation soapAction="http://localhost:61400/VEPIS/Logoff" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="VEPIS">
    <wsdl:port name="SERVER" binding="vepis:SERVER">
      <soap:address location="http://localhost:61400/VEPIS" />
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

2 PRIEDAS. Loginio įrenginio WSDL dokumentas (DEVICE.wsdl)

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="VEPIS" targetNamespace="http://IEC:61400/VEPIS"
xmlns:types="http://IEC:61400/VEPIS/Types"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:vepis="http://IEC:61400/VEPIS">

  <wsdl:types>
    <xsd:schema targetNamespace="http://IEC:61400/VEPIS/Types"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:tns="http://IEC:61400/VEPIS/Types" elementFormDefault="qualified">

  <xsd:complexType name="SERVERType">
    <xsd:sequence>
      <xsd:element name="ServiceAccessPoints" type="xsd:anyURI" minOccurs="1" maxOccurs="unbounded" />
      <xsd:element name="LD" type="tns:LDType" minOccurs="1" maxOccurs="unbounded" />
      <xsd:element name="File" type="xsd:string" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="name" type="tns:ObjectName" default="IEC61400" />
  </xsd:complexType>

  <xsd:complexType name="LDType">
    <xsd:sequence>
      <xsd:element name="LN" type="tns:LNType" minOccurs="3" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="name" type="tns:ObjectName" />
  </xsd:complexType>

  <xsd:complexType name="LNType">
    <xsd:sequence>
      <xsd:element name="DATA" type="tns:DATAType" minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="name" type="tns:ObjectName" />
  </xsd:complexType>

  <xsd:complexType name="CDCBasic" abstract="true">
    <xsd:attribute name="name" type="tns:ObjectName" />
    <xsd:attribute name="type" type="xsd:string" use="optional" />
    <xsd:attribute name="desc" type="xsd:string" use="optional" />
  </xsd:complexType>

  <xsd:complexType name="DACType">
    <xsd:complexContent>
      <xsd:extension base="tns:CDCBasic">
        <xsd:choice>
          <xsd:element name="value" type="xsd:anyType" minOccurs="0" maxOccurs="1" nillable="true" />
          <xsd:element name="DAC" type="tns:DACType" minOccurs="0" maxOccurs="unbounded" />
        </xsd:choice>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="DAType">
    <xsd:complexContent>
      <xsd:extension base="tns:CDCBasic">
        <xsd:choice>
          <xsd:element name="value" type="xsd:anyType" maxOccurs="1" nillable="true" />
          <xsd:element name="DAC" type="tns:DACType" maxOccurs="unbounded" />
        </xsd:choice>
        <xsd:attribute name="FC" type="tns:FunctionalConstraint" use="optional" />
        <xsd:attribute name="TrgOp" type="tns:TriggerOption" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="DATAType">
    <xsd:complexContent>
      <xsd:extension base="tns:CDCBasic">
        <xsd:choice>
          <xsd:element name="DA" type="tns:DAType" maxOccurs="unbounded" />
        </xsd:choice>
        <xsd:attribute name="FC" type="tns:FunctionalConstraint" use="optional" />
```

```
      <xsd:attribute name="TrgOp" type="tns:TriggerOption" use="optional" />
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="ObjectName">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[\w]*" />
    <xsd:maxLength value="32" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ObjectReference">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[\w]*/[\w.]*" />
    <xsd:maxLength value="255" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="FunctionalConstraint">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="ST" />
    <xsd:enumeration value="MX" />
    <xsd:enumeration value="CO" />
    <xsd:enumeration value="SP" />
    <xsd:enumeration value="CF" />
    <xsd:enumeration value="DC" />
    <xsd:length value="2" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="TriggerOption">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="dchg" />
    <xsd:enumeration value="qchg" />
    <xsd:enumeration value="dupd" />
    <xsd:length value="4" />
  </xsd:restriction>
</xsd:simpleType>

