KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Algirdas Mockus

WPF ir Microsoft Visual Studio technologijų panaudojimas vizualizacijos sistemoms kurti

Magistro darbas

Darbo vadovas
doc. dr. Vytautas Pilkauskas

Kaunas
2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Algirdas Mockus

WPF ir Microsoft Visual Studio technologijų panaudojimas vizualizacijos sistemoms kurti

Magistro darbas

Recenzentas
prof. Rimantas Butleris
2009-05-25

Vadovas
doc. dr. Vytautas Pilkauskas
2009-05-25

Atliko
IFM-3/2 gr. stud.
Algirdas Mockus
2009-05-25

Kaunas
2009

TURINYS

SANTRAUKA

Mockus A. WPF ir Microsoft VisualStudio technologijų panaudojimas vizualizacijos sistemoms kurti. Programų sistemų inžinerijos studijų magistro baigiamasis darbas. Mokslinis vadovas Doc. Dr. V. Pilkaukas. Kauno Technologijos Universitetas, Informatikos fakultetas, Kaunas, 2009.

Šio darbo autorius . Šio metodo taikymas gali padėti sumažinti vizualizacijos sistemų kūrimo ir palaikymo kaštus. Naujos sistemos leistų kurti geresnės grafikos kokybės vaizdus bei būtų lengviau išplečiamos ir integruojamos.

Dėl (Windows Presentation Foundation) savo išskirtinių savybių bazinėmis technologijomis pasirinktos WPF ir Microsoft Visual Studio. WPF yra naujas grafikos apdorojimo variklis, skirtas Windows operacinėms sistemoms. Jo pagrindas yra vartotojo sąsajų ir vektorinės grafikos aprašymo kalba XAML. Microsoft Visual Studio yra integruota programų kūrimo aplinka, kuri turi savyje XAML grafinį redaktorių ir palaiko taikomųjų programų kūrimą WPF technologijomis .NET karkasui.

SUMMARY

Mockus A. Applying Microsoft WPF and Visual Studio technologies for creating visualization systems. Masters degree work of Software systems engineering. Scientific leader senior lecturer Dr. V. Pilkauskas. Kaunas University of Technology, Faculty of Informatics, Kaunas, 2009.

The author of this work analyzes possibility to use standard application development techniques for creating process visualization systems. Adoption of this method can result in visualization systems creation and support cost reduction. New system would allow achieving better graphics quality, expandability and lower integration complexity.

WPF (Windows Presentation Foundation) and Microsoft Visual Studio technologies were chosen as the basis because of many exclusive features they have. WPF is brand new graphics engine targeted exclusively for Windows operating system family. Its' core is new XAML language, used for describing either vector graphics or user interfaces in

one manner. Microsoft Visual Studio is an integrated application development environment that contains XAML redactor and supports WPF based application development for .NET framework .

TERMINŲ IR SANTRUMPŲ ŽODYNAS

SCADA	Angl. Supervisory Control And Data Acquisition. Taip vadinamos valdymo ir duomenų surinkimo sistemos, kurios naudojamos automatizuotų procesų pramonėje stebėjimui, parametrizavimui ir valdymui.
WPF	Windows Presentation Foundation. Nauja Microsoft korporacijos platforma vartotojo sąsajoms kurti.
PLC	Programmable Logic Controller. Programuojami (lietuviškai PLV – Programuojamas Loginis Valdiklis). Tai specializuoti įrengimai, kurie atlieka automatizuotų procesų valdymą pagal užduotą algoritmą.
Microsoft Windows DNA	Windows Distributed interNet Applications Architecture
.NET	Karkasas, palengvinantis taikomosios programinės įrangos kūrimą ir skirtas jai vykdyti Windows operacinės sistemos aplinkoje
IT	Informacinės Technologijos
OLE	Object Linking and Embedding
COM	Component Object Model
Web	Dar kitaip World Wide Web, pasaulinis kompiuterių tinklas
SVG	Scalable Vector Graphics.
XML	Extensible Markup Language

ADO.NET	Programinės įrangos komponentų šeima, leidžianti prieiti prie duomenų ir duomenų servisų (pagrinde duomenų bazėse)
Silverlight	Dar vadinama Windows Presentation Foundation/Everywhere. Tai Microsoft technologija skirta vykdyti žiniatinklio programas su sudėtinga vartotojo sąsaja
Java Applet	Specialus JAVA karkaso programos tipas, kuris gali veikti interneto naršyklės aplinkoje
WCF	Windows Communication Foundation
ActiveX	Alternatyvus OLE pavadinimas. Tai Microsoft technologija, skirta kurti pakartotinio panaudojimo objektiškai orientuotiems programinės įrangos komponentams
Modbus	OSI 7 lygio pramoninis komunikacijos protokolas, leidžiantis įvairiems įrenginiams apsikeiti informacija standartizuotu būdu. Būna dviejų rūšių: ModbusTCP (duomenų nešėjas – TCP/IP protokolas), **ModbusSerial** (duomenų nešėjas – RS232 arba RS485 protokolai)
Modbus RTU	Pramoninio duomenų perdavimo protokolo „MODBUS“ versija, skirta duomenims perduoti nuoseklia sąsaja
Modbus TCP	Pramoninio duomenų perdavimo protokolo „MODBUS“ versija, skirta duomenims perduoti per TCP/IP tinklus
OPC	OLE for Process Control.
OSI	Open Systems Interconnection Reference Model. Abstraktus ryšio protokolų, naudojamų ryšio ir kompiuteriniuose tinkluose, aprašymas
XAML	eXtendable Application Markup Language. Korporacijos

	Microsoft XML pagrindu sukurta kalba, skirta aprašyti vartotojo sąsajoms bei grafiniams elementams.
IDE	Integrated Development Environment (Integruota programinės įrangos kūrimo aplinka)
PĮ	Programinė įranga

LENTELIŲ SĄRAŠAS

PAVEIKSLĖLIŲ SĄRAŠAS

ĮVADAS

Pastaruosiuos kelis dešimtmečius esame liudininkai įspūdingos informacinių technologijų raidos. Kiekvienais metais sukuriami vis galingesni ir mažesni kompiuteriai o duomenų laikmenų talpa auga kone eksponentine sparta. Jei dar prieš penkiolika metų džiaugėmės spalvotais kineskopiniais monitoriais, kurie užimdavo didžiąją dalį mūsų stalo, tai šiandien jau galime naudotis plonyčiais lietimui jautriais ekranais tiek savo nešiojamame kompiuteryje tiek telefone. Kartu su aparatūra vystėsi ir programinės sistemos bei programų kūrimo technologijos. Šiai dienai jau niekieno nestebina įspūdinga vartotojo sąsaja su aukštos kokybės grafika ir animuotais elementais.

Deja kiek kitokia situacija yra pramonės automatikos sektoriuje. Čia naudojamos informacinės technologijos, kurių pagrindu kuriamos procesų vizualizacijos, ženkliai atsilieka nuo kitų IT šakų savo pažanga. Tai sąlygoja tiek šias sistemas kuriančių kompanijų technologinis pasyvumas bei verslo politika tiek formuojama bendra nuomonė, kad vizualizacijų estetinė išvaizda nėra svarbi, pakanka, kad sistema būtų funkcionali. Kompanijų pasyvumą lemia tai, kad vizualizacijų sistemų rinka yra ganėtinai maža ir inertiška, todėl jai tenka ir mažesnės investicijos. Šios sistemos turi būti palaikomos ir aptarnaujamos dešimtmetį ir ilgiau. Per šį laiko tarpą labai didelius pinigus kainuojantys automatikos projektai turi atsipirkti ir duoti pelną. Iš kitos pusės sistemų kūrimo ir palaikymo kaštai dideli, nes šios programinės įrangos gamintojai kuria ją nuo pat pamatų patys. Daugumos nuo seno gyvuojančių paveldėtųjų sistemų architektūra yra jau pasenusi, monolitinė ir ganėtinai iškraipyta įvairių pakeitimų ir tobulinimų

Šio darbo tikslas yra vizualizacijų kūrimui pritaikyti šiuolaikines standartizuotas taikomųjų programų kūrimo technologijas, išbandyti tai praktiškai ir ištirti, kokius privalumus ir trūkumus duoda toks sprendimas. Tyrimams bus naudojamas palyginamasis, vertinimo, aprašomasis ir loginis metodai.

1. KOMPIUTERIO VARTOTOJO SĄSAJA IR VIZUALIZACIJOS

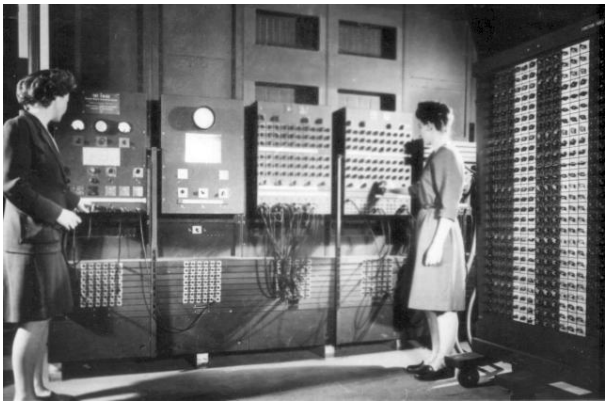
1.1. Vartotojo sąsajos raida

Pažymėtina, kad pirmasis žodžio „kompiuteris“ panaudojimas užfiksuotas dar 1613m. Tuomet taip buvo vadinamas žmogus, kuris atlikdavo skaičiavimus. Nuo 19 amžiaus šiam

žodžiui suteikta visai kita prasmė. Taip pradėta vadinti mašinas, atliekančias matematinius skaičiavimus.

Nustatyta, kad pirmieji kompiuteriai buvo sukurti dar prieš mūsų erą. Tai buvo mechaninės mašinos dažniausiai naudotos astronominiams skaičiavimams atlikti. Jos neturėjo nieko bendro su elektra bei elektronika, kurios yra šiuolaikinių kompiuterių fundamentalusis pagrindas.

Ženklesnė kompiuterių raida prasidėjo antrojo pasaulinio karo metais (apie 1940 metus), kuomet karo pramonei reikėjo naujų technologijų norint įgyti pranašumą prieš priešus. Dar dešimtį metų kompiuterių taikymo sritis nesikeitė, tačiau 1950 metais jie pradėti pardavinėti ir komerciniais tikslais.



Pav. 1 1946m. sukurto kompiuterio ENIAC pagrindinis valdymo skydas

Šaltinis: Wikipedia. Prieiga internetu http://en.wikipedia.org/wiki/ENIAC

Kompiuteris kaip įrenginys visiškai netektų prasmės jei jis negalėtų keistis informacija su žmogumi - jei neturėtų taip vadinamos vartoto sąsajos (ang. UI – User Interface). Kompiuterio vartotojo sąsaja dar dažnai vadinama Žmogus Kompiuteris Sąsaja (angl. HCI – Human Computer Interface) arba Žmogus Mašina Sąsaja (angl. MMI – Man Machine interface). Kompiuterio vartotojo sąsają sudaro dvi dalys:

- Įvestis – leidžia vartotojui valdyti sistemą, perduoti jai reikalingus duomenis ir pan.
- Išvestis – leidžia sistemai pranešti vartotojui apie jo valdymo rezultatus, grąžinti suskaičiuotus duomenis ir pan.

Pirmųjų kompiuterių vartotojo sąsają sudarė daugybė įvairių spalvų lempučių, rodyklinių indikatorių, valdymas buvo atliekamas naudojant mechaninius perjungėjus, kaiščius ir pan. Kiekvienas kompiuteris turėjo unikalią vartotojo sąsają. Vystantis kompiuterių technologijoms sąsaja buvo palaipsniui standartizuojama. Valdymui pradėta naudoti klaviatūras, peles o informacijai atvaizduoti – kineskopinius monitorius, spausdintuvus. Kartu vystėsi ne tik fizinė vartotojo sąsajos įranga bet ir programiniai sprendimai.

Apibendrinant kompiuterio vartotojo sąsajos raidą galima suskirstyti į tris etapus [19]:

- Paketinė sąsaja (angl. batch interface). 1945 – 1968m. Sąsaja neinteraktyvi. Visi reikalingi duomenys turėdavo būti surenkami prieš programos vykdymą ir vienu kartu perduodami kompiuteriui. Rezultatas būdavo grąžinamas tik baigus vykdyti visą programą.

- Komandinė sąsaja (ang. command-line user interface). 1969 – iki dabar. Vartotojas valdo sistemą įvesdamas tekstines komandas ir duomenis kartu. Rezultatą sistema išveda į monitoriaus ekraną (arba spausdintuvą) kaip tekstą. Šio tipo sąsajos naudojamos lig šios dienos, dažniausiai sistemoms administruoti.

- Grafinė vartotojo sąsaja (angl. graphical user interface). 1981 – iki dabar. Grafinė sąsaja standartiškai valdoma klaviatūra ir kompiuterine pele. Rezultatai pateikiami grafiškai monitoriaus ekrane.

1.2. Grafinės vartotojo sąsajos raida

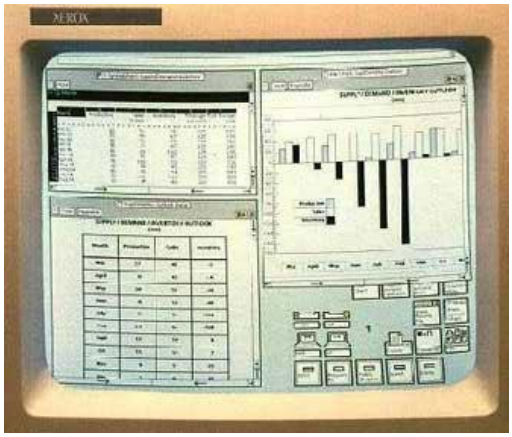
Pirmasis grafinės vartotojo sąsajos prototipas, sukurtas mokslų daktaro Douglas Engelbart ir pavadintas NLS (angl. oN-Line System), buvo pademonstruotas dar 1968 metais. Jis turėjo beveik visas pagrindines savybes, kurios būdingos ir šiandieninėms grafinėms vartotojo sąsajoms: klaviatūrą, panašų įrenginį į dabartinę pelę žymekliui ekrane valdyti, vaizduoklį.

Pav. 2 Sistemos NLS vaizduoklis, klaviatūra ir pelė

Šaltinis: Jeremy Reimer, A History of the GUI. Prieiga internete http://arstechnica.com/articles/paedia/gui.ars

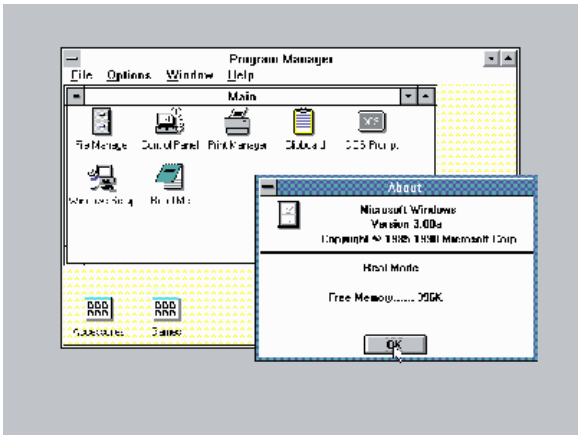
Šis principas buvo pripažintas ir pradėtas naudoti naujuose kompiuterių modeliuose: Xerox Alto (1973m), Xerox Star 8010 (1981m), Apple Lisa (1983m), Apple Macintosh (1984m) ir t.t. [20] Pradedant nuo pirmojo Macintosh kompiuterio grafinė vartotojo sąsaja dar pradėta vadinti WIMP vardu:

- W – Window (langas)
- I – Icon (ikona)
- M – Menu (meniu)
- P – Pointing device (žymeklis ir jo valdymo įrenginys)



Pav. 3 Xerox Star (1987m.)

Šaltinis: Jeremy Reimer, A History of the GUI. Prieiga internete http://arstechnica.com/articles/paedia/gui.ars

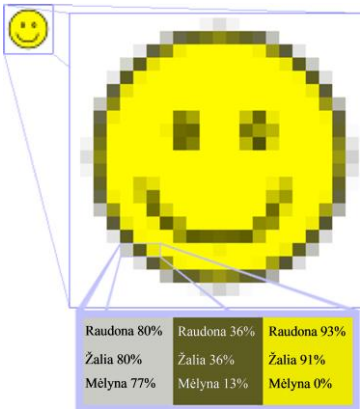


Pav. 4 Windows 3 vartotojo sąsaja

Šaltinis: Jeremy Reimer, A History of the GUI. Prieiga internete http://arstechnica.com/articles/paedia/gui.ars

Nuo to laiko grafinė kompiuterio vartotojo sąsaja evoliucionavo, tobulėjant aparatūriniam išpildymui gerėjo grafikos kokybė, atsirado įvairių karkasų ir metodų, kaip lengvai ir efektyviai kurti programas, pritaikytas šiai sąsajai, pradėta naudoti trimatė grafika, tačiau baziniai principai nesikeitė.

Iki šiol vartotojo sąsajos grafinei informacijai saugoti ir manipuliuoti dažniausiai naudojami taškinės grafikos principai (angl. raster graphics arba pixel graphics).



Pav. 5 Taškinės grafikos pavyzdys

Šaltinis: Wikipedia. Prieiga internete http://en.wikipedia.org/wiki/Raster_graphic

Pav. 6 Taškinės ir vektorinės grafikos palyginimas keičiant vaizdo dydį

Šaltinis: Wikipedia. Prieiga internete http://en.wikipedia.org/wiki/Vector_graphics

Grafiniai elementai, vaizduojami ekrane, yra sudaryti iš daugybės spalvotų taškų. Kiekvieno taško spalvą nusako trys dedamosios: raudona, žalia, mėlyna. Keičiant šių trijų spalvų intensyvumą išgaunami kiti taškų atspalviai. Taip yra taip vadinama RGB spalvų sistema (angl. Red Green Blue – Raudona Žalia Mėlyna). Jei taškų yra daug ir jie nedideli,

grafinis piešinys atrodo vientisas ir natūralus. Vartotojo sąsajos paveikslėliai, ikonos bei kiti elementai yra saugomi kompiuterio atmintyje kaip tokių taškų masyvai. Vaizdavimo metu šie masyvai patalpinami į kompiuterio vaizdo kortos buferius ir atvaizduojami monitoriuje. Šiuo atveju vaizdo kokybė priklauso nuo dviejų faktorių: grafinio elemento rezoliucijos (iš kiek taškų jis sutarytas) ir monitoriaus techninių parametrų. Senieji kineskopiniai monitoriai pagal savo veikimo principus neturėjo aiškiai išskirtų taškų. Juose vaizduojamo vaizdo dydį (rezoliuciją) buvo galima keisti pakankamai dideliame diapozone. Šiuo metu labiausiai paplitusių skystųjų kristalų monitorių ekranai sudaryti iš taškų matricos, kurios dydis nustatomas gamybos metu ir negali būti keičiamas. Šio tipo monitoriuose vaizdo kokybė dar labai priklauso ir nuo paties grafinio vaizdo rezoliucijos. Jei ekrano ir grafinio vaizdo rezoliucija sutampa, gaunamas geriausias kokybės rezultatas. Taškinės grafikos pranašumai:

- Piešimo metu nereikalauja atlikti jokių skaičiavimų ar transformacijų. Grafinio elemento taškų matrica būti gali tiesiai perduodama grafinei kortai apdorojimui. Tokiu būdu užtikrinama gera greitaveika.

- Galima išsaugoti labai sudėtingus grafinius elementus, pvz. nuotraukas, neprarandant jų kokybės ir detalumo (jei rezoliucija yra pakankama).

- Nesudėtingas standartizavimas.

Taškinės grafikos trūkumai:

- Primityvūs grafiniai elementai, kuriuose vyrauja tam tikros geometrinės formos ir vienodos spalvos užima daug vietos

- Grafikos kokybė labai jautri rezoliucijai. Grafinio piešinio kūrimo metu turi būti numatoma, kokio dydžio piešinio reikės. Jei universalumo tikslu yra naudojami didėlės rezoliucijos grafiniai elementai, kurie prieš vaizdavimą yra sumažinami iki reikiamo dydžio, be reikalo išnaudojama labai daug vietos ir sistemos resursų.

- Vaizdavimo metu negalima pakeisti grafinio elemento turinio. Jei pvz. norėtumėme animuoti paveikslėlyje **Error! Reference source not found.** pavaizduoto buteliuko skysčio mažėjimą programos darbo metu, turint vieną taškinį paveikslėlį tai padaryti būtų neįmanoma.