<xsd:element name="SERVER" type="tns:SERVERType" />

<xsd:element name="LD" type="tns:LDType" />

<xsd:element name="LN" type="tns:LNType" />

<xsd:element name="DATA" type="tns:DATAType" />

<xsd:element name="DA" type="tns:DAType" />

<xsd:element name="DAC" type="tns:DACType" />

<xsd:complexType name="ListOfLDRefs">
  <xsd:sequence>
    <xsd:element name="LDRef" type="tns:ObjectName" minOccurs="1" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ListOfLNRefs">
  <xsd:sequence>
    <xsd:element name="LNRef" type="tns:ObjectName" minOccurs="1" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ListOfData">
  <xsd:sequence>
    <xsd:element name="DARef" type="tns:ObjectReference" minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ListOfValues">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="DARef" type="tns:ObjectReference" />
    <xsd:element name="Value" type="xsd:anyType" />
```

```
  </xsd:sequence>
</xsd:complexType>

    <xsd:element name="GetLogicalDeviceDirectoryRequest" type="xsd:string" />

    <xsd:element name="GetLogicalDeviceDirectoryReply" type="tns:ListOfLNRefs" />

    <xsd:element name="GetLogicalNodeDirectoryRequest" type="tns:ObjectName" />

    <xsd:element name="GetLogicalNodeDirectoryReply" type="tns:ListOfData" />

    <xsd:element name="GetDataDirectoryRequest" type="tns:ObjectReference" />

    <xsd:element name="GetDataDirectoryReply" type="tns:ListOfData" />

    <xsd:element name="GetDataDefinitionRequest" type="tns:ObjectReference" />

    <xsd:element name="GetDataDefinitionReply" type="tns:DATAType" />

    <xsd:element name="GetDataValuesRequest" type="tns:ListOfData" />

    <xsd:element name="GetDataValuesReply" type="tns:ListOfValues" />

    <xsd:element name="SetDataValuesRequest" type="tns:ListOfValues" />

    <xsd:element name="SetDataValuesReply">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="AccessResult" type="xsd:int" minOccurs="0" maxOccurs="1" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

  </xsd:schema>
</wsdl:types>

<wsdl:message name="GetDataDefinitionReply">
  <wsdl:part name="GetDataDefinitionReply" element="types:GetDataDefinitionReply" />
</wsdl:message>
<wsdl:message name="GetDataDefinitionRequest">
  <wsdl:part name="GetDataDefinitionRequest" element="types:GetDataDefinitionRequest" />
</wsdl:message>
<wsdl:message name="GetDataDirectoryReply">
  <wsdl:part name="GetDataDirectoryReply" element="types:GetDataDirectoryReply" />
</wsdl:message>
<wsdl:message name="GetDataDirectoryRequest">
  <wsdl:part name="GetDataDirectoryRequest" element="types:GetDataDirectoryRequest" />
</wsdl:message>
<wsdl:message name="GetDataValuesReply">
  <wsdl:part name="GetDataValuesReply" element="types:GetDataValuesReply" />
</wsdl:message>
<wsdl:message name="GetDataValuesRequest">
  <wsdl:part name="GetDataValuesRequest" element="types:GetDataValuesRequest" />
</wsdl:message>
<wsdl:message name="GetLogicalDeviceDirectoryReply">
  <wsdl:part name="GetLogicalDeviceDirectoryReply" element="types:GetLogicalDeviceDirectoryReply" />
</wsdl:message>
<wsdl:message name="GetLogicalDeviceDirectoryRequest">
  <wsdl:part name="GetLogicalDeviceDirectoryRequest" element="types:GetLogicalDeviceDirectoryRequest" />
</wsdl:message>
<wsdl:message name="GetLogicalNodeDirectoryReply">
  <wsdl:part name="GetLogicalNodeDirectoryReply" element="types:GetLogicalNodeDirectoryReply" />
</wsdl:message>
<wsdl:message name="GetLogicalNodeDirectoryRequest">
  <wsdl:part name="GetLogicalNodeDirectoryRequest" element="types:GetLogicalNodeDirectoryRequest" />
</wsdl:message>
<wsdl:message name="SetDataValuesReply">
  <wsdl:part name="SetDataValuesReply" element="types:SetDataValuesReply" />
</wsdl:message>
<wsdl:message name="SetDataValuesRequest">
  <wsdl:part name="SetDataValuesRequest" element="types:SetDataValuesRequest" />
</wsdl:message>