1.3. Vektorinė grafika ir WPF

Pastaruoju metu klasikinis kompiuteris, prie kurio buvome įpratę, pamažu keičiasi. Jis transformuojasi ir išsiskaido į daugybę specializuotų įrenginių, kurie yra labiau integruoti ir pritaikyti kasdieninei žmogaus veiklai. Tarp tokių įrenginių būtų galima paminėti delninius kompiuterius, protingus mobiliuosius telefonus, skaitmeninės televizijos TV priedėlius (angl. Set-top-box) ir kitus panašius įrengimus. Tobulėjant įrenginių techninėms galimybėms ir sudėtingėjant programoms taip pat vis didesni reikalavimai keliami ir vartotojo sąsajai. Ji ne tik turi būti intuityvi, patogi naudojimui bet ir išvaizdi, pasižyminti aukšta grafikos kokybe, animacijomis bei pakankamai lanksti, kad tiktų įvairiems įrenginiams su skirtingais ekranų dydžiais.

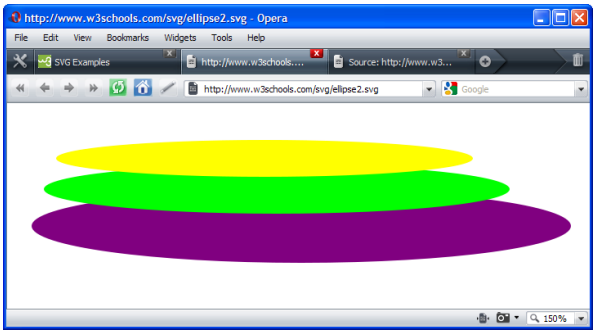
Naujiems inžineriniams vartotojo sąsajos iššūkiams spręsti pradedama naudoti vektorinę grafiką. Macromedia buvo viena iš pirmųjų kompanijų, kuri numatė kokius privalumus vektorinė grafika gali suteikti ir dar 1996 metais rinkai pasiūlė vizualizacijų, animacijų ir interneto programų kūrimo sistemą Flash. 1998 metais IT visuomenei W3C (World Wide Web Concorcium) pristatė SVG (angl. Scalable Vector Graphics) standartą, kuris nusako formatą, kuriuo gali būti aprašomi dvimačiai vektorinės grafikos lementai ir jų kompozicijos. Šis standartas tampa vis populiaresnis. Vis daugiau įrenginių ir programinės įrangos gamintojų įtraukia jį į savo produkciją. Apjungiant tokias technologijas, kaip SVG, HTML, DOM, JavaScript ir AJAX galima sukurti tvirtą bazę universalioms internetinėms programoms kurti.

Paplitus XML kalbai, ji kaip pagrindas naudojama ir vektorinės grafikos standartuose. Paveikslėlyje Pav. 7 pavaizduotas SVG sintaksės fragmentas, kuriame aprašomas trijų skirtingų spalvų ir dydžių elipsių kūrimas. Šalia esančiame Opera naršyklės lange pavaizduotas gautas vaizdas. Didžiausias vektorinės grafikos privalumas yra tai, kad atmintyje yra išsaugomas ne galutinis piešinys, bet instrukcijos, kaip jį nupiešti. Dinamiškai keičiant šias instrukcijas bei jų parametrus galima ne tik pakeisti galutinio piešinio dydį, bet ir spalvas formą, poziciją ir pan.

```
<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">

<ellipse cx="240" cy="100" rx="220" ry="30"
style="fill:purple"/>
<ellipse cx="220" cy="70" rx="190" ry="20"
style="fill:lime"/>
<ellipse cx="210" cy="45" rx="170" ry="15"
style="fill:yellow"/>

</svg>
```



Pav. 7 Pavyzdinis SVG kalbos tekstas ir rezultatas, nupieštas interneto naršyklėje OPERA

Iki pasirodant Microsoft Windows Vista operacinei sistemai vektorinę grafiką vartotojo sąsajoms kurti naudojo tik nedideli IT projektai, įvairius eksperimentus atliko mokslo įstaigos. Tačiau situacija pasikeitė, kai su naująja operacine sistema Microsoft pristatė ir naująją vartotojo sąsajų kūrimo platformą WPF (angl. Windows Presentation Foundation). Jos pagrindą sudaro vektorinės grafikos ir vartotojo sąsajos aprašymo kalba XAML bei naujas piešimo variklis.

Vektorinė grafika neturi taškinės grafikos trūkumų, tačiau reikalauja daugiau aparatūrinio palaikymo, nes grafikos apdorojimo metu reikia atlikti daug skaičiavimų, transformavimų ir piešimo operacijų. Ši problema sprendžiama:

- Panaudojant egzistuojančius trimatės grafikos variklius. Pvz. WPF naudoja DirectX programavimo sąsają ir grafinės kortos centrinį procesorių. Qt ir kai kurie linux projektai naudoja OpenGL tam, kad paspartinti SVG apdorojimą.

- Kuriant specialius standartus aparatūriniam vektorinės grafikos palaikymui. Pvz. OpenVG.

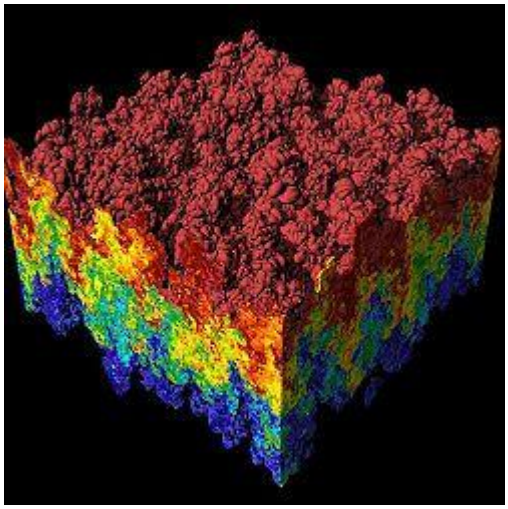
1.4. Vizualizacija

Žodis vizualinis (lot. kalboje *visualis*) reiškia „regimas“. Taigi bendrai vizualizaciją (angl. visualization) galima apibūdinti kaip „padarymą regimu“ arba įvairų rūšių informacijos kodavimą/vertimą į regimuosius vaizdus. Vizualizavimo principas yra pakankamai senas. Jo užuomazgas galima aptikti dar prieš mūsų erą. Tai piešiniai olose, Egiptiečių hieroglifai, Graikų geometrija ir t.t.

Vystantis kompiuterinėms technologijoms ir kompiuterinei grafikai šie suteikė visiškai naujas galimybes vizualizacijoms, praplėtė jų pritaikymo sritį. Iš kompiuterinės grafikos

perspektyvos vizualizaciją galima apibrėžti kaip duomenų regimąjį pateikimą, informacijos atvaizdavimą monitoriaus, projektoriaus ar kitame ekrane tam, kad atskleisti tam tikras informacijos savybes, sustiprinti jos suvokimą. Išskiriamos tokios kompiuterinės vizualizacijos rūšys [22]:

- Mokslinė vizualizacija. Apima duomenų, gautų mokslinių eksperimentų arba simuliacijų metu, išrinkimą, transformavimą ir atvaizdavimą. Duomenys atvaizduojami pritaikant tam tikras geometrines struktūras.



Pav. 8 Mokslinės vizualizacijos pavyzdys

Šaltinis: Wikipedia. Prieiga internete http://en.wikipedia.org/wiki/Visualization_(computer_graphics)

- Mokomoji vizualizacija. Panaudojant imitaciją kompiuterio pagalba sukuriamas grafinis vaizdas objektų, kuriuos kitu būdu pamatyti būtų sunku arba visai neįmanoma. Tai palengvina mokymąsi, tų objektų studijavimą, tyrinėjimą, suvokimą. Pavyzdžiai galėtų būti: atomo sandara, žmogaus kaulinio skeleto struktūra ir pan.

- Informacijos vizualizacija (InfoViz). Koncentruojasi į kompiuterinių priemonių panaudojimą tyrinėti didelius kiekius abstrakčios informacijos. Lyginant su moksline vizualizacija, abstraktūs duomenys neturi jokios paveldėtos geometrinės struktūros. Svarbios šio tipo vizualizacijos savybės yra vaizdavimo dinamiškumas ir interaktyvumas. Vartotojas gali keisti vizualizaciją realiame laike.

- Žinių vizualizacija. Pagrindinis jos tikslas yra pagerinti žinių apsikeitimą tarp skirtingų žmonių panaudojant kompiuteriu (ir ne tik) sukurtas vizualizacijas. Tokių vizualizacijų pavyzdžiai gali būti įvairios diagramos, eskizai, nupiešti įsivaizduojami tam

tikrų objektų vaizdai. Lyginant su informacijos vizualizacija, kurios pagrindiniai tikslai yra panaudojant kompiuterines priemones sukurti naujus informacijos suvokimo lygmenis, žinių vizualizacijos pagrindiniai tikslai yra naujų žinių perdavimas grupėms.

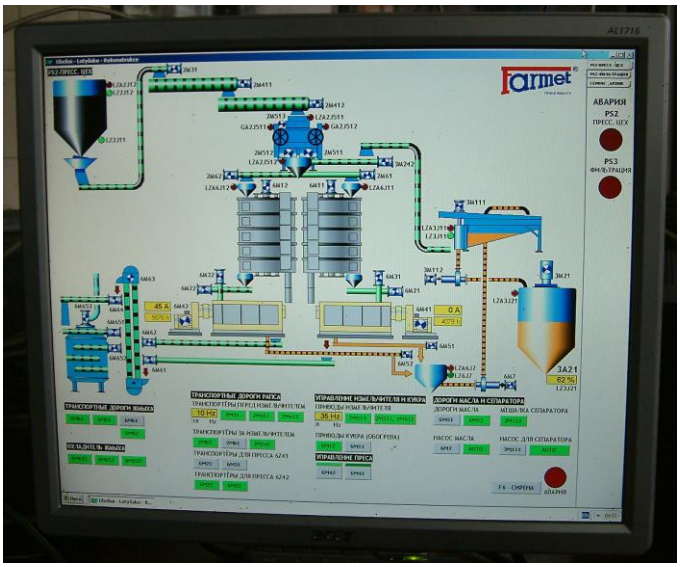
- Produktų vizualizacija. Apima vizualizavimo programinės įrangos technologijas, skirtas trimačių modelių, brėžinių ir kitų su gamybos procesais susijusių dokumentų peržiūrai ir manipuliacijai. Produktų vizualizacija suteikia galimybes pamatyti būsimą produktą jo dar nepagaminus.

1.5. Procesų vizualizacija ir SCADA sistemos

Procesų vizualizacija yra viena iš sudėtinių informacijos vizualizacijos (InfoViz) dalių [21]. Pagrindinė jos užduotis yra suprantamai atvaizduoti vykstančio proceso būseną ir kitą susijusią informaciją jį prižiūrinčiam personalui, kad šie galėtų priimti valdymo ir kitus sprendimus. Keli procesų ir jų vizualizacijų pavyzdžiai:

- miesto vandentiekio sistema. Siurblių darbo ir slėgio vamzdynuose stebėjimas

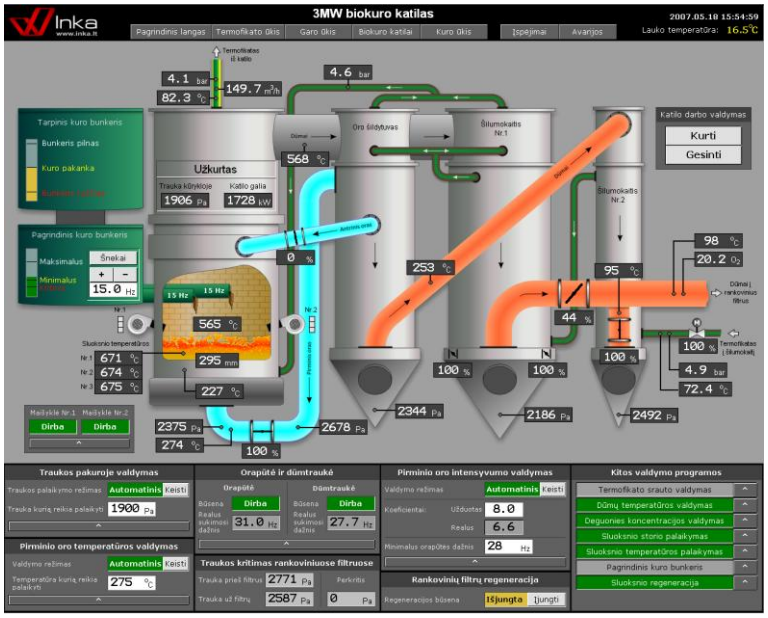
- gamybos linijų ir kitų įrengimų darbo stebėjimas gamybinėse įmonėse



Pav. 9 Rapsų aliejaus spaudimui naudojamo pramoninio preso vizualizacijos ir valdymo sistema

Šaltinis: UAB „INKA“ nuotraukų archyvas

- šiltą vandenį miestui ruošianti katilinė. Šilumą gaminančių katilų darbo parametrų, slėgių ir temperatūrų trasose stebėjimas



Pav. 10 Šilumą miestui gaminančio katilo vizualizacija

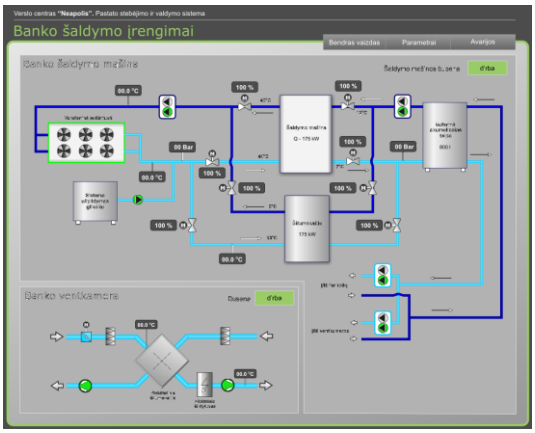
Šaltinis: UAB „INKA“ nuotraukų archyvas

- bokštinis kranas. Keliančiųjų mechanizmų darbo parametrų, vėjo greičio stebėjimas

- gyvenamasis namas arba biurų pastatas. Šaldymo, vėdinimo agregatų parametrų, patalpų temperatūrų, energijos suvartojimo, įjungto apšvietimo stebėjimas

Pav. 11 Namų vizualizacijos panelė

Šaltinis: UAB „INKA“ nuotraukų archyvas



Pav. 12 Šaldymo ir vėdinimo agregatų pastate vizualizacija

Šaltinis: UAB „INKA“ nuotraukų archyvas

- grūdų džiovykla. Temperatūrų džiovinimo kameros skirtingose zonose stebėjimas, grūdų drėgnumo skirtinguose lygiuose sekimas

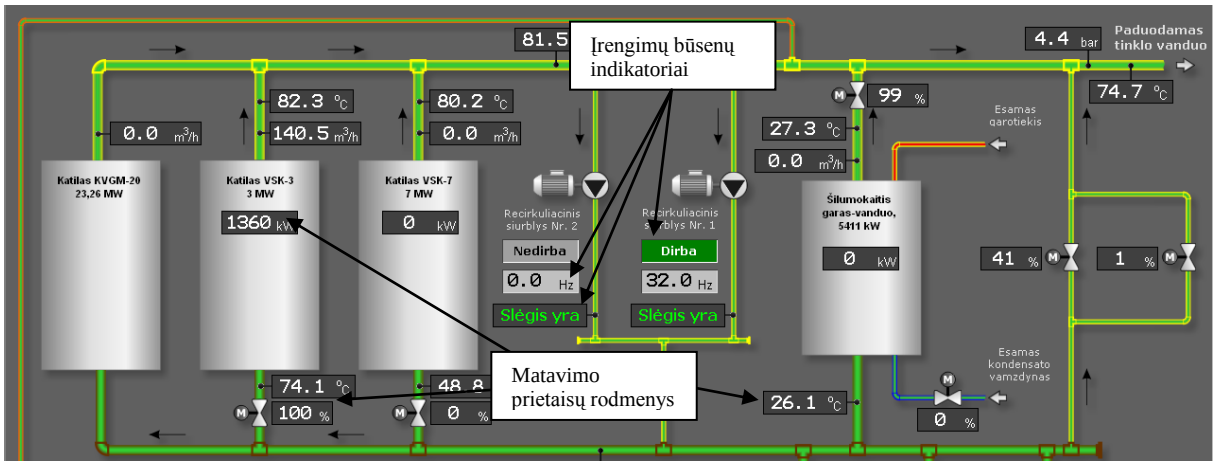
Galima pateikti dar daugybę kitų procesų pavyzdžių, kuriems stebėti naudojamos vizualizavimo technologijos. Svarbiausia tai, kad egzistuotų techniniai sprendimai, kurie leistų nustatyti vizualizuojamo proceso būseną ir paversti ją į skaitmeninį formatą (pvz. temperatūros, slėgio, traukos, vandens srauto, deguonies kiekio matavimo prietaisai).

Standartinę proceso vizualizaciją sudaro:

1. Dvimatis schematinis proceso/objekto vaizdas. Jo paskirtis yra suprantamai atvaizduoti esmines proceso/objekto dalis bei procese dalyvaujančius įrengimus. Detalumo lygis priklauso nuo vizualizacijos paskirties ir apimties. Schemoje gali būti nurodoma ir papildoma informacija, pvz. įrengimų pavadinimai, matavimo prietaisų kodai, skysčių tekėjimo kryptys ir pan. Schemose nesant išimtinėms situacijoms nėra atvaizduojami: objektų dydžiai; tikslios dydžių proporcijos tarp objektų; atstumai tarp objektų; objektų tikslios formos; patalpos ar aplinka, kurioje objektai randasi; objektai, kurie neturi įtakos proceso vyksmui.

2. Matavimo prietaisų rodmenys. Dažniausiai atvaizduojami skaičiais, tačiau gali būti naudojama ir tam tikrų fizinių prietaisų imitacija (pvz. skalė su rodyklėmis)[3]. Matavimo prietaisai padedami (arba linija nurodoma) toje schemos dalyje, kurioje jie iš tikrųjų yra ir kurios parametrus matuoja.

3. Įvairūs įrengimų būsenų indikatoriai. Schemoje išdėstomi įrengimų (pvz. variklių, sklendžių ir pan.) simboliai kurių būsena atvaizduojama arba keičiant simbolių spalvas arba pridedant papildomus užrašus, skaičius ir pan.



Pav. 13 Katilinės termofikacinės dalies vizualizacijos fragmentas

Šaltinis: UAB „INKA“ nuotraukų archyvas

Pramonėje, energetikoje, pastatuose vizualizacijos dažniausiai yra naudojamos kartu su procesų automatizavimo technologijomis ir yra SCADA sistemų dalis. SCADA angl. Supervisory Control And Data Acquisition. Taip vadinamos specializuotos vizualizacijos, valdymo ir duomenų surinkimo sistemos. Pagrindinis šių sistemų tikslas yra surinkti duomenis iš matavimo prietaisų bei įvairių įrenginių, valdančių procesus (pvz. programuojamų loginių valdiklių, variklių dažnio keitiklių ir pan.), juos apdoroti ir vizualiai ar kitais būdais (pvz. garsiniai signalai) pateikti operatoriams, prižiūrintiems procesų vyksmą. Šios sistemos nuo paprastų vizualizacijų skiriaisi tuo, kad leidžia ne tik stebėti procesą bet ir jį valdyti siunčiant operatorių įvestus duomenis į valdymo įrenginius. Surenkami duomenys gali būti archyvuojami duomenų bazėse vėlesnei kompleksinei analizei, perduodami aukštesnio lygio informacinėms sistemoms, tokioms kaip gamybos valdymo, apskaitos ir panašioms. SCADA sistemos yra neatsiejama šiuolaikinės automatikos dalis.

Augant kompiuterinės įrangos prieinamumui, kylant žmonių kompiuteriniam raštingumui bei sudėtingėjant procesams, kuriuos žmogus užsibrėžia valdyti, SCADA sistemos yra ir bus naudojamos vis dažniau, keliami vis didesni reikalavimai šių sistemų lankstumui, adaptyvumui, patogumui, jose vaizduojamo turinio išvaizdai.

2. PROBLEMA, DARBO TIKSLAI IR UŽDAVINIAI

Šiame darbe vizualizacijos sistemomis toliau apibendrintai vadinsime tiek SCADA sistemas tiek paprastas procesų vizualizacijas.

Dėl rinkos ypatumų ir kompanijų technologinio pasyvumo dauguma vizualizacijų kūrimo sistemų savo techninėmis galimybėmis ženkliai atsilieka nuo kitų IT sričių. To rezultatas:

- Sistemos turi daug paveldėtojo išeities kodo ir yra priklausomos nuo jau nenaudojamų technologijų. Tokių sistemų integracija, palaikymas ir priežiūra yra sudėtinga.
- Dėl mažos rinkos ir investicijų vizualizacijų kūrimo programos nėra patogios darbui ir tokiu būdu mažina produktyvumą.
- Sistemų grafinės galimybės yra ganėtinai ribotos. Neįmanoma neprarandant kokybės keisti vizualizacijos lango dydį.