<wsdl:portType name="DEVICE">
```

```
    <wsdl:operation name="GetLogicalDeviceDirectory">
      <wsdl:documentation> Operacija, naudojama norint sužinoti kokie loginiai mazgai sudaro nurodytą
loginį įrenginį.</wsdl:documentation>
      <wsdl:input message="vepis:GetLogicalDeviceDirectoryRequest" />
      <wsdl:output message="vepis:GetLogicalDeviceDirectoryReply" />
    </wsdl:operation>
    <wsdl:operation name="GetLogicalNodeDirectory">
      <wsdl:documentation> Visų nurodytam loginiam mazge matomų duomenų klasių pavadinimus grąžinanti
operacija.</wsdl:documentation>
      <wsdl:input message="vepis:GetLogicalNodeDirectoryRequest" />
      <wsdl:output message="vepis:GetLogicalNodeDirectoryReply" />
    </wsdl:operation>
    <wsdl:operation name="GetDataDefinition">
      <wsdl:documentation> Duomenų klasės (DATA) elementus ir visus atributus apibūdinanti operacija.
Gražina medžio struktūrą su apibūdinimais ir galimų reikšmių aprašymais.</wsdl:documentation>
      <wsdl:input message="vepis:GetDataDefinitionRequest" />
      <wsdl:output message="vepis:GetDataDefinitionReply" />
    </wsdl:operation>
    <wsdl:operation name="GetDataDirectory">
      <wsdl:documentation> Ši operacija padeda sužinoti kokie atributai matomi DATA
elemente.</wsdl:documentation>
      <wsdl:input message="vepis:GetDataDirectoryRequest" />
      <wsdl:output message="vepis:GetDataDirectoryReply" />
    </wsdl:operation>
    <wsdl:operation name="GetDataValues">
      <wsdl:documentation> Duomenų pareikalavimo operacija. Pateikiami pilni keliai iki duomenų
atributo.</wsdl:documentation>
      <wsdl:input message="vepis:GetDataValuesRequest" />
      <wsdl:output message="vepis:GetDataValuesReply" />
    </wsdl:operation>
    <wsdl:operation name="SetDataValues">
      <wsdl:documentation> Reikšmių nustatymo operacija. Pateikiamas pilnas keliaas iki duomenų atributo
ir nustatoma reikšmė. Galima nustatyti po keletą iš karto.</wsdl:documentation>
      <wsdl:input message="vepis:SetDataValuesRequest" />
      <wsdl:output message="vepis:SetDataValuesReply" />
    </wsdl:operation>
  </wsdl:portType>

  <wsdl:binding name="DEVICE" type="vepis:DEVICE">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="GetLogicalDeviceDirectory">
      <soap:operation soapAction="http://localhost:61400/VEPIS/GetLogicalDeviceDirectory" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetLogicalNodeDirectory">
      <soap:operation soapAction="http://localhost:61400/VEPIS/GetLogicalNodeDirectory" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetDataDefinition">
      <soap:operation soapAction="http://localhost:61400/VEPIS/GetDataDefinition" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetDataDirectory">
      <soap:operation soapAction="http://localhost:61400/VEPIS/GetDataDirectory" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
```

```
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GetDataValues">
      <soap:operation soapAction="http://localhost:61400/VEPIS/GetDataValues" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="SetDataValues">
      <soap:operation soapAction="http://localhost:61400/VEPIS/SetDataValues" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="VEPIS">
    <wsdl:port name="DEVICE" binding="vepis:DEVICE">
      <soap:address location="http://localhost:61400/VEPIS/LD" />
    </wsdl:port>
  </wsdl:service>

</wsdl:definitions>
```

3 PRIEDAS. Programos išeities tekstai (Java kalba)

```
package iec61400.vepis.iface;
public interface SERVER {
    public iec61400.vepis.types.GetServerDirectoryReplyDocument
getServerDirectory(iec61400.vepis.types.GetServerDirectoryRequestDocument input);
    public iec61400.vepis.types.LogoffReplyDocument logoff(iec61400.vepis.types.LogoffRequestDocument
input);
    public iec61400.vepis.types.LogonReplyDocument logon(iec61400.vepis.types.LogonRequestDocument input);
}
```

```
package iec61400.vepis.iface;
public interface DEVICE {
    public iec61400.vepis.types.GetLogicalDeviceDirectoryReplyDocument
getLogicalDeviceDirectory(iec61400.vepis.types.GetLogicalDeviceDirectoryRequestDocument input);
    public iec61400.vepis.types.GetLogicalNodeDirectoryReplyDocument
getLogicalNodeDirectory(iec61400.vepis.types.GetLogicalNodeDirectoryRequestDocument input);
    public iec61400.vepis.types.GetDataDefinitionReplyDocument
getDataDefinition(iec61400.vepis.types.GetDataDefinitionRequestDocument input);
    public iec61400.vepis.types.GetDataDirectoryReplyDocument
getDataDirectory(iec61400.vepis.types.GetDataDirectoryRequestDocument input);
    public iec61400.vepis.types.GetDataValuesReplyDocument
getDataValues(iec61400.vepis.types.GetDataValuesRequestDocument input);
    public iec61400.vepis.types.SetDataValuesReplyDocument
setDataValues(iec61400.vepis.types.SetDataValuesRequestDocument input);
}
```

```
package iec61400.vepis;
import iec61400.vepis.iface.SERVER;
import iec61400.vepis.types.GetServerDirectoryReplyDocument;
import iec61400.vepis.types.GetServerDirectoryRequestDocument;
import iec61400.vepis.types.LogoffReplyDocument;
import iec61400.vepis.types.ListOfLDRefs;
import iec61400.vepis.types.LogoffReplyDocument.LogoffReply;
import iec61400.vepis.types.LogoffRequestDocument;
import iec61400.vepis.types.LogonReplyDocument;
import iec61400.vepis.types.LogonReplyDocument.LogonReply;
import iec61400.vepis.types.LogonRequestDocument;

public class VEPISServer implements SERVER {

private VEPISDevice[] DEVICES = null;

public GetServerDirectoryReplyDocument getServerDirectory(GetServerDirectoryRequestDocument input) {
        GetServerDirectoryReplyDocument reply = GetServerDirectoryReplyDocument.Factory.newInstance();
        ListOfLDRefs result = reply.addNewGetServerDirectoryReply();
        for (byte i = 0; i < DEVICES.length; i++)
                result.addLDRef(DEVICES[i].getDEVICE().getName());
        return reply;
```

```
}

public LogoffReplyDocument logoff(LogoffRequestDocument input) {
        LogoffReplyDocument reply = LogoffReplyDocument.Factory.newInstance();
        LogoffReply content = reply.addNewLogoffReply();
        content.setClientRequestHandle(input.getLogoffRequest().getClientRequestHandle());
        return reply;
}

public LogonReplyDocument logon(LogonRequestDocument input) {
        LogonReplyDocument reply = LogonReplyDocument.Factory.newInstance();
        LogonReply content = reply.addNewLogonReply();
        content.setClientRequestHandle(input.getLogonRequest().getClientRequestHandle());
        return reply;
}

public VEPISDevice[] getDEVICES() {
        return DEVICES;
}

public void addDEVICE(VEPISDevice device) {
        VEPISDevice[] ldn = {device};
        VEPISDevice[] old = DEVICES;
        if (old != null) {
                ldn = new VEPISDevice[old.length + 1];
                System.arraycopy(old, 0, ldn, 0, old.length);
                ldn[ldn.length - 1] = device;
        };
        DEVICES = ldn;
}
}
```

```
package iec61400.vepis;
import iec61400.vepis.iface.DEVICE;
import iec61400.vepis.types.GetDataDefinitionReplyDocument;
import iec61400.vepis.types.GetDataDefinitionRequestDocument;
import iec61400.vepis.types.GetDataDirectoryReplyDocument;
import iec61400.vepis.types.GetDataDirectoryRequestDocument;
import iec61400.vepis.types.GetDataValuesReplyDocument;
import iec61400.vepis.types.GetDataValuesRequestDocument;
import iec61400.vepis.types.GetLogicalDeviceDirectoryReplyDocument;
import iec61400.vepis.types.GetLogicalDeviceDirectoryRequestDocument;
import iec61400.vepis.types.GetLogicalNodeDirectoryReplyDocument;
import iec61400.vepis.types.GetLogicalNodeDirectoryRequestDocument;
import iec61400.vepis.types.ListOfData;
import iec61400.vepis.types.ListOfLNRefs;
import iec61400.vepis.types.ListOfValues;
import iec61400.vepis.types.LDDocument;
import iec61400.vepis.types.LDType;
import iec61400.vepis.types.LNDocument;
```