- Norint sukurti išplėtimo modulius tenka nagrinėti pačių gamintojų sukurtas programų kūrimo sąsajas (API, angl. Application Programming Interface), kurios yra prastai dokumentuotos, ribotų galimybių, naudoja pasenusias technologijas.

Vizualizacijos sistemos yra labai panašios į taikomosias programomas. Su pastarosiomis jos turi labai daug panašumų: langai, mygtukai, duomenų įvedimo laukai, įvairūs grafiniai elementai ir t.t. Šio darbo tikslas yra ištirti galimybę panaudoti modernias taikomųjų programų kūrimo technologijas vizualizacijos sistemoms kurti. Pakartotoinis įrankių panaudojimas gali ženkliai sumažinti sistemų kūrimo kaštus, pagerinti jų kokybę. Iškeliami sekantys klausimai:

- Kokią naudą suteiktų vektorinės grafikos naudojimas vizualizacijų kūrime
- Kokius pranašumus WPF karkasas turi prieš kitas vektorinės grafikos sistemas
- Kokias naujojo WPF karkaso savybes galima pritaikyti vizualizacijų kūrimui
- Ar galima panaudoti Microsoft Visual Studio aplinką kaip bazę vizualizacijų kūrimo sistemai sudaryti. Kokius privalumus tai suteikia.

Idėjoms išbandyti ir atsakymams rasti į kai kuriuos klausimus bus sukurtas jau egzistuojančios vizualizacijos sistemos analogas su apribotomis galimybėmis. Naujoji sistema bus kuriama tik su Microsoft Visual Studio programų kūrimo paketu. Vartotjo sąsajai kurti bus naudojamas WPF ir .NET karkasai.

3. ESAMA SITUACIJA

SCADA sistemos nėra naujiena ir naudojamos pramoninėje automatikoje jau ne pirmą dešimtmetį, tad jų pasiūla rinkoje pakankamai didelė. Šiuo metu praktiškai kiekviena didesnė automatikos įrangą gaminanti įmonė gali pasiūlyti ir savo vizualizacijos programinę įrangą, kuri be abejo geriausiai yra suderinama su pačios kompanijos gaminama produkcija. Tokia situacija susiklostė todėl, kad dar praeitame dešimtmetyje, kuomet standartizacija nebuvo toli pažengusi šiame rinkos segmente, kiekvienas automatizacijos įrangos gamintojas kūrė savo specifinę įrangą ir komunikacijos protokolus, kuriais buvo galima su ja komunikuoti. Jeigu

projekte buvo naudojama tam tikro gamintojo įranga, tuomet užsakovas turėdavo rinktis ir to paties gamintojo SCADA sistemą, nes kitos sistemos nesugebėdavo „susikalbėti“ su įrenginiais. Tačiau iki šios dienos situacija ganėtinai pasikeitė. Jau sukurtas ne vienas standartas komunikacijos protokolams bei SCADA sistemoms (OPC, Modbus, DeviceNet, CANopen, Profibus ir t.t). OPC standartų šeima standartizavo ne tik sąsają su išoriniais įrenginiais bet ir įgalino kelių gamintojų skirtingas SCADA sistemas perduoti duomenis viena kitai (3). To pasekoje atsirado ne viena kompanija, kuri užsiima vien tik SCADA sistemų kūrimu.

SCADA sistemų rinka yra gerokai mažesnė ir inertiškesnė nei kitų plačiam vartotojų ratui skirtų sistemų, todėl naujovės, kurias jau senai matome ir naudojame visuotinai paplitusiuose programiniuose paketuose, sunkiau prigyja vizualizacijos sistemose. Tai sąlygoja didelės tokių sistemų kūrimo ir realizavimo išlaidos bei faktas, jog šios sistemos turi būti palaikomos ir aptarnaujamos dešimtmetį ir ilgiau. Per šį laiko tarpą labai didelius pinigus kainuojantys automatikos projektai turi atsipirkti ir duoti pelną. Iš kitos pusės SCADA sistemų kūrimo ir palaikymo kaštai dideli, nes šios programinės įrangos gamintojai kuria ją nuo pat pamatų patys. Daugumos nuo seno gyvuojančių sistemų architektūra yra jau pasenusi, monolitinė ir ganėtinai iškraipyta įvairių pakeitimų ir tobulinimų.

3.1. Monitor Pro

Tai galingiausia kompanijos Schneider-Electric SCADA sistema. Jos pagrindą sudaro kompanijos Technomatix produktas - FactoryLink serveris, skirtas duomenims surinkti ir apdoroti. Monitor Pro prie minėto serverio prideda vizualizacijų kūrimo įrankį, vadinamą Client Builder bei keletą papildomų serverio komponentų. Ši sistema nesusilaukė didesnio populiarumo ir po neilgo savo gyvavimo laikotarpio jos tolimesnis kūrimas buvo nutrauktas.

Monitor Pro sistema yra ganėtinai griozdiška. FactoryLink serveris buvo sukurtas Microsoft Windows DNA technologijos pagrindu, kurią jau senai pakeitė tos pačios kompanijos .NET platforma. Yra numatytas serverio galimybių praplėtimo mechanizmas, tačiau jis prastai dokumentuotas ir norint juo pasinaudoti reikia pirkti papildomas priemones iš gamintojo. Kurti skriptus galima dvejomis kalbomis (7):

- Mathlogic – ja galima kurti skriptus serveriui. Ši kalba yra kompanijos Technomatix kūrinys ir savo sintakse labai panaši į Pascal programavimo kabą. Šios skriptavimo kalbos galimybės skurdžios ir labai prastai dokumentuotos.
- Visual Basic for Applications – tai labai paplitusi skriptavimo kalba, sukurta Visual Basic programavimo kalbos pagrindu., Ją galima naudoti tokiose programose kaip Microsoft Excel, Microsoft Word. Šia skriptavimo kalba galima naudotis tik Monitor Pro vizualizacijų kūrimo įrankyje Client Builder. Nors patik skriptavimo kalba yra pakankamai galinga, tačiau pats Client Builder objektinis modelis, prieinamas šiai kalbai, yra ribotų galimybių.

Monitor Pro sistemos konfigūracijos sąsaja yra ganėtinai nepatogi ir atrodo senoviškai. Vizualizacijų kūrimo įrankiui trūksta lankstumo ir kai kurių labai naudingų bei darbą pagreitinančių funkcijų. Sukurta vizualizacija yra nekompiliuojama o interpretuojama tos pačios sistemos. Tai leidžia greitai keisti režimus iš konfigūravimo į darbo bei atvirkščiai visai sistemai arba tam tikram vienam vizualizacijos langui. Ši savybė labai patogi testuojant ir derinant sukurtą vizualizaciją. Kuriant vizualizacijas taipogi galima panaudoti ActiveX bei Java Beans komponentus (8).

3.2. VijeoLook

Tai dar vienas kompanijos Shneider-Electric kūrinys. Ši SCADA sistema yra buvo sukurta Monitor Pro vizualizacijų kūrimo įrankio Client Builder pagrindu, todėl paveldėjo visas šios sistemos blogąsias ir gerąsias savybes. VijeoLook sistema yra skirta kurti nedidelės apimties ir ribotų architektūrinių galimybių SCADA sistemos kurti. Ji yra monolitinė ir sukurta principu „viskas viename“ nenaudojant kliento-serverio architektūros.

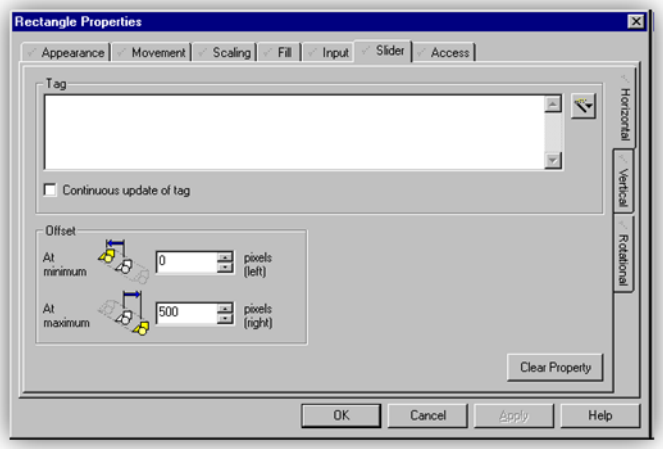
3.3. VijeoCitect

Pakankamai populiari SCADA sistema, kurią kuria Australų kompanija Citect. Pavyzdžiui šis vizualizacijos paketas yra naudojamas AB „Lifosa“ gamykloje, esančioje Kėdainiuose, gamybiniams procesams vizualizuoti.

Šis vizualizacijos paketas naudoja pakankamai pasenusius komponentus (pvz. visi projekto nustatymai saugomi dBase tipo duomenų bazėje), vartotojo sąsaja ganėtinai nepatogi ir vizualiai atrodo pasenusi, sistemos įsisavinimas trunka nemažai laiko, nes didžiąją dalį

funkcionalumo galima pasiekti tik panaudojant integruotą skriptavimo kalbą. Dėl pastarosios priežasties sistemą sunkiau perpranta programuotojo žinių neturintys asmenys.

Skriptų kūrimui galima naudoti dvi integruotas kalbas: Cicode ir CitectVBA. Nedidelis trūkumas yra tai, kad šių dviejų kalbų galimybės nėra vienodos ir su kiekviena iš jų galima pasiekti skirtingą funkcionalumą. Cicode yra skriptavimo kalba, sukurta pačios Citect kompanijos. Jos semantika labai panaši į „C“ šeimos programavimo kalbas. Daugelis specialistų pripažįsta, kad Citect sistemos skriptavimo galimybes yra pačios didžiausios lyginant su kitomis SCADA sistemomis. Tai ypatingai pravartu tose situacijose, kai reikia įgyvendinti nestandartinį funkcionalumą per ribotą laiką ir su ribotomis sąnaudomis.

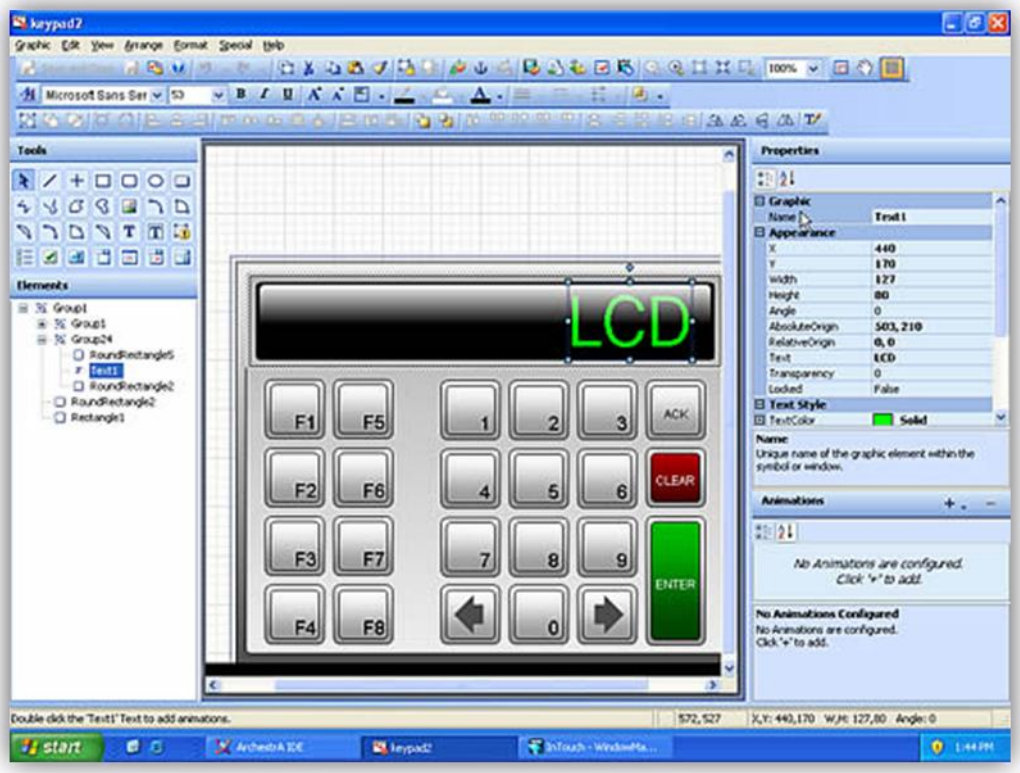


Pav. 14 Vijeo Citect sistemos grafinio objekto parametrų langas

Pav. 15 paveikslėlyje pavaizduotas tipinis bet kurio grafinio vizualizacijos elemento parametrų nustatymo langas. Šioje sistemoje unikalu tai, kad bet kurį objekto parametrą galima susieti arba su tam tikru kintamuoju (dar kitaip vadinamu „tag“), kuris gali būti tarpinis skaičiavimų rezultatas arba nuskaitomas iš išorinio prietaiso, arba su tam tikra Cicode kalbos išraiška (9). Naujai kuriamoje vizualizacijos sistemoje būtinai turėtų būti realizuotos panašios į Vijeo Citetct skriptavimo galimybės, tačiau dažniausiai naudojamam funkcionalumui pasiekti turėtų būti numatyta ir grafinė vartotojo sąsaja su dialogo langais ir pan. (pvz. įvedamų duomenų valiavimas).

3.4. InTouch

Tai šiuo metu Lietuvoje (ir netik) pati populiariausia vizualizacijos sistema, kurią kuria JAV kompanija Wonderware. Šioje SCADA sistemoje yra labai gerai suderintas sistemos funkcionalumas ir paprastumas naudoti, vartotojo sąsaja labai patogi ir atrodo moderniai. Norint praplėsti sistemos funkcionalumą galima ne tik skirptuoti, naudoti ActiveX ar COM komponentus (kaip kad kitose minėtose SCADA sistemose) bet ir įkelti grafinius Microsoft .NET komponentus. Tai išskiria šią sistemą iš augelio kitų SCADA paketų. Jei ši sistema naudojama didelės apimties procesams vizualizuoti, galima panaudoti kitą tos pačios kompanijos produktą, vadinamą „ArchestraA“. Tai yra serverio programinė įranga, kuri leidžia surinkti įvairaus tipo duomenis iš išorinių šaltinių (valdiklių, kitų valdymo sistemų, duomenų bazių), juos apdoroti, registruoti, pateikti InTouch vizualizacijos sistemai. Išskirtinis šios sistemos bruožas – galimybė kurti objektiškai orientuotas duomenų struktūras. Šis principas pasiskolintas iš objektiškai orientuoto programavimo ideologijos. Tai leidžia logiškai struktūrizuoti didelį informacijos kiekį, lengviau jį valdyti bei panaudoti (2).



Pav. 16 InTouch 10 vizualizacijų kūrimo sistemos pagrindinis langas

4. PASIRINKTO SPTRENDIMO ANALIZĖ

4.1. SCADA sistemų savybės

Išanalizavus egzistuojančius SCADA sistemų sprendimus galima nusakyti pagrindines šių sistemų savybes:

- Vizualizacijos kūrimas panaudojant: integruotus į aplinką komponentus (mygtukus, duomenų įvedimo laukus ir pan.), grafinius primityvus, įvairių formatų paveikslėlius, COM, ActiveX bei .NET komponentus
- Vartotojo komponentų kūrimas panaudojant jau esamus komponentus bei grafinius primityvus, paveikslėlius ir pan.
- Animacijos naudojimas
- Daugiakalbiškumas
- Funkcionalumo praplėtimas naudojant skriptvimą
- Konfigūruojami duomenų šaltiniai (išoriniai valdymo įrenginiai, kitos vizualizacijos sistemos, duomenų bazės)
- Ataskaitų formavimas ir spausdinimas
- Kintamųjų registravimas duomenų bazėse
- Vidinių kintamųjų kūrimas

4.2. Įgyvendinimo problemos

4.2.1. Požiūris į vizualizacijos sistemų kūrimą

Praktiškai kiekvienas SCADA sistemų gamintojas yra sukūręs savo unikalią vizualizacijų kūrimo aplinką ir metodologiją. Nors visos sistemos ir skirtingos, tačiau turi vieną bendrą savybę – galutinis produktas nėra taikomoji programa, kurią kurti yra įpratę IT sektoriuje dirbantys programuotojai. Vizualizacijų kūrimo procesas dažniausiai yra vadinamas SCADA paketų konfigūravimu, kurio pasekoje sukuriamas vizualizacijos projektas, susidedantis iš įvairaus formato tekstinių ir binarinių failų. Vėliau šis projektas užkraunamas į tą pačią SCADA sistemą rodymo režimu kaip duomenys ir interpretuojamas. Kai kurios sistemos projektus kompiliuoja, tačiau to pasekoje gaunama ne standartinė programa, o tam tikro formato suoptimizuoti ir valiuoti duomenys, kuriuos suprasti gali tik ta pati sistema.

Visumoje tiek įprastas taikomąsias programas tiek vizualizacijos sistemas galima sutapatinti. Tiek viena tiek kita turi labai daug bendrų bruožų ir tikslų. Sukurta vizualizacija taipogi yra taikomoji, ji skirta konkrečiam atvejui. Mano manymu nuo seno taikomas vizualizacijos sistemų kūrimo atskyrimas nuo taikomųjų programų kūrimo yra netikslingas ir sukuria daug problemų, su kuriomis susiduria ne vienas SCADA sistemų gamintojas. Tiesiog kyla klausimas, kam išradinėti dviratį. Juk taikomųjų programų kūrimas yra pakankamai gerai išdirbtas:

1. Sukurta ne viena pilnavertiška taikomųjų programų kūrimo aplinka su išeities kodo redaktoriais, grafinės vartotojo sąsajos kūrimo priemonėmis ir pan.
2. Galima rinktis iš daugybės programavimo kalbų, turinčių gerai išdirbtą funkcionalumą
3. Sukurtas ir sėkmingai naudojamas ne vienas programinės įrangos karkasas (framework), palengvinantis sistemų kūrimą ir leidžiantis sistemoms nepriklausyti nuo operacinės sistemos
4. Kuriama ideologija kaip programinės įrangos kūrimą tam tikrai sričiai padaryti efektyvesnį (programinės įrangos gamyklos)(10)
5. Gerai išvystytos projektavimo ir testavimo priemonės

4.2.2. Infrastruktūros pasirinkimas

Šio projekto tikslas yra pritaikyti taikomųjų programų kūrimo technologijas vizualizacijoms kurti. Kadangi pastarųjų technologijų yra pakankamai daug, reikia aiškiai apsibrėžti kas, kaip ir kodėl bus naudojama

Operacinės sistemos pasirinkimas

Šiuo metu pakankamai sparčiai auga atviro kodo sistemos Linux naudojimas. Kai kurie šaltiniai teigia, kad Unix šeimos operacinės sistemos tinka SCADA aplikacijoms [13]. Linux operacinė sistema jau buvo naudota ne viename projekte, tačiau pramoninės automatikos sektoriuje tvirtas pozicijas vis dar užima kompanijos Microsoft operacinių sistemų šeima Windows. Taip yra todėl, kad dauguma standartų pritaikyti tik Windows aplinkai. Pvz. OPC šifruojasi OLE for Process Control. OLE yra kompanijos Microsoft komponentinė technologija. Išties paskutinės OPC standarto versijos naudoja OLE pasekėją - technologiją COM [3]. Šiuo metu kuriamas naujas OPC standartas, vadinamas OPC UA (OPC Unified Architecture), kuris vietoj COM technologijos naudos Web Servisus. Tai

įgalins sistemas, dirbančiose skirtingose operacinėse sistemose ir sukurtas skirtingomis technologijomis, keistis duomenimis [5][3]. Taigi atsiras galimybė plačiau naudoti ne tik Windows operacines sistemas, tačiau dar ilgą laiką išliks personalo, aptarnaujančio „egzotinę“ įrangą problema. Šiuo metu pramoniniuose objektuose dirbantis IT personalas dažniausiai naudoja ir moka dirbti su Windows operacinėmis sistemomis, o jų perkvalifikavimas arba naujų darbuotojų paieška bei samdymas kainuos brangiau nei kelių Windows licencijų nupirkimas.

Programinės įrangos kūrimo karkaso (framework) pasirinkimas

Kadangi bus naudojama Windows tipo operacinė sistema, tai galima rinktis iš trijų pakankamai populiarių programinės įrangos kūrimo karkasų: .NET, Java ir QT. .NET technologija turi daug privalumų lyginant su kitomis dvejomis:

a. Kadangi .NET yra Microsoft kompanijos produktas, tai jis užtikrina patį geriausią suderinamumą su Windows operacine sistema ir leidžia išnaudoti visas jos galimybes
b. Windows aplinkoje .NET technologija sukurtos programos yra ženkliai greitesnės už Java (tik ne už QT). Sparta SCADA sistemoms yra aktuali, nes vienu metu tenka vaizduoti ir apdoroto pakankamai didelius kiekius duomenų, tad vartotojo sąsaja turi veikti greitai.
c. .NET karkasas turi labai galingą ir modernią vartotojo sąsajos kūrimo technologiją WPF (Windows Presentation Foundation)[6][11], kuriai neprilygsta nei Java nei QT technologijos. Šiuo metu yra mėginama pritaikyti SVG standartą vektorinės grafikos vartotojo sąsajai kurti Java aplinkoje, tačiau SVG iš esmės nėra skirtas tam tikslui. Tačiau kol kas tai tik tyrimų ir bandymų objektas [17].

4.3. Vizualizacijų kūrimo aplinkos pasirinkimas

Kadangi tikslas yra panaudoti kuo daugiau jau sukurtų ir naudojamų technologijų, tai vizualizacijų programinės įrangos kūrimui mes naudosime gerai žinomą ir populiarų Microsoft Visual Studio 2005 paketą. Šis įrankis yra skirtas kurti taikomajai programinei įrangai dirbančiai su .NET karkasu Windows operacinės sistemos aplinkoje. Šiuo atveju vizualizacijos sistema būtų kuriama kaip standartinė taikomoji programa. Kadangi su SCADA sistemomis labai dažnai dirba ir programuotojo patirties neturintys asmenys, tai turi būti

galimybė naudojantis tik dialogo langais išgauti pagrindinį ir dažniausiai naudojamą vizualizacijos sistemų funkcionalumą. Kodas šiuo atveju būtų generuojamas automatiškai. Šiam tikslui pasiekti reikėtų kurti Visual Studio paketo išplėtimo komponentus vadinamus VSPackage [14].

4.4. Grafinė vartotojo sąsaja

Grafinė vartojo sąsaja yra vienas iš svarbiausių elementų vizualizacijos sistemose. Daugelis SCADA sistemų vartotojo sąsajai kurti naudoja taškinė grafiką ir firmines uždaro kodo technologijas [18]. Naujausia Microsoft grafinių vartotojo sąsajų kūrimo technologija WPF turi visas savybes, kurios būtinos vizualizacijos sistemose. Priedo ši technologija turi papildomas galimybes, leidžiančias kurti vektorinės grafikos piešinius, dokumentus, palaiko daugiakalbiškumą, turi bibliotekas, skirtas apdoroti ir vaizduoti įvairaus formato paveikslėlius, video medžiagą [11].

Vektorinė grafika

Taškinė grafika turi labai didelį trūkumą, kuomet tenka naudoti skirtingo dydžio ir rezoliucijos monitorius, įgyvendinti priartinimo/atitolinimo funkcijas. Dėl savo savybių taškinė grafika praranda kokybę kuomet yra keičiamas jos dydis [18]. Todėl dažniausiai vizualizacija yra kuriama tik tam tikros rezoliucijos ekranui. Kadangi WPF technologija iš pagrindų buvo kurta vektorinei grafikai, tad ši problema išsisprendžia savaime.

Vartotojo sąsajos elementų kūrimas ir išdėstymas

Viena iš WPF technologijos ypatybių yra vidinė jos architektūra. Visi grafiniai elementai yra sukurti panaudojant kompozicijos projektavimo pavyzdį. Tai be didelių pastangų leidžia sukurti labai sudėtingas vartotojo sąsajos elementų darinius [16]. Dauguma SCADA sistemų palaiko tik absoliutinį elementų išdėstymą, todėl keičiant vizualizacijos lango dydį jo turinys neprisitaiko prie pokyčių. To pasekoje arba dalis vaizdo pasislepia arba iš šonų lieka tušti kraštai. WPF technologija turi visą eilę išdėstymo elementų, kurie padės išspręsti šias problemas. Be abejo kurti prisitaikančią vartotojo sąsają yra ženkliai sudėtingiau, todėl to nenorintys daryti sistemos vartotojai galės pasirinkti patys, kokio tipo išdėstymo sistemą jie nori naudoti (absoliučią ar ne).

Vartotojo sąsajos stilizavimas

Nei viena iš žinomų SCADA sistemų nepalaiko stilių. Pramoniniuose objektuose nėra didelio poreikio kaitalioti vizualizacijos sistemos išvaizdą tam kad būtų gražu, todėl ši savybė nėra prioritetinė šia prasme. Čia svarbiausias yra funkcionalumas. Tačiau stilizavimas gali labai pagelbėti derinant sukurtos vizualizacijos bendrą stilių ir patogumą (vizualizacija turi neerzinti akių nes į ją žiūrima pakankamai daug laiko) arba užsakovui pateikti kelių spalvinių variantų derinius. Operatoriai galėtų išsirinkti juos labiausiai tenkinantį variantą esant skirtingiems patalpos apšvietimams ir pan. Pasinaudojant stiliais nereikia rankiniu būdu keisti dešimčių ar šimtų grafinių elementų išvaizdos. Kita vertus stiliai įgyja visai kitą prasmę, jei vizualizacija kuriam ne pramonei o pvz. pastatų arba namų automatikai. Taigi vartotojo sąsajos stilizavimas padaro sistemą universalesnę.

WPF turi vartotojo sąsajos stilizavimo savybes todėl papildomų pastangų šiam funkcionalumai pasiekti nereikia [11].

Daugiakalbiškumas

Jei vizualizacija kuriama ne vienetiniam projektui o tam tikram produktui, kuris bus parduodamas kitose valstybėse kartu su pačia vizualizacija, yra būtinas vartotojo sąsajos daugiakalbiškumas. Vartotojas turi turėti galimybę pakeisti kalbą į vieną iš galimų. Be to turėtų būti galimybė pridėti naujas kalbas jau po sistemos sukūrimo, jei keičiasi vartotojo poreikiai. Daugiakalbiškumo funkcionalumas yra realizuotas pačioje WPF technologijoje, todėl papildomai nieko kurti nereikia.

Parametrų susiejimas su išoriniais duomenimis

Kuriant vizualizacijas neišvengiamai tenka susieti kai kuriuos vartotojo sąsajos elementų parametrus su tam tikrai išoriniais duomenimis. Pvz. priklausomai nuo tam tikros kintamojo reikšmės valdymo įtaise rodyti atitinkamą tekstą, keisti teksto spalvą arba vaizduoti bei leisti keisti kintamojo reikšmę duomenų įvedimo laukelyje. Kadangi siekiame mes naudoti standartinę vartotojo sąsajos kūrimo technologiją, tam tikslui galima panaudoti dekoratoriaus projektavimo pavyzdį, kuris leidžia pridėti prie objekto papildomą funkcionalumą dinamiškai [16] arba naudoti klasių paveldėjimą generuojant sistemos kodą. Tačiau naudojant šią metodiką mes negalime panaudoti kitų grafinės vartotojo sąsajos redaktorių, nes turime kurti ne standartinio tipo elementus o išvestinio, kurie buvo sukurti paveldėjimo metodu. Siekiant

išvengti šių trūkumų geriau yra panaudoti WPF technologijoje realizuotas duomenų susiejimo galimybes. Jos leidžia bet kurį grafinio elemento parametrą susieti su:

a. bet kurio objekto atributu ar metodo rezultatu
b. duomenimis XML faile
c. duomenimis duomenų bazėje (su ADO.NET objektu)
d. su kito grafinio elemento parametru

Duomenų susiejimas gali būti vienkryptis arba daugiakryptis, papildomai galima panaudoti duomenų valiavimo funkcionalumą [11][6]. Naudodami standartines WPF technologijos galimybes išsaugome galimybę kurti pačią vartotojo sąsają kitame įrankyje ir vėliau ją įkelti į vizualizacijos sistemą.

4.5. Žiniatinklio ar paprasta programa

Naudoti žiniatinklio programas stacionarioje darbo vietoje nėra tikslo. Kita vertus žiniatinklio programos turi nemažai apribojimų ir negali lygintis sparta ir vartotojo sąsajos galimybėmis su stacionariomis programomis. Standartinės žiniatinklio programų galimybės (HTML, JavaScriptm, CSS) yra nepakankamos vizualizacijos sistemoms kurti. Nors norint pasiekti žiniatinklio programų universalumą ir pernešamumą dažnai naudojama Java Applet technologija [12]. Išties tai nėra visiškai gryna žiniatinklio programa. Tiesiog specialiai sukurta Java taikomoji programa dirba interneto naršyklės aplinkoje. Vartotojui nereikia galvoti apie programos diegimą ar išsaugojimą kompiuteryje.

Atsiradus .NET ir WPF technologijoms riba tarp stacionarių ir žiniatinklio programų Windows aplinkoje pamažu nyksta. Taikomoji programinė įranga, sukurta .NET karkasui, nereikalauja ypatingo diegimo. Ji gali veikti tiek viena nepriklausomai tiek Windows interneto naršyklės Internet Explorer aplinkoje. Jei WPF technologiją norima naudoti kitose operacinėse sistemose ar interneto naršyklėse, galima pasitelkti Microsoft Silverlight technologiją, kuri gali vykdyti apriboto WPF funkcionalumo programas [6][11].

Nors dauguma darbo vietų pramoninėje automatikoje yra stacionarios ir nereikalauja žiniatinklio programų savybių, tačiau šis papildomas funkcionalumas labai pageidautinas. Panaudojant žiniatinklio programas paprastu būdu galima leisti valdyti procesą per internetą, leisti kitiems suinteresuotiems asmenims jį stebėti skirtingose darbo vietose.

4.6. Sistemos praplečiamumas

Norint praplėsti daugumos SCADA sistemų funkcionalumą tenka rašyti kodą tam tikra skriptavimo kalba, naudoti ActiveX arba COM komponentus. Kadangi mes vizualizacijos sistemoms kurti naudojame taikomųjų programų kūrimo metodiką (įskaitant ir kodo generaciją), tai šiuo atveju skriptavimo kalbą pakeičia viena iš gerai žinomų ir turinčių dideles galimybes .NET karkaso programavimo kalbų (C#, VB.NET, C++). Be to į tokią sistemą nesudėtingai galima integruoti bet kuriuos .NET karkasui sukurtus komponentus.

5. REIKALAVIMŲ SPECIFIKAVIMAS

5.1.1. Projekto varovai (Project Drivers)

Projekto kūrimo pagrindas (pagrindimas)

SCADA sistemos nėra naujiena ir naudojamos pramoninėje automatikoje jau ne pirmą dešimtmetį, tad jų pasiūla rinkoje pakankamai didelė. Šiuo metu praktiškai kiekviena didesnė automatikos įrangą gaminanti įmonė gali pasiūlyti ir savo vizualizacijos programinę įrangą, kuri be abejo geriausiai yra suderinama su pačios kompanijos gaminama produkcija. Tokia situacija susiklostė todėl, kad dar praeitame dešimtmetyje, kuomet standartizacija nebuvo toli pažengusi šiame rinkos segmente, kiekvienas automatizacijos įrangos gamintojas kūrė savo specifinę įrangą ir komunikacijos protokolus, kuriais buvo galima su ja komunikuoti. Jeigu projekte buvo naudojama tam tikro gamintojo įranga, tuomet užsakovas turėdavo rinktis ir to paties gamintojo SCADA sistemą, nes kitos sistemos nesugebėdavo „susikalbėti“ su įrenginiais. Tačiau iki šios dienos situacija ganėtinai pasikeitė. Jau sukurtas ne vienas standartas komunikacijos protokolams bei SCADA sistemoms (OPC, Modbus, DeviceNet, CANopen, Profibus ir t.t). OPC standartų šeima standartizavo ne tik sąsają su išoriniais įrenginiais bet ir įgalino kelių gamintojų skirtingas SCADA sistemas perduoti duomenis viena kitai. To pasekoje atsirado ne viena kompanija, kuri užsiima vien tik SCADA sistemų kūrimu.

SCADA sistemų rinka yra gerokai mažesnė ir inertiškesnė nei kitų plačiam vartotojų ratui skirtų sistemų, todėl naujovės, kurias jau senai matome ir naudojame visuotinai paplitusiuose programiniuose paketuose, sunkiau prigyja vizualizacijos sistemose. Tai sąlygoja didelės tokių sistemų kūrimo ir realizavimo išlaidos bei faktas, jog šios sistemos turi būti palaikomos ir aptarnaujamos dešimtmetį ir ilgiau. Per šį laiko tarpą labai didelius pinigus kainuojantys automatikos projektai turi atsipirkt ir duoti pelną. Iš kitos pusės SCADA sistemų

kūrimo ir palaikymo kaštai dideli, nes šios programinės įrangos gamintojai kuria ją nuo pat pamatų patys. Daugumos nuo seno gyvuojančių sistemų architektūra yra jau pasenusi, monolitinė ir ganėtinai iškraipyta įvairių pakeitimų ir tobulinimų. Naujai kuriamos sistemos sėkmės garantas gali būti naujų technologijų panaudojimas, įvairių komponentų pakartptinis panaudojimas, lengva ir grakšti architektūra su gerai apgalvotu praplečiamumo modeliu.

Sistemos tikslai (paskirtis)

SCADA sistemas galima vadinti labai specializuotomis taikomosiomis programomis. Su pastarosiomis jos turi labai daug panašumų: langai, mygtukai, duomenų įvedimo laukai, įvairūs grafiniai elementai ir t.t. Šio vieno iš projekto etapų (magistro darbo) tikslas yra panaudojant modernias firmos Microsoft technologijas, skirtas taikomosioms programoms kurti (Visual Studio, WPF, .NET), sukurti eksperimentinę vizualizacijos sistemą, kuri leistų išbandyti ir pademonstruoti technologijų galimybes.

5.1.2. Užsakovai, pirkėjai ir kiti sistema suinteresuoti asmenys

Planuojama sistema yra eksperimentinė ir skirta įmonės vidaus poreikiams. Todėl tiek užsakovas tiek pirkėjas yra tas pats asmuo – gamybos direktorius Žilvinas Savickis. Kiti sistemos kūrimą įtakojantys asmenys – magistrantas Algirdas Mockus bei KTU elektros ir valdymo inžinerijos fakulteto mokslų daktaras Darius Ezerskis.

5.1.3. Vartotojai

Visus eksperimentinės sistemos vartotojus galima priskirti vienai grupei:

1. Proceso stebėtojai arba operatoriai
    a. Vartotojo kategorija – inžinierius (gali būti ir mokinys)
    b. Vartotojo sprendžiami uždaviniai – proceso stebėjimas ir esant reikalui valdymo sprendimų priėmimas.
    c. Patirtis dalykinėje srityje – įprastas darbuotojas arba srities specialistas
    d. Patirtis informacinėse technologijose – gebėjimas dirbti kompiuteriu bei tam tikromis programomis

5.2. Projekto Apribojimai

5.2.1. Įpareigojantys apribojimai

Apribojimai sprendimui

- Vizualizacijų grafinė dalis turi būti aprašoma XAML kalba.
- Sistema turi būti kuriama .NET platformos pagrindu C# programavimo kalba
- Sistema turi būti kuriama su Visual Studio integruota programinės įrangos kūrimo aplinka (IDE)

Diegimo aplinka

Pačios vizualizacijų kūrimo sistemos (kuri ir yra pagrindinis šio projekto tikslas) diegimo aplinka nėra kokiomis nors savybėmis ypatinga.

Bendradarbiaujančios sistemos

Vizualizacijos sistema turi bendrauti su daugybe išorinių sistemų. Kūrimo metu netgi ne visos jos yra žinomos. Bendradarbiaujančias sistemas galima suskirstyti į šias grupes:

a. Automatizuoto valdymo įrengimai. Tai įvairūs programuojami loginiai valdikliai, dažnio keitikliai, protokolų keitikliai ir kiti įrenginiai, tiesiogiai dalyvaujantys ar glaudžiai susijusią tam tikro proceso valdyme. Duomenų apsikeitimas su šia įranga vyksta per pramoninius komunikacijos protokolus. Būtinai turi būti galimybė pridėti naujų protokolų palaikymą sistemos eksploatacijos metu.
b. OPC (OLE for Prorocess Control) serveriai.
c. Duomenų saugyklos. Turimos omenyje įvairios DBVS (duomenų bazių valdymo sistemos)
d. Gamybos planavimo, tiekimo grandinės ir kitos aukštesniojo lygio sistemos. Su jomis dažniausiai bendraujama per WEB Servisus arba kitus XML standartu pagrįstus protokolus (pvz. B2MML – Business To Manufacturing Markup Language)
e. Išorinės iš anksto dar nežinomos sistemos. Integruojant naujas valdymo sistemas į jau veikiančią automatizacijos sistemą vizualizacijos programoms gali tekti bendrauti su įvairiausiomis jau pasenusiomis arba mažai žinomomis sistemomis. Todėl turėtų būti palikta galimybė aukšto lygio IT specialistams sudaryti naujus ryšius naudojant numatytas praplėtimo sąsajas ar pan.

Komerciniai specializuoti programų paketai

- Planuojama sistemą kurti komercinės programinės įrangos kūrimo aplinkos Microsoft VisualStudio pagrindu. Ši sistema buvo kuriama turint omenyje išplečiamumą ir turi net tris plečiamumo mechanizmus, kuriuos panaudojant PĮ kūrimą galima automatizuoti. Projektuojant vizualizacijų kūrimo sistemą reikėtų turėti omenyje, kad ateityje ją gali tekti migruoti į kitą panašaus tipo į VisualStudio platformą (pvz. atviro kodo SharpDevelop).

Sistemos kūrimo terminai

Pirmoji nepilno funkcionalumo eksperimentinė sistemos versija turi būti sukurta iki 2008 metų spalio mėnesio. Tuomet sistema bus pristatyta įmonės darbuotojams ir susijusiems specialistams, bus vertinama sistemos koncepcija ir perspektyvumas.

Sistemos kūrimo biudžetas

Bandomajai sistemos versijai paruošti nuspręsta skirti iki 20 000 Lt.

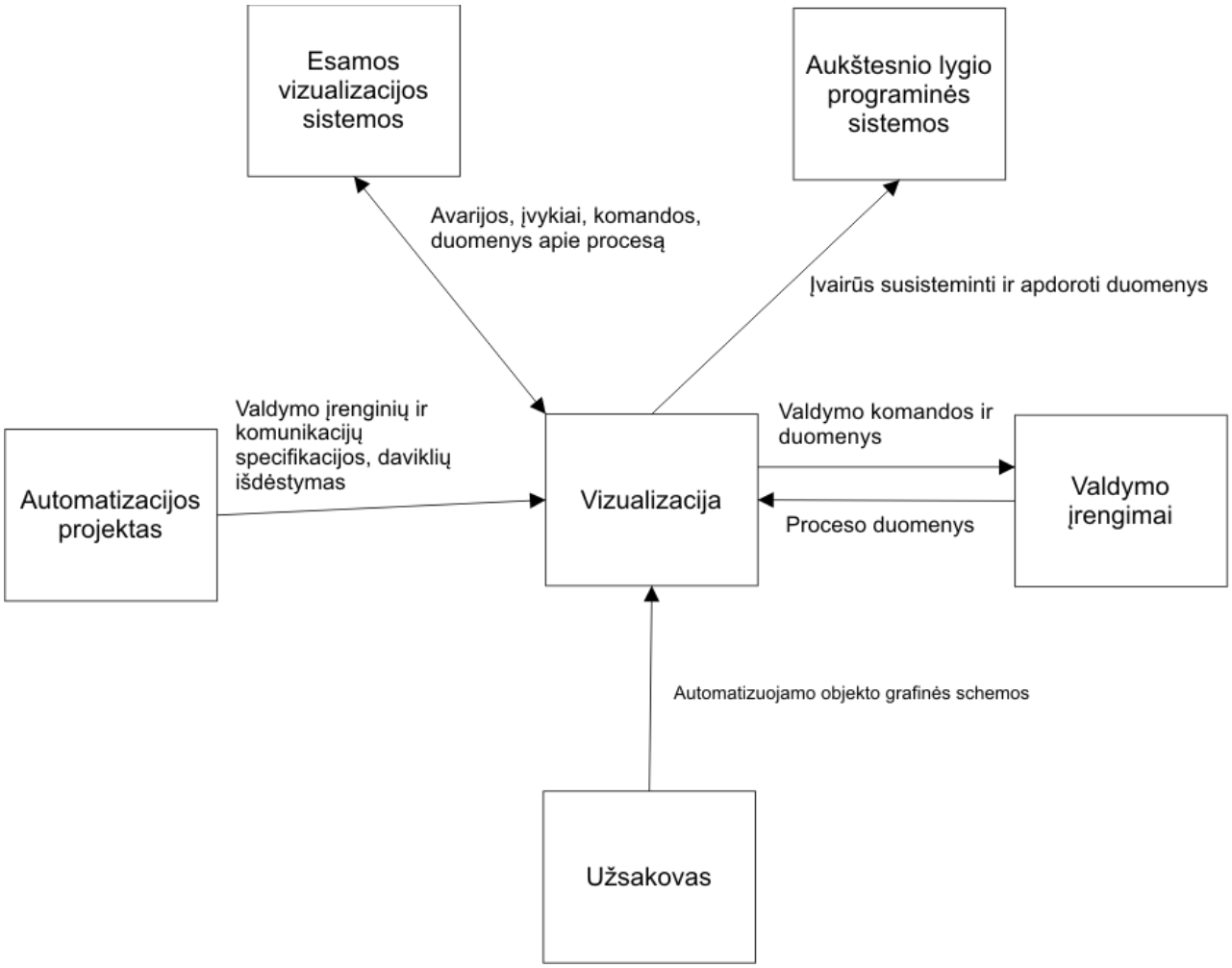
5.2.2. Svarbūs faktai ir prielaidos

- Sistema dirbs įvairių sugebėjimų kompiuterių srityje vartotojai. Darbo aplinka turi būti pritaikyta tiek vieniems tiek kitiems.
- WPF vartotojo sąsajos technologija yra pakankamai jauna ir nuolat vystosi bei tobulėja. Vystymasis neišvengiamai keičia programavimo sąsajas (API), todėl kur tik įmanoma reikia naudoti MVC projektavimo pavyzdį.