```
import iec61400.vepis.types.LNType;
import iec61400.vepis.types.DATAType;
import iec61400.vepis.types.DAType;
import iec61400.vepis.types.SetDataValuesReplyDocument;
import iec61400.vepis.types.SetDataValuesRequestDocument;
import org.apache.xmlbeans.SimpleValue;
import org.apache.xmlbeans.XmlObject;

public class VEPISDevice implements DEVICE {

private LDDocument DEVICE = LDDocument.Factory.newInstance();

public VEPISDevice(String name) {
        DEVICE.addNewLD().setName(name);
}

public GetLogicalDeviceDirectoryReplyDocument getLogicalDeviceDirectory(
                GetLogicalDeviceDirectoryRequestDocument input) {
        GetLogicalDeviceDirectoryReplyDocument reply =
GetLogicalDeviceDirectoryReplyDocument.Factory.newInstance();
        ListOfLNRefs result = reply.addNewGetLogicalDeviceDirectoryReply();
        for (byte i = 0; i < DEVICE.getLD().sizeOfLNArray(); i++)
                result.addLNRef(DEVICE.getLD().getLNArray(i).getName());
        return reply;
}

public GetLogicalNodeDirectoryReplyDocument getLogicalNodeDirectory(
        GetLogicalNodeDirectoryRequestDocument input) {
        String node = input.getGetLogicalNodeDirectoryRequest();
        String pathExpression = "$this/*:LD/*:LN[@name='" + node + "']/*:DATA/@name";
        XmlObject[] results = DEVICE.selectPath(pathExpression);
        GetLogicalNodeDirectoryReplyDocument reply =
GetLogicalNodeDirectoryReplyDocument.Factory.newInstance();
        ListOfData result = reply.addNewGetLogicalNodeDirectoryReply();
        for (byte i = 0; i < results.length; i++)
                result.addDARef(node + "." + ((SimpleValue) results[i]).getStringValue());
        return reply;
}

public GetDataDefinitionReplyDocument getDataDefinition(GetDataDefinitionRequestDocument input) {
        String reference = input.getGetDataDefinitionRequest();
        GetDataDefinitionReplyDocument reply = GetDataDefinitionReplyDocument.Factory.newInstance();
        DATAType definition = reply.addNewGetDataDefinitionReply();
        if (reference.indexOf(".") > 0) {
                String node = reference.substring(0, reference.indexOf("."));
                String data = reference.substring(reference.indexOf(".")+1);
                String path = "$this/*:LD/*:LN[@name='" + node + "']/*:DATA[@name='" + data + "']/*:DA";
                XmlObject[] results = DEVICE.selectPath(path);
                if (results.length > 0)
                        definition.setDAArray((DAType[])results);
```

```
        };
        return reply;
}

public GetDataDirectoryReplyDocument getDataDirectory(GetDataDirectoryRequestDocument input) {
        String reference = input.getGetDataDirectoryRequest();
        GetDataDirectoryReplyDocument reply = GetDataDirectoryReplyDocument.Factory.newInstance();
        ListOfData result = reply.addNewGetDataDirectoryReply();
        if (reference.indexOf(".") > 0) {
                String node = reference.substring(0, reference.indexOf("."));
                String data = reference.substring(reference.indexOf(".")+1);
                String path = "$this/*:LD/*:LN[@name='" + node + "']/*:DATA[@name='" + data +
"']/*:DA/@name";
                XmlObject[] results = DEVICE.selectPath(path);
                for (byte i = 0; i < results.length; i++)
                        result.addDARef(reference + "." + ((SimpleValue) results[i]).getStringValue());
        };
        return reply;
}

public GetDataValuesReplyDocument getDataValues(GetDataValuesRequestDocument input) {
        String[] references = input.getGetDataValuesRequest().getDARefArray();
        GetDataValuesReplyDocument reply = GetDataValuesReplyDocument.Factory.newInstance();
        ListOfValues result = reply.addNewGetDataValuesReply();
        for (byte i = 0; i < references.length; i++)
                if (references[i].indexOf(".") > 0) {
                        String node = references[i].substring(0, references[i].indexOf("."));
                        String data = references[i].substring(references[i].indexOf(".")+1);
                        String path = "$this/*:LD/*:LN[@name='" + node + "']";
                        while (data.indexOf(".") > 0) {
                                String name = data.substring(0, data.indexOf("."));
                                data = data.substring(data.indexOf(".")+1);
                                path = path + "/*[@name='" + name + "']";
                        };
                        path = path + "/*[@name='" + data + "']/*:value";
                        XmlObject[] results = DEVICE.selectPath(path);
                        if (results.length > 0) {
                                result.addDARef(references[i]);
                                result.addNewValue().set(results[0]);
                        };
                };
        return reply;
}

public SetDataValuesReplyDocument setDataValues(SetDataValuesRequestDocument input) {
        ListOfValues list = input.getSetDataValuesRequest();
        int result = 0;
        for (byte i = 0; i < list.sizeOfDARefArray(); i++)
                if (list.getDARefArray(i).indexOf(".") > 0) {
```