5.3. Funkciniai reikalavimai

5.3.1. Veiklos sudėtis (The scope of the work)

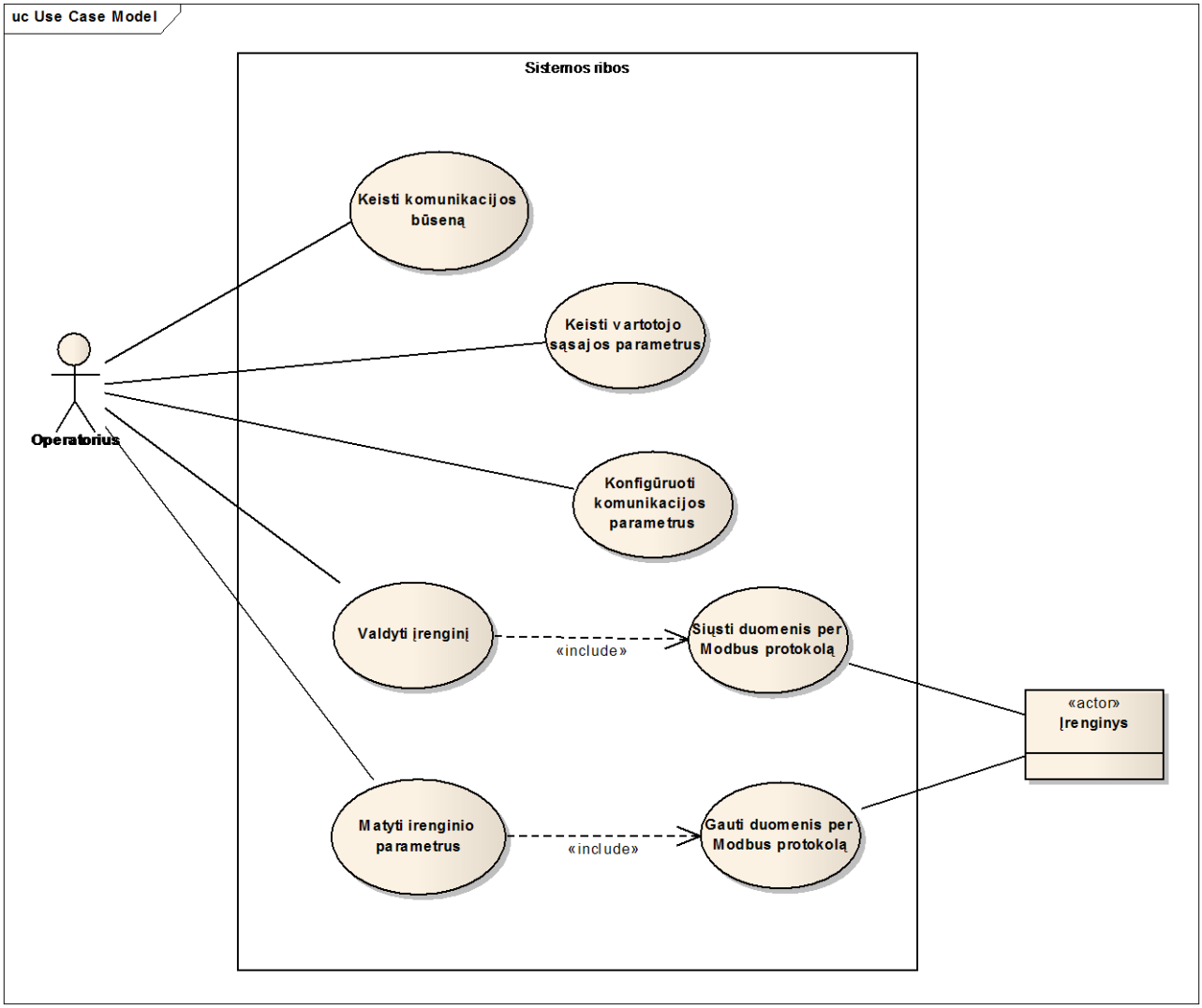
Veiklos kontekstas (pateikiama konteksto diagrama)



Pav. 17 Veiklos konteksto diagrama

5.3.2. Sistemos sudėtis (The scope of the product)

Sistemos ribos



Pav. 18 Panaudojimo atvejų diagrama

Panaudojimo atvejų sąrašas

1. PANAUDOJIMO ATVEJIS: Keisti komunikacijos būseną

Vartotojas/Aktorius: Operatorius

Aprašas: Apima prisijungimą prie įrenginio, atsijungimą, komunikacijos būsenos stebėjimą

Prieš sąlyga: Nurodyti prisijungimo parametrai;

Sužadinimo sąlyga: Nuspaudžiamas mygtuką „prisijungti“ arba atsijungti

Po-sąlyga: Atliekamas prisijungimo arba atsijungimo veiksmas

2. PANAUDOJIMO ATVEJIS: Keisti vartotojo sąsajos parametrus

Vartotojas/Aktorius: Operatorius

Aprašas: Apima įvairių programos vartotojo sąsajos parametrų nustatymą: GUI dydį, stilių, efektų naudojimą.

Prieš sąlyga: Atidarytas nustatymų langas.

Sužadinimo sąlyga: Išsirenkamas ir keičiamas parametras

Po-sąlyga: Pakeisti parametrai išsaugomi ir pritaikomi

3. PANAUDOJIMO ATVEJIS: Konfigūruoti komunikacijos parametrus

Vartotojas/Aktorius: Operatorius

Aprašas: Apima procesą, kurio metu galima nustatyti įvairius susijungimo su įrenginių parametrus

Prieš sąlyga: Atsijungta nuo įrenginio; Atidarytas įrenginio nustatymų langas;

Sužadinimo sąlyga: Išsirenkamas ir keičiamas parametras

Po-sąlyga: Pakeisti parametrai išsaugomi ir pritaikomi

4. PANAUDOJIMO ATVEJIS: Valdyti įrenginį

Vartotojas/Aktorius: Operatorius

Aprašas: Apima procesą, kurio metu vartotojo keičiami duomenys perduodami įrengimui

Prieš sąlyga: Prisijungta prie įrenginio

Sužadinimo sąlyga: Vartotojo sąsajoje keičiamas parametras

Po-sąlyga: Įrenginys gauna pakeistus parametrus

5. PANAUDOJIMO ATVEJIS: Matyti įrenginio parametrus

Vartotojas/Aktorius: Operatorius

Aprašas: Apima procesą, kurio metu duomenys parsiunčiami iš įrenginio ir įvairiomis formomis atvaizduojami vartotojui.

Prieš sąlyga: Prisijungta prie įrenginio

Sužadinimo sąlyga: Atidarytas langas, kuriame vaizduojami duomenys

Po-sąlyga: Parametrai nuskaitomi iš įrenginio

6. PANAUDOJIMO ATVEJIS: Siųsti duomenis per Modbus protokolą
Vartotojas/Aktorius: Įrenginys
Aprašas: Apima procesą, kurio metu duomenys įrašomi į įrenginį
Prieš sąlyga: Prisijungta prie įrenginio.
Sužadinimo sąlyga: Duomenys pasikeitė
Po-sąlyga: Duomenys siunčiami įrenginiui

7. PANAUDOJIMO ATVEJIS: Gauti duomenis per Modbus protokolą
Vartotojas/Aktorius: Įrenginys
Aprašas: Apima procesą, kurio metu duomenys nuskaitomi iš įrenginio
Prieš sąlyga: Prisijungta prie įrenginio.
Sužadinimo sąlyga: Duomenys nuskaitomi ciklišk ai
Po-sąlyga: Duomenys nuskaitomi iš įrenginio

5.3.3. Funkciniai reikalavimai ir reikalavimai duomenims

Funkciniai reikalavimai

Reikalavimas #:	*1*	**Reikalavimo tipas:**	*10*	**Įvykis/panaudojimo atvejis #:**	*7;5*
Aprašymas:	*Matyti ekrane grafiškai atvaizduotus įrenginio parametrus*				
Pagrindimas:	*Proceso stebėjimas*				
Šaltinis:	***Užsakovas***				
Tikimo kriterijus:	*Įrenginio parametrai išdėstyti ant shematiško įrenginio vaizdo*				
Užsakovo tenkinimas:	*5*			**Užsakovo netenkinimas:**	*0*
Priklausomybės:	*Nėra*			**Konfliktai:**	***Nėra***
Papildoma medžiaga:	***Nėra***				
Istorija:	*Užregistruotas 2008 balandžio 15 d.*				

Reikalavimas #:	*2*	**Reikalavimo tipas:**	*10*	**Įvykis/panaudojimo atvejis #:**	*4;6*
Aprašymas:	*Galimybė valdyti įrenginį keičiant jo parametrus*				
Pagrindimas:	*Nuotolinis proceso/įrenginio valdymas*				
Šaltinis:	***Užsakovas***				
Tikimo kriterijus:	*Įrenginys turi reaguoti į siunčiamas komandas*				
Užsakovo tenkinimas:	*5*			**Užsakovo netenkinimas:**	*0*
Priklausomybės:	*Nėra*			**Konfliktai:**	***Nėra***
Papildoma medžiaga:	***Nėra***				
Istorija:	*Užregistruotas 2008 balandžio 15 d.*				

Reikalavimas #:	*3*	**Reikalavimo tipas:**	*10*	**Įvykis/panaudojimo atvejis #:**	*1*
Aprašymas:	*Matyti komunikacijos su įrenginiais būseną*				
Pagrindimas:	*Nėra*				
Šaltinis:	***Užsakovas***				
Tikimo kriterijus:	*Rodoma ar ryšys su įrenginiu yra ar ne*				
Užsakovo tenkinimas:	*5*			**Užsakovo netenkinimas:**	*0*
Priklausomybės:	*Nėra*			**Konfliktai:**	***Nėra***
Papildoma medžiaga:	***Nėra***				
Istorija:	*Užregistruotas 2008 balandžio 15 d.*				

Reikalavimas #:	*4*	**Reikalavimo tipas:**	*10*	**Įvykis/panaudojimo atvejis #:**	*1*
Aprašymas:	*Prisijungti/atsijungti nuo įrenginio*				
Pagrindimas:	*Nėra*				
Šaltinis:	***Užsakovas***				
Tikimo kriterijus:	*Keičiasi komunikacijos būsena*				
Užsakovo tenkinimas:	*5*			**Užsakovo netenkinimas:**	*0*

Priklausomybės:	*Nėra*	**Konfliktai**:	***Nėra***
Papildoma medžiaga:	***Nėra***		
Istorija:	*Užregistruotas 2008 balandžio 15 d.*		

Reikalavimas #:	*5*	**Reikalavimo tipas**:	*10*	**Įvykis/panaudojimo atvejis #**:	*3*
Aprašymas:	*Pasirinkti prie kurio įrenginio jungtis*				
Pagrindimas:	*Nėra*				
Šaltinis:	***Užsakovas***				
Tikimo kriterijus:	*Nėra*				
Užsakovo tenkinimas:	*5*			**Užsakovo netenkinimas**:	*0*
Priklausomybės:	*Nėra*			**Konfliktai**:	***Nėra***
Papildoma medžiaga:	***Nėra***				
Istorija:	*Užregistruotas 2008 balandžio 15 d.*				

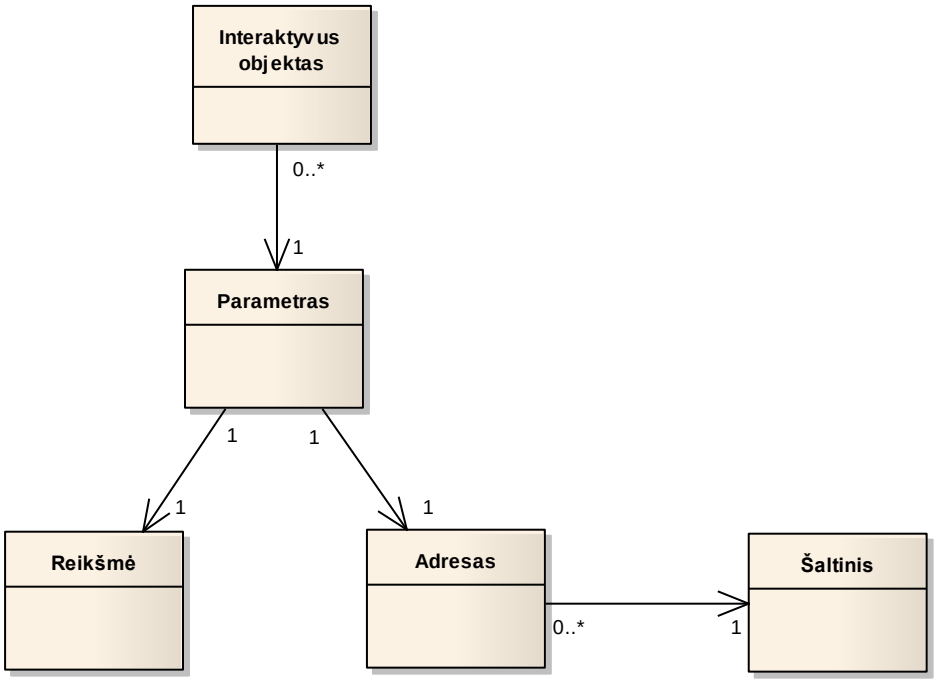
Reikalavimas #:	*6*	**Reikalavimo tipas**:	*10*	**Įvykis/panaudojimo atvejis #**:	*2*
Aprašymas:	*Galimybė keisti vartotojo sąsajos dydį*				
Pagrindimas:	*Keičiasi monitoriaus rezoliucija. Vaizdas per smulkus arba per stambus*				
Šaltinis:	***Užsakovas***				
Tikimo kriterijus:	*Nėra*				
Užsakovo tenkinimas:	*5*			**Užsakovo netenkinimas**:	*0*
Priklausomybės:	*Nėra*			**Konfliktai**:	***Nėra***
Papildoma medžiaga:	***Nėra***				
Istorija:	*Užregistruotas 2008 balandžio 15 d.*				

Reikalavimas #:	*7*	**Reikalavimo tipas**:	*10*	**Įvykis/panaudojimo atvejis #**:	*2*
Aprašymas:	*Galimybė keisti vartotojo sąsajos stilių*				
Pagrindimas:	*Nėra*				

Šaltinis:	***Užsakovas***		
Tikimo kriterijus:	*Nėra*		
Užsakovo tenkinimas:	*5*	**Užsakovo netenkinimas:**	*0*
Priklausomybės:	*Nėra*	**Konfliktai:**	***Nėra***
Papildoma medžiaga:	***Nėra***		
Istorija:	*Užregistruotas 2008 balandžio 15 d.*		

Reikalavimai duomenims

Žemiau pateikiamas pradinis duomenų modelis, kuris pavaizduojamas klasių diagrama.



Pav. 19 Duomenų modelis

5.4. Nefunkciniai reikalavimai

5.4.1. Reikalavimai sistemos išvaizdai (Look and feel)

Vartotojo sąsajai keliami šie reikalavimai:

1. Paprastas (nesudėtingas) panaudojimas
2. Prisitaikymas prie lango dydžio

3. Nejkyri (nedaug spalvų)

5.4.2. Reikalavimai panaudojamumui (Usability)

Pagrindinis reikalavimas panaudojamumui – sistema turi sugebėti naudotis IT išsilavinimo neturintys asmenys. Pagrindinė dalis visų funkcijų turi būti prieinama per aiškią dialogų arba konfigūracijos langų sąsają. Prie sudėtingesnių parametrų tyri būti pridėti paaiškinimai.

5.5. Projekto išeiga (Project issues)

5.5.1. Atviri klausimai (problemos)

Nėra aišku, ar nauja vartotojų sąsajų kūrimo technologija WPF pilnai tenkins savo darbo greitaveika esant ribotiems resursams. Pirmųjų bandomųjų sistemos versijų metu ketinama testuoti šią technologiją, kad būtų galima priimti sprendimą ar ją keisti kita ar ne.

Pagaminti komponentai, kurie gali būti panaudoti

Kadangi grafinio redaktoriaus kūrimas yra pakankamai imlus laikui, galima mėginti pritaikyti jau sukurtus WPF redaktorius

1. Microsoft Cider – tai kodinis redaktoriaus pavadinimas, kurį kompanija Microsoft naudoja Visual Studio sistemoje. Šį komponentą būtų galima panaudoti nemokamai, tačiau tektų sugaišti daug laiko aiškinantis jo architektūrą ir galimybes susieti su savu kodu. Aplamai kyla klausimas, ar šis komponentas turėtų visas reikiamas plečiamumo savybes.
2. Mobiform Software Aurora – tai vartotojo sąsajų kūrimo įskiepis, sukurtas .NET platformai. Jis turi aiškiai dokumentuotą praplėtimo mechanizmą ir yra skirtas būtent integravimui į kitas sistemas. Jei WPF technologija pasiteisintų, tektų rimtai svarstyti šio komponento įsigijimą.

5.5.2. Naujos problemos

Įtaka jau instaliuotoms sistemoms

Diegiant sukurtą sistemą į ta patį kompiuterį taip pat teks įdiegti ir tam tikros versijos Visual Studio paketą. Jei tame pačiame kompiuteryje jau bus įdiegtas kitos versijos Visual Studio paketas, gali kilti nedidelių nesuderinamumų dėl failų tipų surišimų su skirtingomis

Visual Studio versijomis . Visos pastarosios sistemos versijos naudoja tuos pačius failų plėtinius, todėl jie bus atidaromi su vėliausiai įdiegta versija.

Neigiamas vartotojų nusiteikimas

Kadangi kuriama sistemą iš esmės naudoja kitą vizualizacijų kūrimo ideologiją, kurį laiką vartotojai gali jausti nepatogumų, nes teks daug dalykų įsisavinti iš naujo.

Kliudantys diegimo aplinkos apribojimai

Darbui su naujuoju WPF karkasu reikalingi aukštesni kompiuterio našumo parametrai (kokie, bus nustatoma prototipo testavimo metu), todėl dirbant su senesne aparatūra gali kilti diskomforto jausmas dėl vėluojančio ir strikinėjančio vaizdo.

5.5.3. Uždaviniai

Sistemos pateikimo žingsniai (etapai)

Kadangi pirmame sistemos etape bus pagamintas tik ribotų funkcijų prototipas, jis pilnai nepakeis jau naudojamų sistemų. Vartotojų kompiuteriuose, kuriuose bus planuojama testuoti prototipą, bus įdiegta Visual Studio aplinka bei reikalingi įskiepiai. Diegimas esamų sistemų neįtakos.

Vystymo etapai

Sistemą vystyti planuojama trimis etapais:

1. I etapas – prototipo sukūrimas ir technologijų išbandymas. Šio etapo tikslas: išbandyti WPF bei Visual Studio technologijas bei įsikinti, ar jos yra tinkamos užsibrėžtiems tikslams siekti. Šio etapo metu su Visual Studio ir WPF technologijomis bus sukurta demonstracinė vizualizacijos sistemos versija, bus išbandomi kai kurie architektūriniai sprendimai. Šį etapą planuojam įgyvendinti kaip magistrinį darbą.
2. II etapas – pilnai funkcionuojančios demonstracinės versijos sukūrimas. Šiam etapui bus sukurta veikianti sistema su visomis planuojamomis funkcijomis. Tikėtina, kad pristačius platesniam ratui suinteresuotų asmenų atsiras papildomų reikalavimų bei idėjų.