```
                        String node = list.getDARefArray(i).substring(0,
list.getDARefArray(i).indexOf("."));
                        String data = list.getDARefArray(i).substring(list.getDARefArray(i).indexOf(".")+1);
                        String path = "$this/*:LD/*:LN[@name='" + node + "']";
                        while (data.indexOf(".") > 0) {
                                String name = data.substring(0, data.indexOf("."));
                                data = data.substring(data.indexOf(".")+1);
                                path = path + "/*[@name='" + name + "']";
                        };
                        path = path + "/*[@name='" + data + "']/*:value";
                        XmlObject[] results = DEVICE.selectPath(path);
                        if (results.length > 0) {
                                results[0].set(list.getValueArray(i));
                                result++;
                        } else result--;
                };
        SetDataValuesReplyDocument reply = SetDataValuesReplyDocument.Factory.newInstance();
        reply.addNewSetDataValuesReply().setAccessResult(result);
        return reply;
}

public LDType getDEVICE() {
        return DEVICE.getLD();
}

public void createLogicalNodes(String[] LogicalNodes, String Path) throws Exception {
        LNType[] LN = new LNType[LogicalNodes.length];
        for (byte i = 0; i < LogicalNodes.length; i++) {
                LNDocument LNDoc = LNDocument.Factory.parse(new java.io.File("." + Path + LogicalNodes[i] +
".xml"));
                LN[i] = LNDoc.getLN();
        }
        DEVICE.getLD().setLNArray(LN);
}

}
```

```
package iec61400.vepis;
import xsul.xservo.XService;
import xsul.xservices_xbeans.XmlBeansBasedService;
import xsul.xservo_soap_http.HttpBasedServices;

public class VEPIS {
private static final int SERVER_PORT = 61400;
private static final String SERVER_WSDL = "SERVER.wsdl";
private static final String DEVICE_WSDL = "DEVICE.wsdl";
private static final String SERVER_NAME = "VEPIS";
private static final String[] DEVICE_NAMES = {"V52-1", "V52-2", "V52-3"};
private static final String[] LOGICAL_NODES = { "LLN0", "LPHD", "WTUR", "WROT", "WGEN" };
private static final String NODES_DIRECTORY = "\\IEC61400\\VEPIS\\nodes\\";
```

```
private static HttpBasedServices httpServices;

public static void main(String[] args) throws Exception {
  httpServices = new HttpBasedServices(SERVER_PORT);
  //    inicijuojam VEPIS serverio objekta
  VEPISServer logicalServer = new VEPISServer();
  XService service =
    httpServices.addService( new XmlBeansBasedService(SERVER_NAME, SERVER_WSDL, logicalServer));
  service.startService();
  System.out.println("VEPIS serveris sukurtas adresu " + httpServices.getServer().getLocation() + "/" +
SERVER_NAME + "?wsdl");
  //    inicijuojam VEPIS serverio irenginius - vejo elektrines ir ju loginius mazgus
  for (byte i = 0; i < DEVICE_NAMES.length; i++) {
        VEPISDevice newDevice =       new VEPISDevice(DEVICE_NAMES[i]);
        newDevice.createLogicalNodes(LOGICAL_NODES, NODES_DIRECTORY);
        logicalServer.addDEVICE(newDevice);
       XService device =
         httpServices.addService( new XmlBeansBasedService(DEVICE_NAMES[i], DEVICE_WSDL, newDevice));
       device.startService();
        System.out.println("VEPIS įrenginio adresas " + httpServices.getServer().getLocation() + "/" +
DEVICE_NAMES[i] + "?wsdl");
  };
  System.out.println("Spauskite [ENTER] norėdami baigti darbą ir sustabdyti servisus..");
  System.in.read();
  httpServices.shutdown();
 }

}
```

4 PRIEDAS. Loginių mazgų pavyzdžiai (nodes/LLN0.xml ir nodes/LPHD.xml)

```
<?xml version="1.0" encoding="UTF-8"?>
<LN name="LLN0" xmlns="http://IEC:61400/VEPIS/Types" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://IEC:61400/VEPIS/Types
VEPIS.xsd ">
  <DATA name="NamPlt" type="LPL" desc="Name plate">
    <DA name="vendor" type="VISIBLE_STRING" FC="DC"><value xsi:type="xsd:string">VESTAS</value></DA>
    <DA name="swRev" type="VISIBLE_STRING" FC="DC"><value xsi:type="xsd:string">V52 - 0.85MW</value></DA>
    <DA name="d" type="VISIBLE_STRING" FC="DC" desc="loglcal node description"><value
xsi:type="xsd:string">example logical node based on IEC 61400-25</value></DA>
  </DATA>
</LN>


<?xml version="1.0" encoding="UTF-8"?>
<LN name="LPHD" xmlns="http://IEC:61400/VEPIS/Types" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://IEC:61400/VEPIS/Types
VEPIS.xsd ">
  <DATA name="PhyNam" type="WDPL" desc="Physical device name plate">
    <DA name="vendor" type="VISIBLE_STRING" FC="DC"><value xsi:type="xsd:string">VESTAS</value></DA>
    <DA name="hwRev" type="VISIBLE_STRING" FC="DC"><value xsi:type="xsd:string"/></DA>
    <DA name="swRev" type="VISIBLE_STRING" FC="DC"><value xsi:type="xsd:string"/></DA>
    <DA name="serNum" type="VISIBLE_STRING" FC="DC"><value xsi:type="xsd:string"/></DA>
    <DA name="model" type="VISIBLE_STRING" FC="DC"><value xsi:type="xsd:string"/></DA>
    <DA name="location" type="VISIBLE_STRING" FC="DC"><value xsi:type="xsd:string"/></DA>
    <DA name="tmOffset" type="INT16U" FC="CF" desc="Offset from UTC in minutes (excluding daylight saving
time correction)"><value xsi:type="xsd:unsignedShort" xsi:nil="true" /></DA>
    <DA name="tmUseDT" type="BOOLEAN" FC="CF" desc="Flag, indicating if this location uses daylight saving
(summer) time"><value xsi:type="xsd:boolean" xsi:nil="true" /></DA>
    <DA name="d" type="VISIBLE_STRING" FC="CF" desc="Textual description of class"><value
xsi:type="xsd:string">Data of this common data class are used to identify entities like primary equipment
or physical devices</value></DA>
  </DATA>
  <DATA name="PhyHealth" type="INS" desc="Physical Device Health">
    <DA name="stVal" type="INT32" FC="ST" TrgOp="dchg" desc="integer status"><value xsi:type="xsd:int"
xsi:nil="true">5</value></DA>
    <DA name="q" type="Quality" FC="ST" TrgOp="qchg" desc="identifiers that specify the quality and
validity of information">
      <DAC name="validity" type="CODED_ENUM" desc="good | invalid | reserved | questionable"/>
      <DAC name="detailQual" type="PACKED_LIST">
        <DAC name="overflow" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="outOfRange" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="badReference" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="oscillatory" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="failure" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="oldData" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="inconstistent" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="inaccurate" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
      </DAC>
      <DAC name="source" type="CODED ENUM" desc="process | substituted"><value
xsi:type="xsd:string">process</value></DAC>
      <DAC name="test" type="BOOLEAN" desc="DEFAULT false"><value
xsi:type="xsd:boolean">false</value></DAC>
      <DAC name="operatorBlocked" type="BOOLEAN" desc="DEFAULT false"><value
xsi:type="xsd:boolean">false</value></DAC>
    </DA>
    <DA name="t" type="TimeStamp" FC="ST" desc="UTC time representation with the epoch of midnight
(00:00:00) of 1970-01-01">
      <DAC name="SecondsSinceEpoch" type="INT32" desc="(0..MAX) interval in seconds continuosly counted
from the epoch 1970-01-01 00:00:00 UTC"><value xsi:type="xsd:int" xsi:nil="true" /></DAC>
      <DAC name="FractionOfSeconds" type="INT24U" desc="draction of the current second shall be calculated
as SUM from I = 0 to 23 of bi*2**-(I+1) s"><value xsi:type="xsd:nonNegativeInteger" xsi:nil="true"
/></DAC>
      <DAC name="TimeQuality" type="TimeQuality" desc="information about the time source">
        <DAC name="LeapSecondsKnown" type="BOOLEAN" desc="true indicates that the value of
SecondsSinceEpoch takes into account all leap seconds occured"><value xsi:type="xsd:boolean"
xsi:nil="true" /></DAC>
        <DAC name="ClockFailure" type="BOOLEAN" desc="indication that the time source of the sending
device is unreliable (this discards TimeStamp value)"><value xsi:type="xsd:boolean" xsi:nil="true"
/></DAC>
```