3. III etapas – komercinio produkto sukūrimas. Pirmosios komercinio produkto versijos sukūrimas. Toliau planuojamas nuolatinis produkto tobulinimas ir naujų versijų išleidimas.

5.5.4. Pritaikymas (Cutover)

Naudinga funkcija, kurią planuojama įdiegti į produktą pirmosiomis komercinėmis versijoms yra esamų vizualizacijos sistemų projektų transformavimas į kuriamos sistemos projektus. Tam būtų kuriamas specialus modulis, kuris dialogo langų pagalba padėtų vartotojams įsikelti su kitomis sistemomis kurtas vizualizacijas.

5.5.5. Rizikos

Galimos sistemos kūrimo rizikos

Lentelė 1. Galimos sistemos kūrimo rizikos

Rizikos faktorius	***Tikimybinis įvertinimas (%)***
Reikalavimų specifikacijos pasikeitimai realizavimo fazėje	40
Ateityje gali tekti migruoti sistemą į kitą nei Visual Studio IDE	70
Visual Studio praplėtimo mechanizmas nėra gerai įsisavintas. Nėra aišku ar bus įmanoma realizuoti visą pageidaujamą funkcionalumą per numatytą laiką ir skiriamą biudžetą	10
Nėra aišku ar WPF greitaveika tenkins reikalavimus	20

Atsitiktinumų (rizikų) planas

Lentelė 2. Rizikos faktoriai ir numatomi planai problemoms spręsti

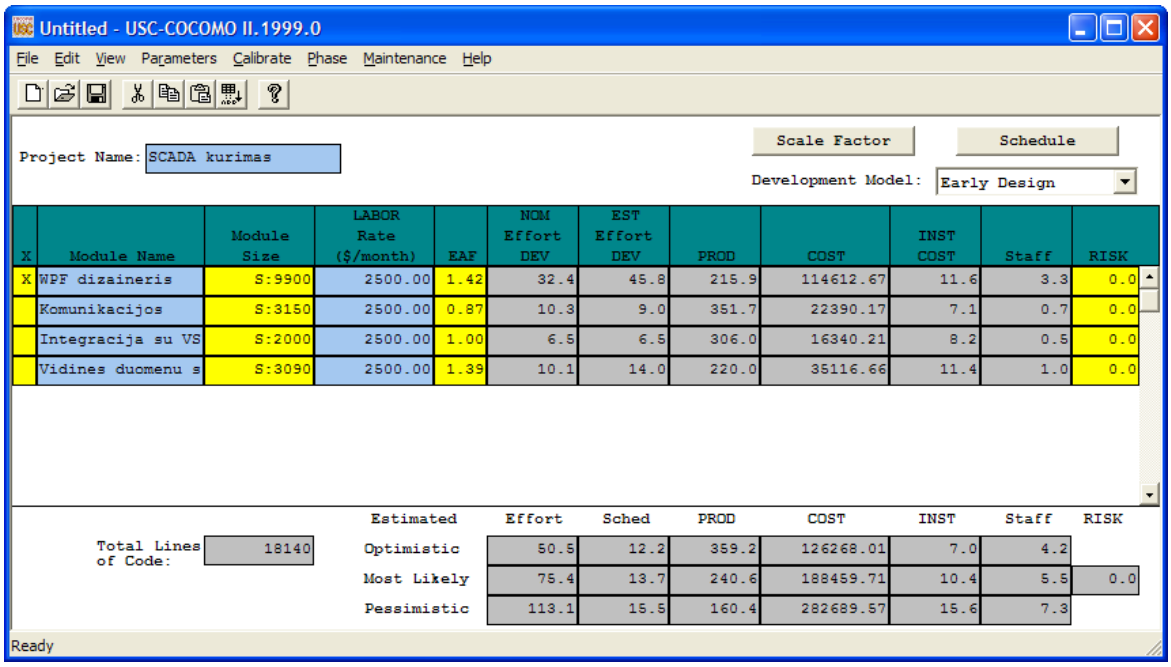
Rizikos faktorius	***Problemos sprendimas***
Reikalavimų specifikacijos pasikeitimai realizavimo fazėje	Sistemą realizuoti kaip susietus įskiepius Visual Studio aplinkai. Modulinė struktūra leis lengviau

	adaptuotis prie besikeičiančių reikalavimų
Ateityje gali tekti migruoti sistemą į kitą nei Visual Studio IDE	Stengtis išlaikyti sistemos architektūrą paprastą, išskaidytą į aiškius komponentus. Kurti sistemos architektūrą orientuojantis netik į Visual Studio bet it įvertinant kitų žinomų IDE ypatybes (galbūt įnešti abstrakcijos lygmenį)
Visual Studio praplėtimo mechanizmas nėra gerai įsisavintas. Nėra aišku ar bus įmanoma realizuoti visą pageidaujamą funkcionalumą per numatytą laiką ir skiriamą biudžetą	Artimiausiu metu paruošti nedidelės apimties sistemos prototipą, kurio pagalba būtų galima išbandyti ir ištirti sąsają su Visual Studio sistema
Nėra aišku ar WPF greitaveika tenkins reikalavimus	Artimiausiu metu paruošti nedidelės apimties sistemos prototipą, kurio pagalba būtų galima išbandyti ir ištirti WPF sistemos greitaveiką ir tam reikalingus kompiuterio resursus

5.5.6. Kaina

Preliminarūs kainos ir reikalingo laiko sistemos sukūrimui duomenys buvo gauti panaudojant COCOMO modelį. Šie skaičiavimai buvo atliekami galutinei planuojamai sistemai. Reziumė:

1. Sistemai sukurti reikėtų parašyti apie 18140 kodo eilučių
2. Vienas žmogus tai padaryti užtruktų apie 75 mėn (siūloma 6 žmonių komanda – apie 13 mėn)
3. Jei traktuosime, kad programuotojui mokame apie 2500lt/mėn, tai sistemą sukurti kainuotų apie 188459lt
4. Daugiausiai kainuotų WPF grafinio redaktoriaus sukūrimas. Todėl verta apsvarstyti jau sukurto komponento įsigijimą.



Pav. 20 Pagrindinis COCOMO programos langas su paskaičiavimų duomenimis

5.5.7. Vartotojo dokumentacija ir apmokymas

Kadangi kol kas kuriamas tik sistemos prototipas, kuris bus bandomas kelių specialiai tam parinktų vartotojų, dėl sąnaudų taupymo dokumentacija rašoma nebus. Vartotojams visos reikalingos žinios bus perduotos žodžiu. Be to sistemą vartotojai bandys kartu su jos kūrėjais.

6. ARCHITEKTŪROS SPECIFIKACIJA

6.1. Architektūros pateikimas

Sistemos architektūra yra pateikiama UML 2 standartą atitinkančių diagramų pavidalu bei prie kai kurių pridedami žodiniai aprašymai jei tai yra reikalinga.

Išsamiam sistemos architektūros pateikimui naudojami šie pagrindiniai UML vaizdai: panaudojimo atvejų, statinis, dinaminis, išdėstymo. Vaizdų sudėtis:

- Panaudojimo atvejų vaizdas – panaudojimo atvejų diagrama
- Statinis vaizdas – sistemos paketų diagrama, kiekvieno paketo klasių diagrama
- Dinaminis vaizdas – sąveikos, būsenų, veiklos diagramos.

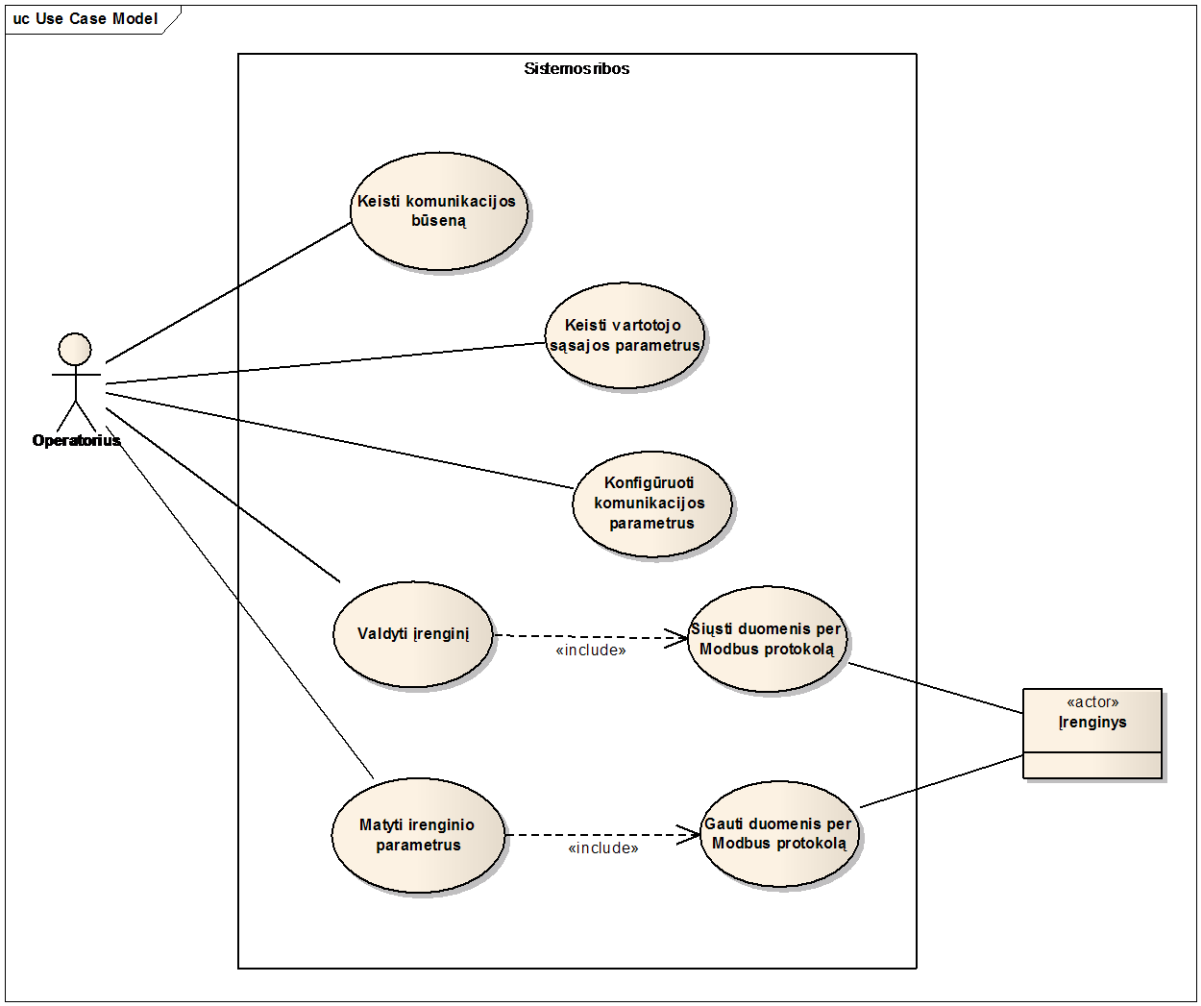
- Išdėstymo – išdėstymo diagrama

6.2. Architektūros tikslai ir apribojimai

Šio projekto etapo tikslas yra panaudojant modernias firmos Microsoft technologijas, skirtas taikomosioms programoms kurti (Visual Studio, WPF, .NET), sukurti eksperimentinę vizualizacijos sistemą. Projekto tikslas ir idėja savaime sukuria ne vieną apribojimą sistemos architektūrai. Sistema turi būti kuriama su MS Visual Studio taikomųjų programų kūrimo įrankiu ir naudoti tik pastarosios galimybes. Taip pat apibrėžta, kad vizualizacijos turi būti kuriamos WPF technologijos pagrindu. Bendra projekto idėja – sukurti vizualizacijos sistemą perpanaudojant kuo daugiau komponentų ir technologijų – automatiškai skatina pakartotinį kodo ar komponentų panaudojimą.

Sistemos architektūra bus kuriama naudojant Sparx Enterprise Architect UML modeliavimo aplinką.

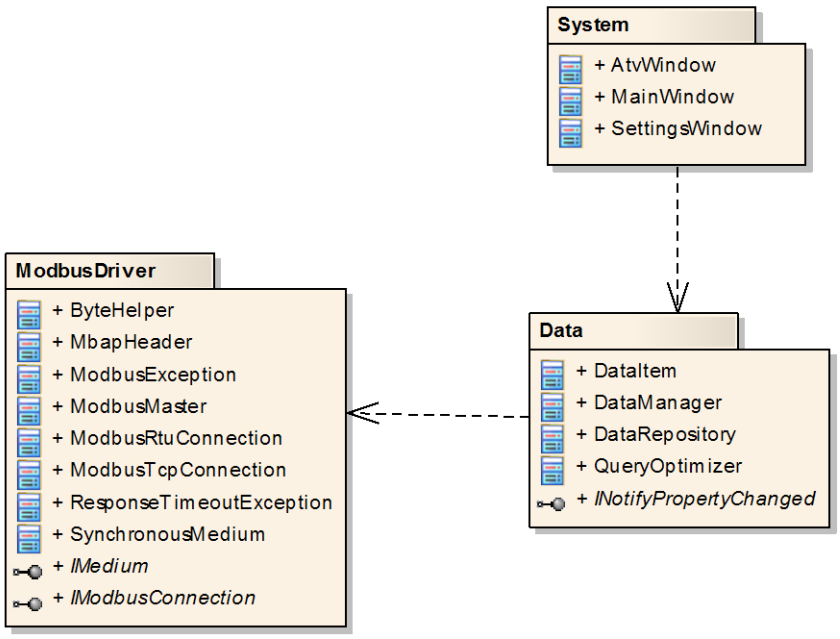
6.3. Panaudojimo atvejų vaizdas



Pav. 21 Panaudojimo atvejų diagrama

6.4. Sistemos statinis vaizdas

6.4.1. Apžvalga

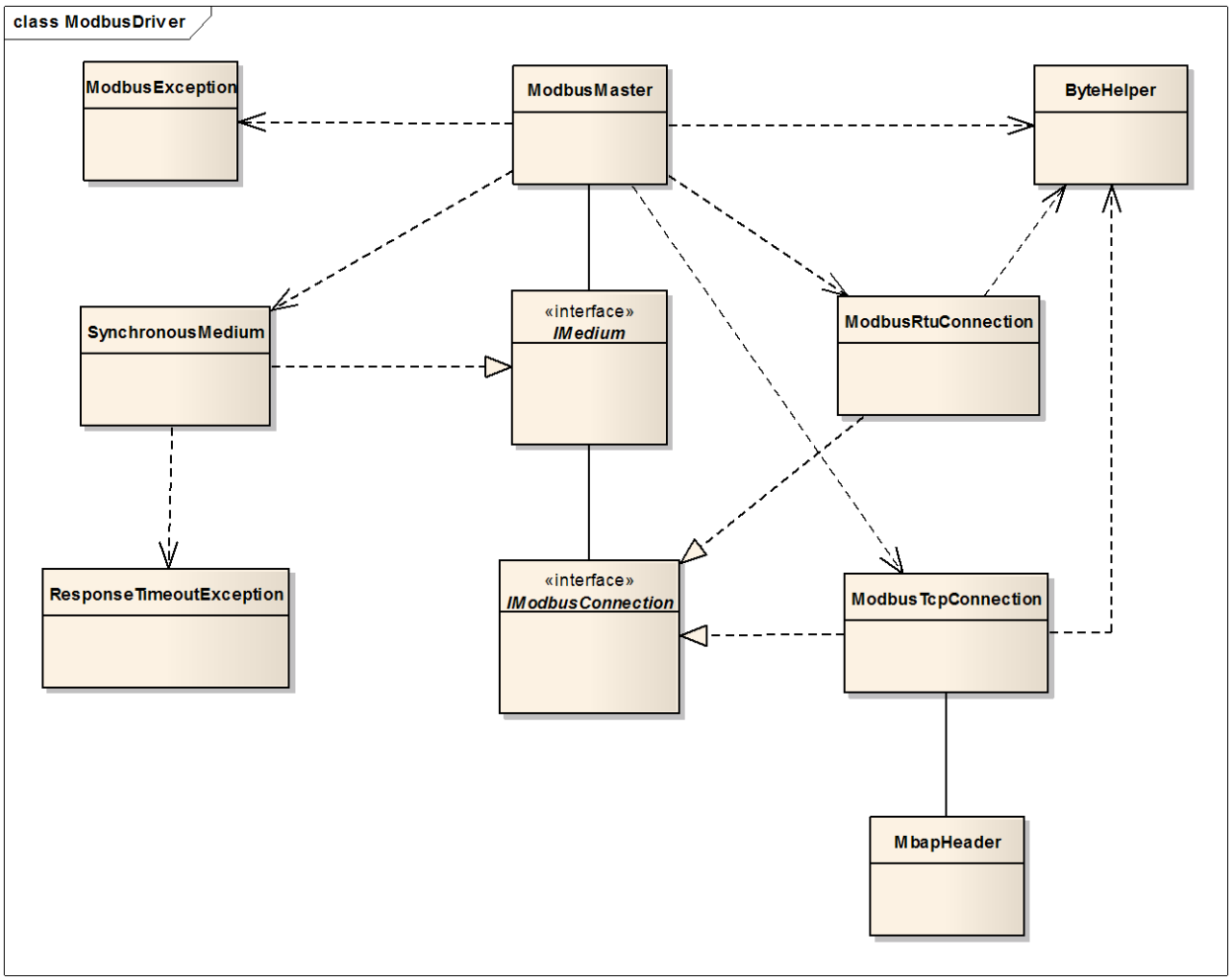


Pav. 22 Sistemos išskaidymas į paketus

6.4.2. Paketų detalizavimas

Paketas „ModbusDriver“

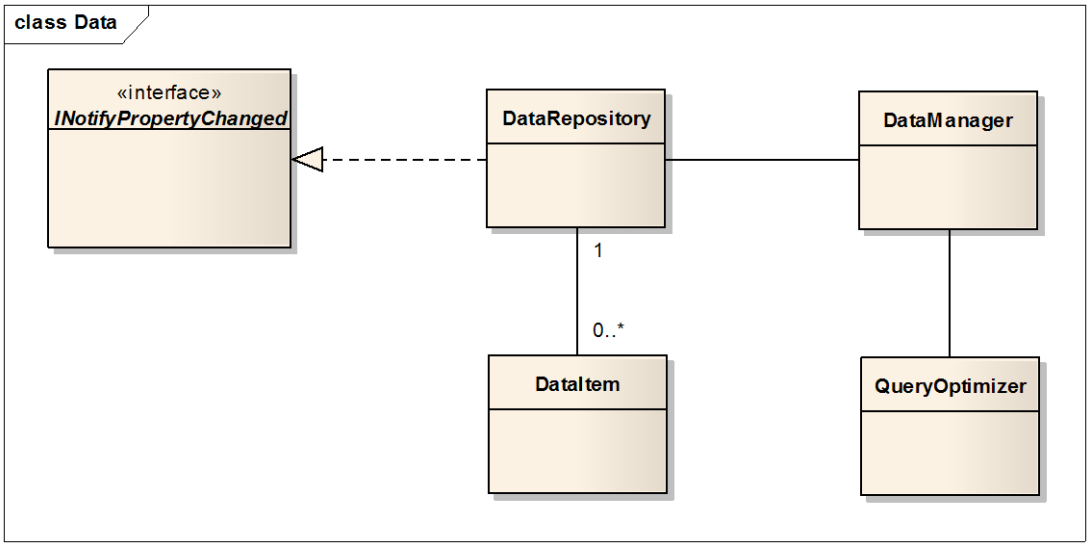
Šis paketas apima sistemos dalį, kuri atsakinga už komunikacijas su išoriniais įrengimais ar kitais duomenų šaltiniais per pramoninį MODBUS protokolą. Į šį paketą įeina visos komunikacijos tvarkyklės.