```
        <DAC name="ClockNotSynchronized" type="BOOLEAN" desc="indication that the time source of the
sending device is not synchronized with the external UTC time"><value xsi:type="xsd:boolean"
xsi:nil="true" /></DAC>
        <DAC name="TimeAccuracy" type="CODED_ENUM" desc="number of significant bits in the
FractionOfSecond: 7 (T0) | 10 (T1) | 14 (T2) | 16 (T3) | 18 (T4) | 20 (T5)"/>
      </DAC>
    </DA>
  </DATA>
  <DATA name="Proxy" type="SPS" desc="Indicates if this LN is a proxy">
    <DA name="stVal" type="BOOLEAN" FC="ST" TrgOp="dchg"><value xsi:type="xsd:boolean" xsi:nil="true"
/></DA>
    <DA name="q" type="Quality" FC="ST" TrgOp="qchg" desc="identifiers that specify the quality and
validity of information">
      <DAC name="validity" type="CODED_ENUM" desc="good | invalid | reserved | questionable"/>
      <DAC name="detailQual" type="PACKED LIST">
        <DAC name="overflow" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="outOfRange" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="badReference" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="oscillatory" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="failure" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="oldData" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="inconstistent" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
        <DAC name="inaccurate" type="BOOLEAN"><value xsi:type="xsd:boolean" xsi:nil="true" /></DAC>
      </DAC>
      <DAC name="source" type="CODED_ENUM" desc="process | substituted"><value
xsi:type="xsd:string">process</value></DAC>
      <DAC name="test" type="BOOLEAN" desc="DEFAULT false"><value
xsi:type="xsd:boolean">false</value></DAC>
      <DAC name="operatorBlocked" type="BOOLEAN" desc="DEFAULT false"><value
xsi:type="xsd:boolean">false</value></DAC>
    </DA>
    <DA name="t" type="TimeStamp" FC="ST" desc="UTC time representation with the epoch of midnight
(00:00:00) of 1970-01-01">
      <DAC name="SecondsSinceEpoch" type="INT32" desc="(0..MAX) interval in seconds continuosly counted
from the epoch 1970-01-01 00:00:00 UTC"><value xsi:type="xsd:int" xsi:nil="true" /></DAC>
      <DAC name="FractionOfSeconds" type="INT24U" desc="draction of the current second shall be calculated
as SUM from I = 0 to 23 of bi*2**-(I+1) s"><value xsi:type="xsd:nonNegativeInteger" xsi:nil="true"
/></DAC>
      <DAC name="TimeQuality" type="TimeQuality" desc="information about the time source">
        <DAC name="LeapSecondsKnown" type="BOOLEAN" desc="true indicates that the value of
SecondsSinceEpoch takes into account all leap seconds occured"><value xsi:type="xsd:boolean"
xsi:nil="true" /></DAC>
        <DAC name="ClockFailure" type="BOOLEAN" desc="indication that the time source of the sending
device is unreliable (this discards TimeStamp value)"><value xsi:type="xsd:boolean" xsi:nil="true"
/></DAC>
        <DAC name="ClockNotSynchronized" type="BOOLEAN" desc="indication that the time source of the
sending device is not synchronized with the external UTC time"><value xsi:type="xsd:boolean"
xsi:nil="true" /></DAC>
        <DAC name="TimeAccuracy" type="CODED_ENUM" desc="number of significant bits in the
FractionOfSecond: 7 (T0) | 10 (T1) | 14 (T2) | 16 (T3) | 18 (T4) | 20 (T5)"/>
      </DAC>
    </DA>
  </DATA>
</LN>
```