Pav. 23 Paketo "ModbusDriver" klasių diagrama

Paketas „Data“

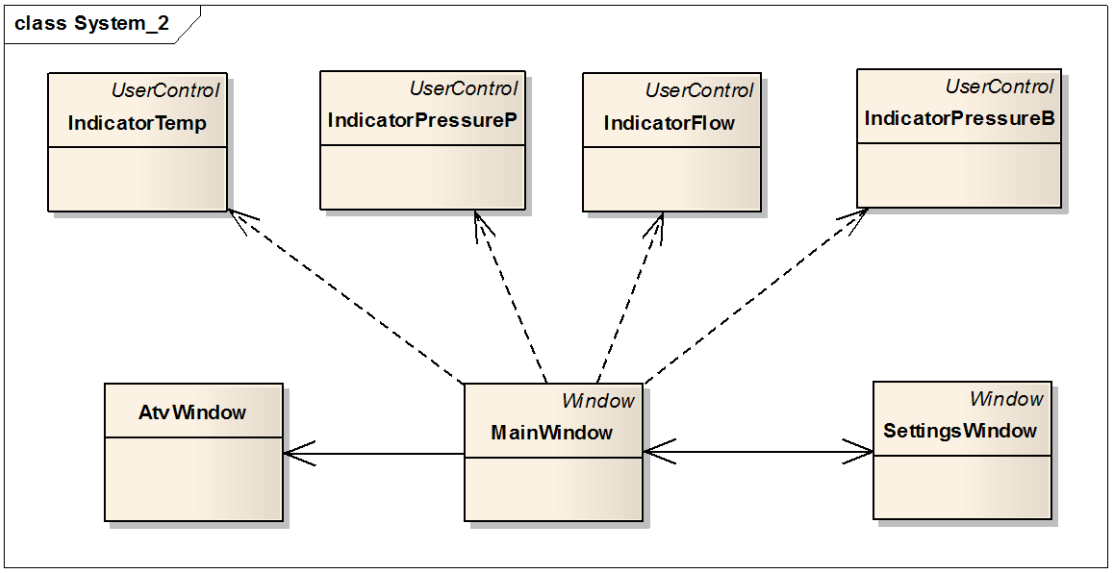
Šiame pakete randasi duomenų nuskaitymo ir valdymo klasės.



Pav. 24 Paketo "Data" klasių diagrama

Paketas „System“

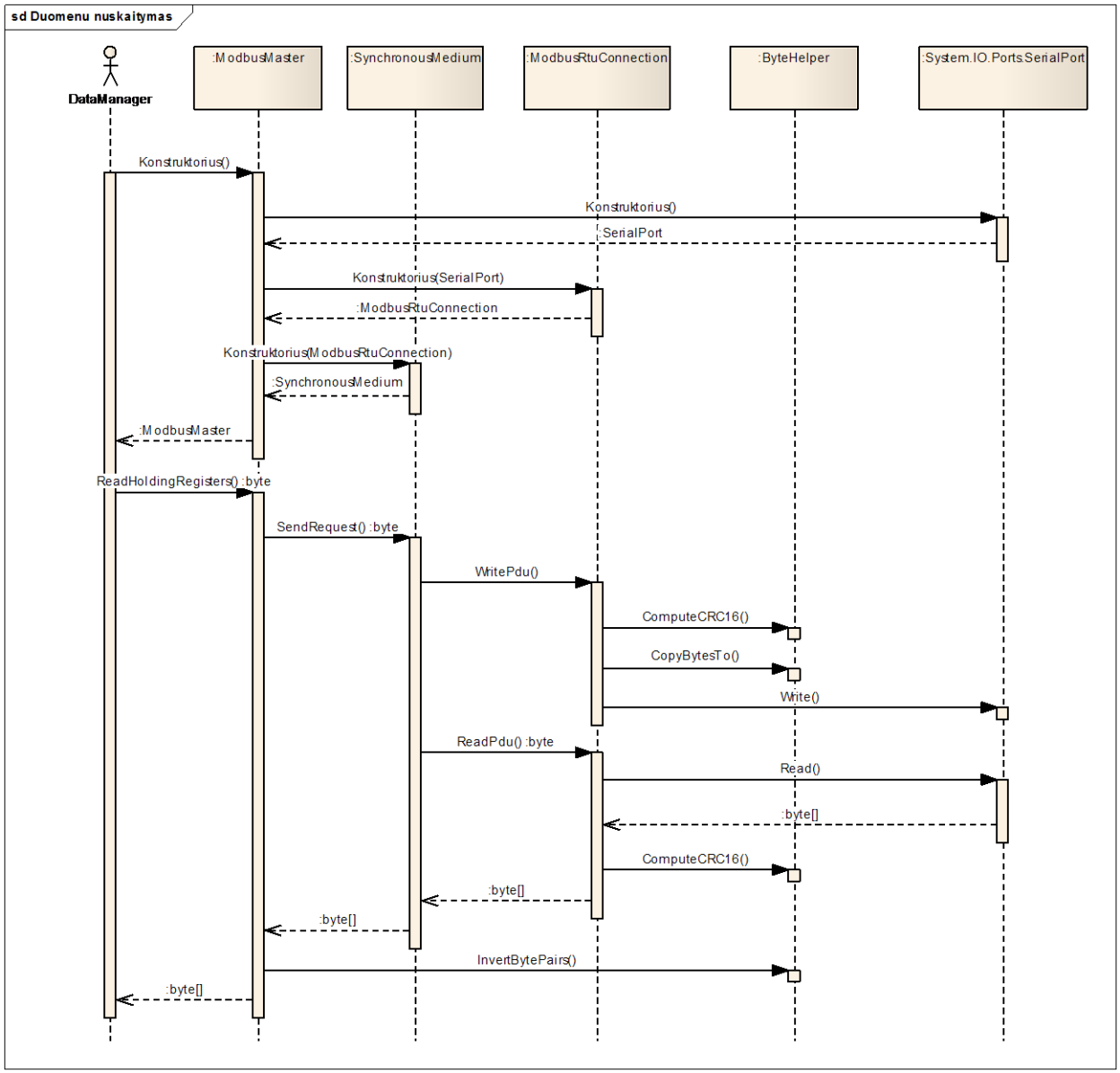
Šiame pakete randasi bazinės sistemos vartotojo sąsajos klasės.



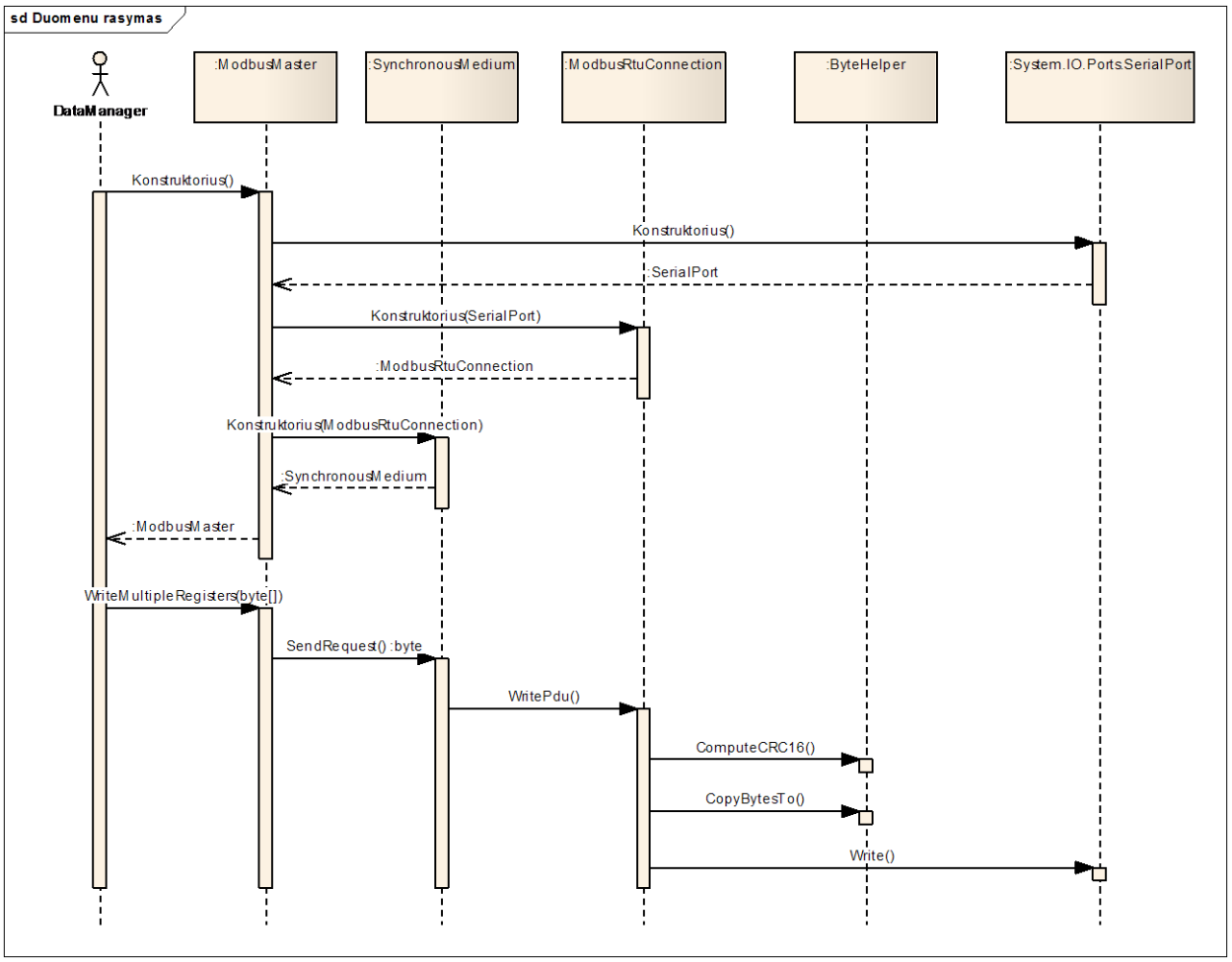
Pav. 25 Paketo "System" klasių diagrama

6.5. Sistemos dinaminis vaizdas

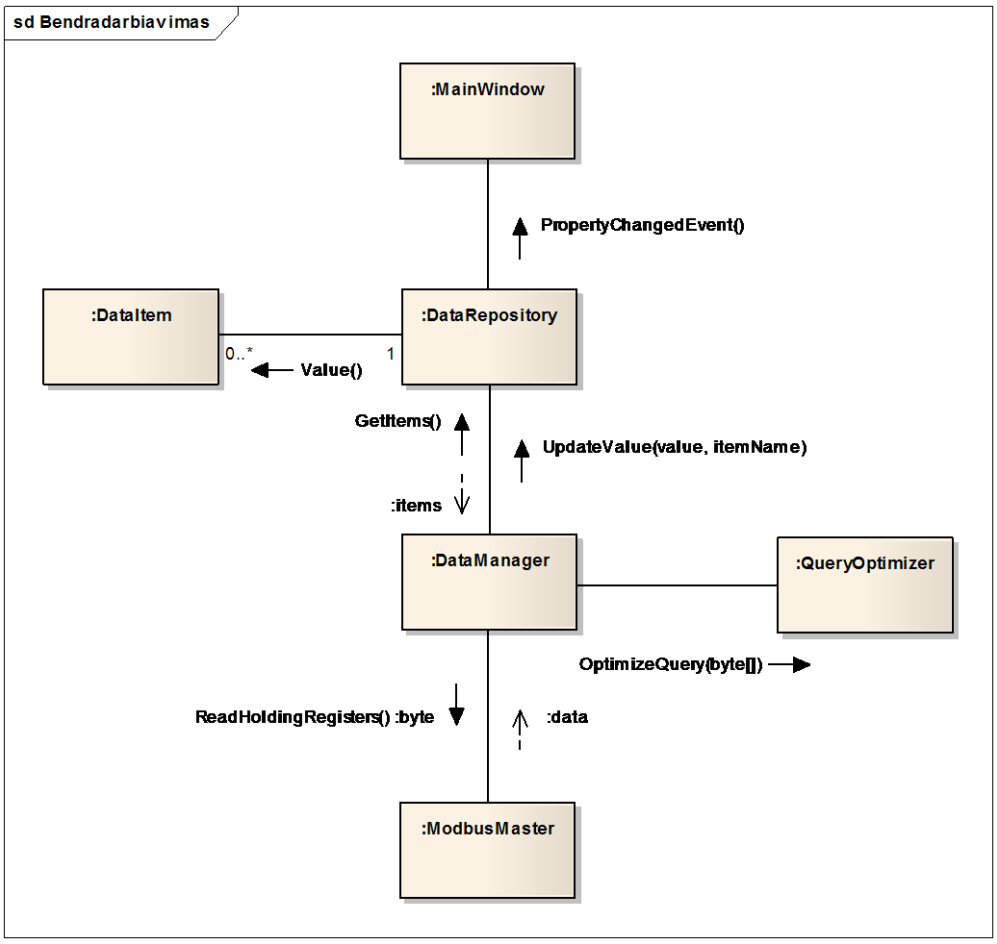
6.5.1. Abstrakčios sąveikos diagramos



Pav. 26 Sekų diagrama "Duomenų užklausimas ir nuskaitymas"

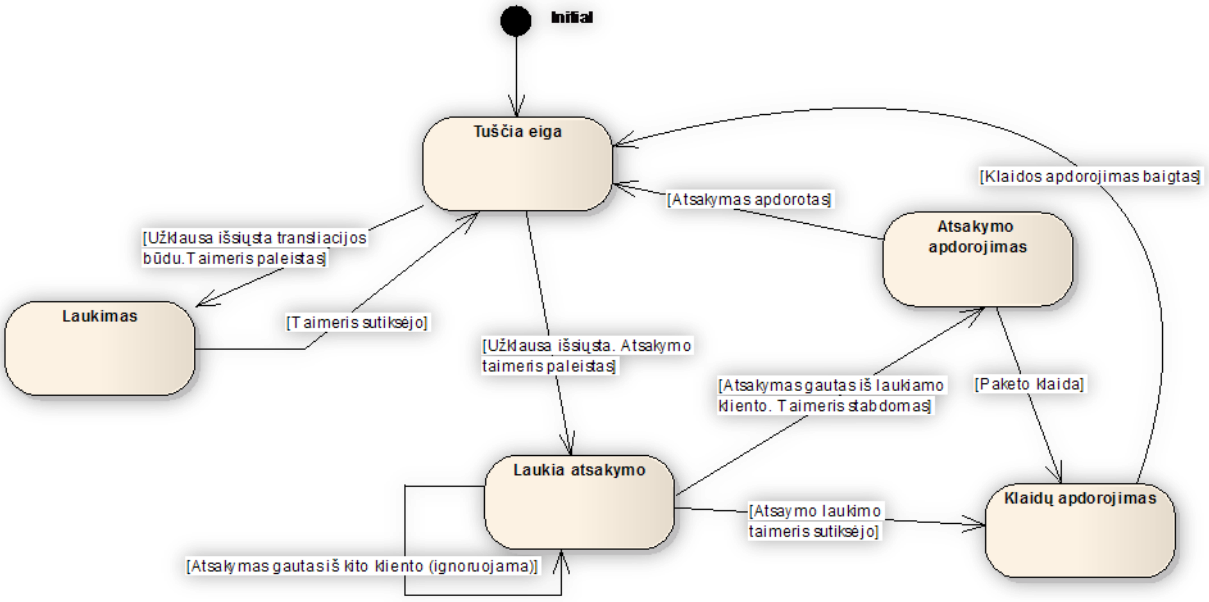


Pav. 27 Sekų diagrama "Valdymo komandų siuntimas"



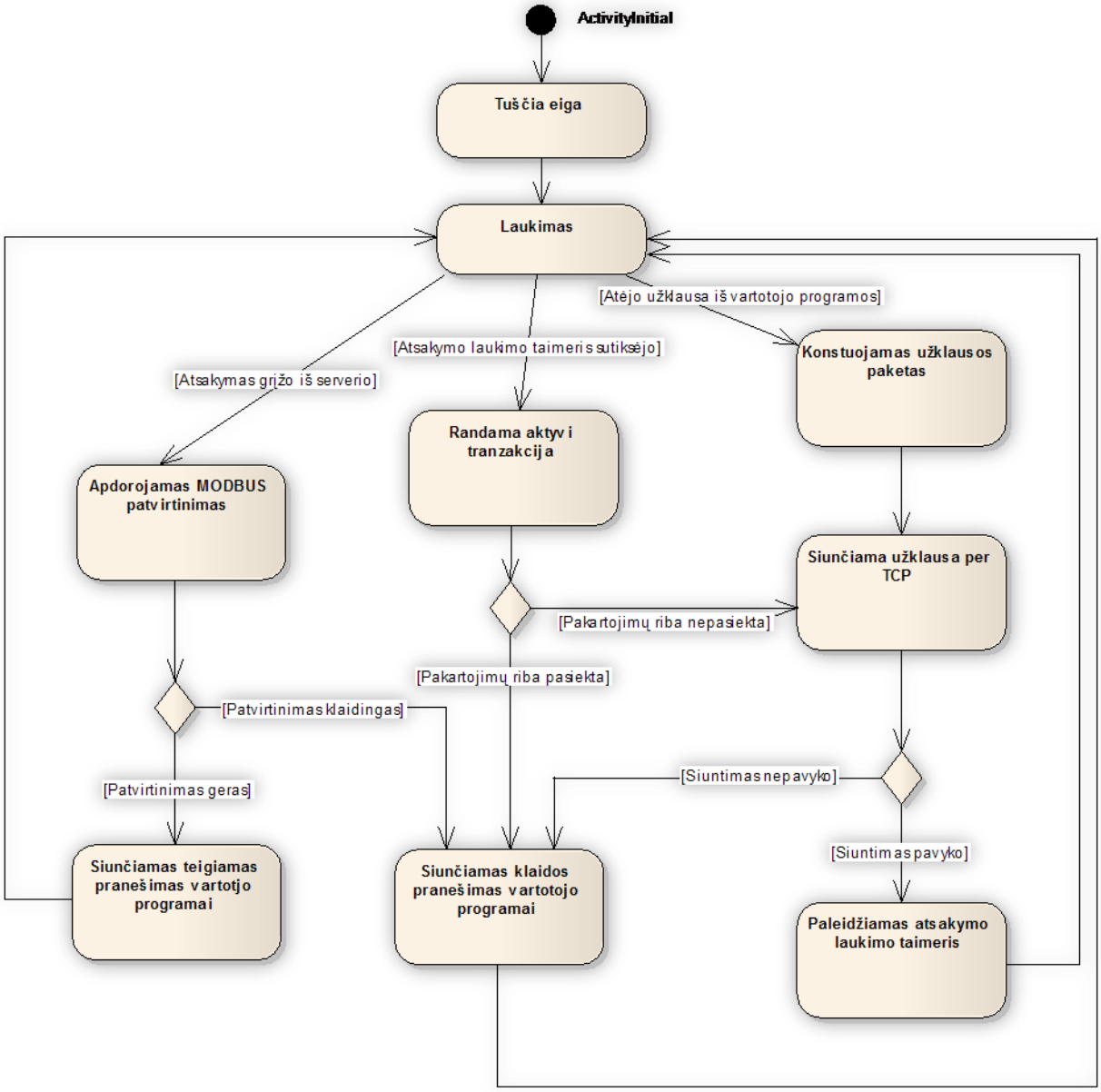
Pav. 28 Duomenų nuskaitymo bendradarbiavimo diagrama

6.5.2. Būsenų diagramos

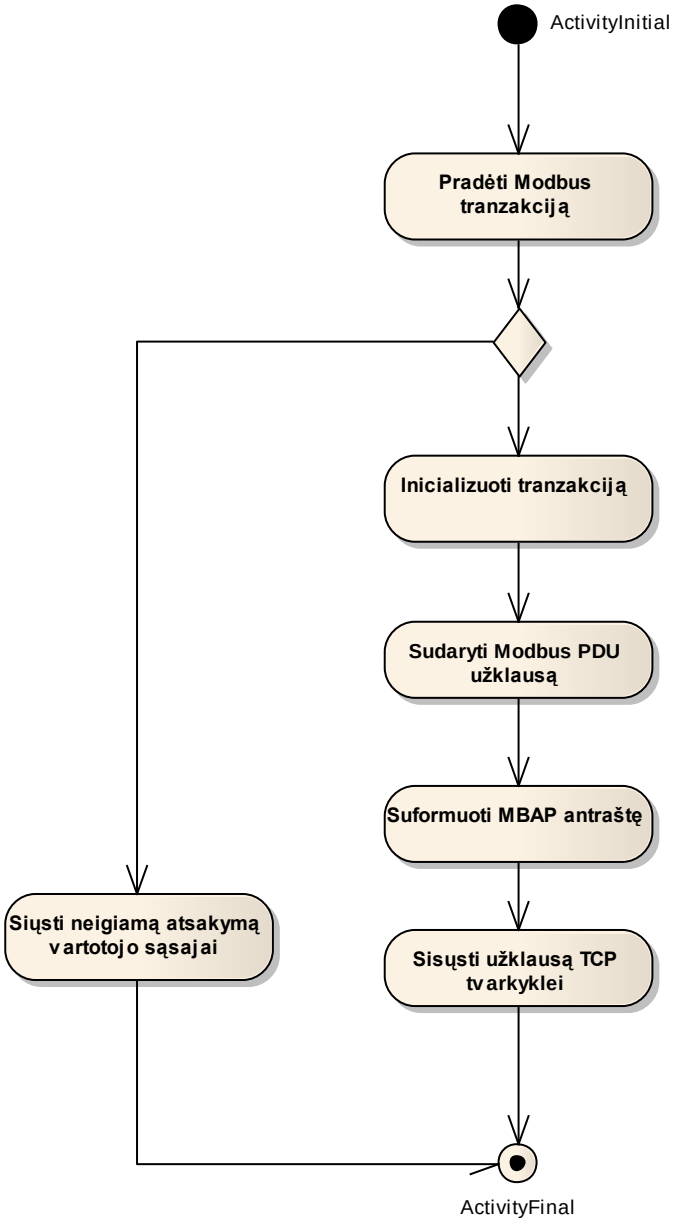


Pav. 29 Modbus tvarkyklės būsenų diagrama

Veiklos diagramos



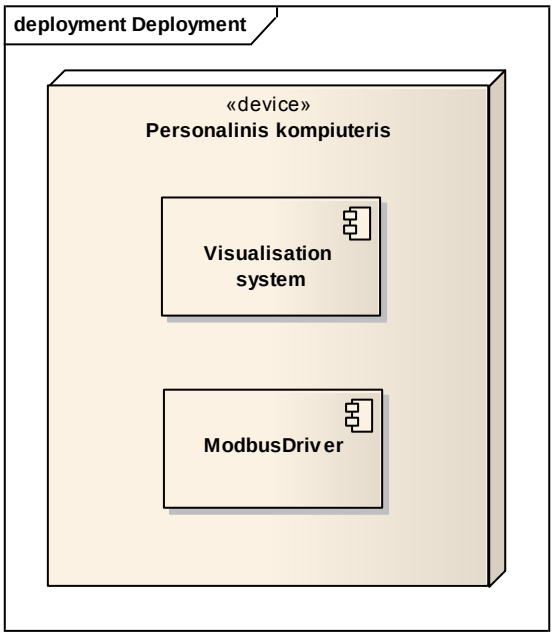
Pav. 30 Užklausos siuntimo per Modbus tvarkyklę veiklos diagrama



Pav. 31 Modbus užklausos formavimo veiklos diagrama

6.6. Išdėstymo (deployment) vaizdas

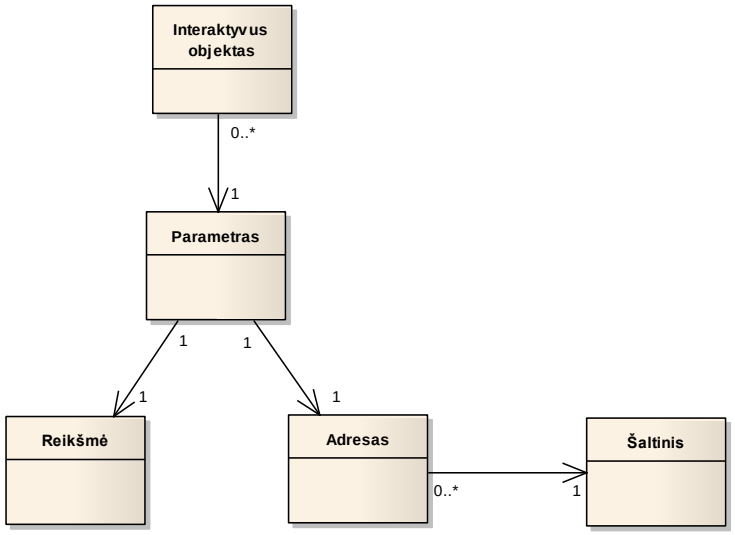
Kuriama sistema nėra paskirstyta ir visi jos komponentai bus diegiami viename kompiuteryje. Sistema galės būti diegiama į bet kurį personalinį kompiuterį, kuriame yra įdiegtas Microsoft .NET 3.5 karkasas.



Pav. 32 Sistemos išdėstymo diagrama

6.7. Duomenų vaizdas

Kuriama sistema nenaudos reliacinės duomenų bazės. Čia pateikiamas dalies vidinių duomenų objektinis modelis.



Pav. 33 Duomenų vieneto objektinis modelis

6.8. Kokybė

- Išplečiamumas – iš anksto numatant sistemoje išplečiamumo mechanizmus labai suspaprastėja jos vystymas.

- Pernešamumas – Sistemos pernešamumas tarp skirtingų platformų negalimas, todėl kad jos kūrimas paremtas vien tik Microsoft technologijomis.

7. DETALIOS ARCHITEKTŪROS SPECIFIKACIJA

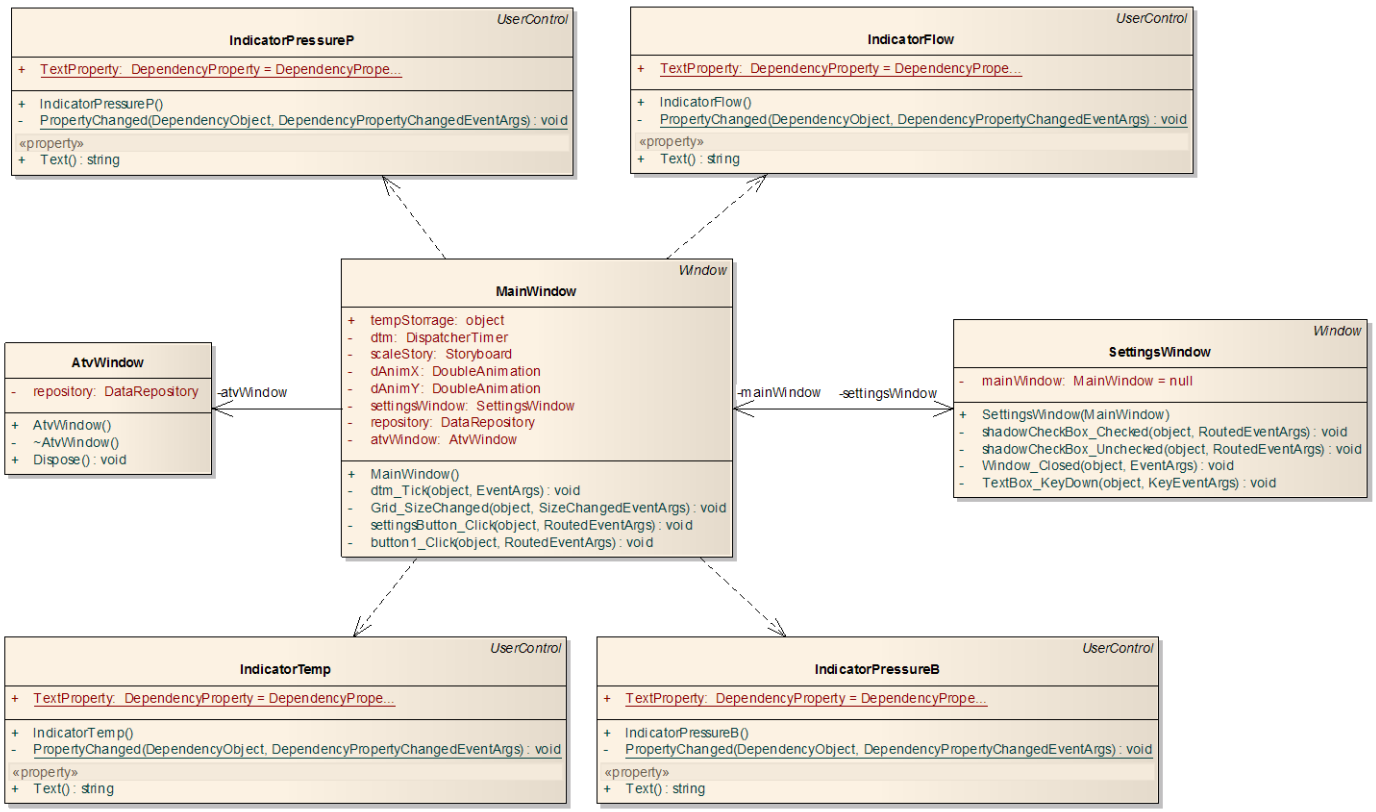
7.1. Sistemos komponentai

Komponentų sąraše pateikti bandomąjį sistemos prototipą sudarantys komponentai:

- System – vartotojo sąsajos branduolys
- Data – duomenų nuskaitymo ir valdymo mechanizmai
- ModbusDriver – pramoninio MODBUS protokolo tvarkyklės

7.2. Vartotojo sąsajos komponentas “System”

7.2.1. Detalizuota klasių diagrama



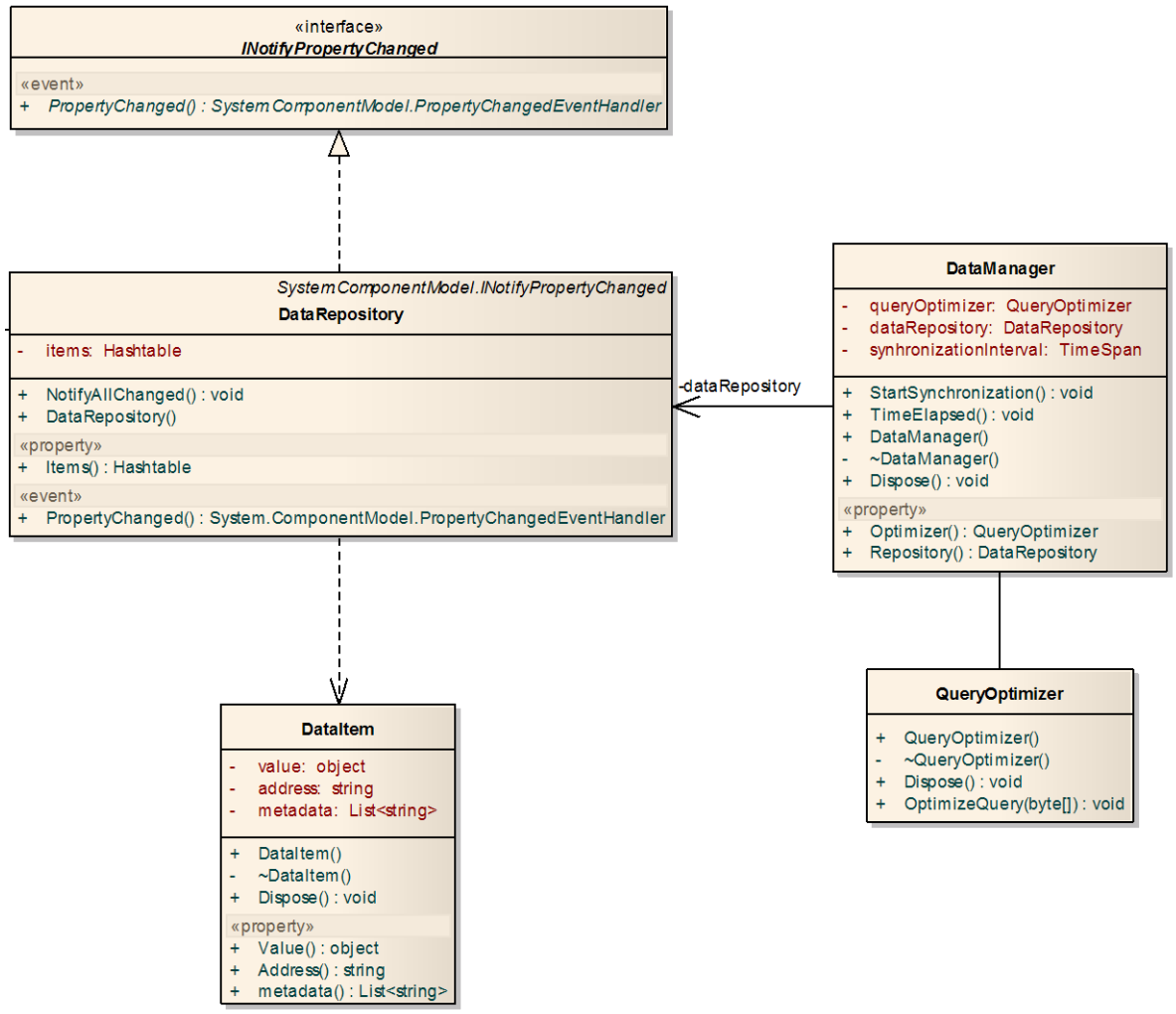
Pav. 34 Komponento "System" detalizuota klasių diagram

7.2.2. Atributai

Klasifikacija	Modulis (biblioteka wpfScada.dll)
Apibrėžimas	Šiame modulyje realizuota sistemos vartotojo sąsaja. Modulis taip pat atsakingas už kitų modulių iškvietimą ir valdymą.
Atsakomybės	Modulis atsakingas už vartotojo sąsajos elementų vaizdavimą, reagavimą į vartotojo veiksmus, duomenų atnaujinimą jiems keičiantis.
Apribojimai	Šis komponentas dirbti gali tik tose sistemose, kuriose yra įdiegtas Microsoft .Net Framework versijos 3.5.
Struktūra	Šio modulio struktūra pateikta "*System"* klasių diagramoje
Sąveikavimas	Komponentas sąveikauja su klasėmis, esančiomis modulyje "Data"
Resursai	Kompiuteryje turi būti įdiegta DirecyX 9 sistemos versija, vaizdo korta turėtų turėti ne mažiau nei 64Mb RAM ir turėtų palaikyti PixelShader2 technologiją.

7.3. Duomenų nuskaitymo/rašymo komponentas “Data”

7.3.1. Detalizuota klasių diagrama



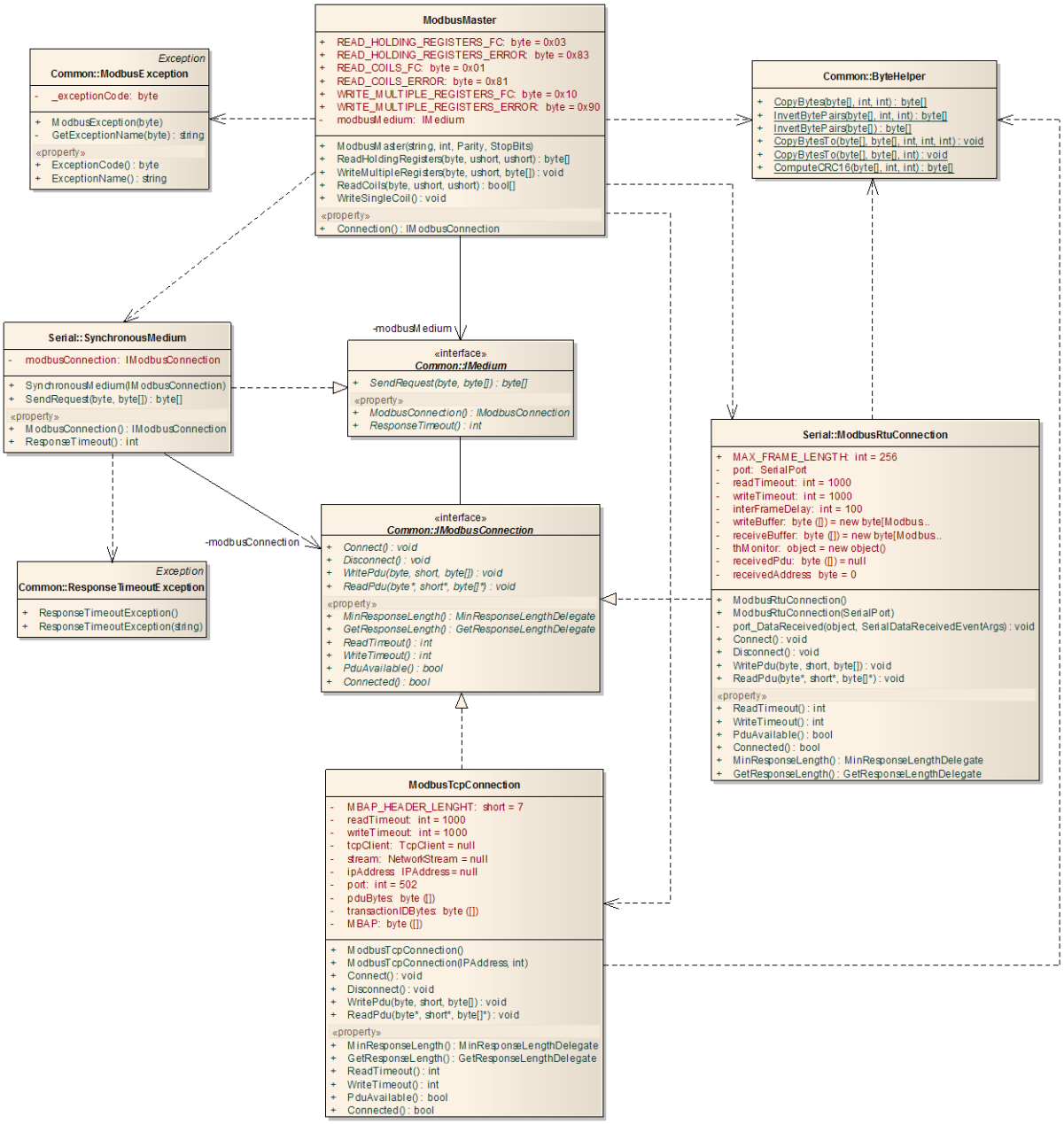
Pav. 35 Komponento "Data" detalizuota klasių diagrama

7.3.2. Atributai

Klasifikacija	Modulis (biblioteka wpfScada.dll)
Apibrėžimas	Modulis atsakingas už duomenų struktūros saugojimą ir mainus tarp įrenginių ir vizualizacijos vartotojo sąsajos.
Atsakomybės	Modulis atsakingas už šiuos veiksmus: ➢ Duomenų nuskaitymą iš įrenginių reguliariu intervalu ➢ Duomenų siuntimą įrenginiams kai juos pakeičia vartotojas ➢ Pranešimus vartojo sąsajos komponentams apie pasikeitusius duomenis ➢ Komunikacijos klaidų valdymą, pakartotinį prisijungimą prie įrenginių
Apribojimai	Nėra
Struktūra	Šio modulio struktūra pateikta “Data” klasių diagramoje
Sąveikavimas	Komponentas bendrauja su vartotojo sąsajos komponentais. Apie pasikeitusius nuskaitytus duomenis pranešama per INotifyPropertyChanged interfeisą.
Resursai	Ypatingi reikalavimai resursams nekeliami

7.4. Komunikacijas su įrenginiais per MODBUS protokolą valdantis komponentas “ModbusDriver”

7.4.1. Detalizuota klasių diagrama



Pav. 36 Komponento "ModbusDriver" klasių diagram

7.4.2. Atributai

Klasifikacija	Modulis (biblioteka ModbusDriver.dll)
Atsakomybės	Atsakingas už duomenų mainus tarp programos vidinės duomenų struktūros ir išorinių įrenginių per pramoninį MODBUS protokolą.
Apribojimai	Nėra
Struktūra	Šio modulio struktūra pateikta ModbusDriver klasių diagramoje
Sąveikavimas	Šį komponentą naudoja modulio "Data" klasė DataManager.
Resursai	Ypatingi reikalavimai resursams nekeliami

8. TESTAVIMO MEDŽIAGA

8.1. Testavimo tikslai ir objektai

Šio testavimo plano tikslas – apibrėžti programinės įrangos dalį, kuri turės būti testuojama ir nustatyti gaires bei rekomendacijas testavimui. Šiuo planu nesiekiama visiškai suvaržyti testuotojų. Jis nusako rekomendacijas, kas būtinai turėtų būti padaryta, kad testavimas būtų išsamus ir pilnas. Kuo tiksliau bus parinkti testai ir kuo jų bus daugiau, tuo daugiau klaidų bus pašalinta iš PĮ, ji bus stabilesnė.

8.2. Testavimo apimtis

Numatomas šių tipų sistemos testavimas:

- Vienetų testavimas. Tikrinamos atskiros modulio dalys. Šį testavimą gali atlikti programuotojas, naudodamas automatizuotus įrankius.
- Integravimo testavimas. Tikrinamas modulio darbas kliento aplinkoje. Ar gali modulis dirbti viename tinkle su kitomis programomis, ar jo darbas yra suderinamas su fizine įranga, su kuria jis turės bendrauti. Šis testavimas bus atliekamas rankiniu būdu.

- Stresinis testavimas. Kaip modulis elgsis esant ribiniam ir didesniam numatomam jo apkrovimui. Šiame testavime bus naudojama testinė kliento programa ir fizinė įranga.
- Našumo testavimas. Tikrinama, ar modulis užtikrina pakankamą našumą. Šiame testavime bus naudojama modifikuota testinė kliento programa (kuri bus naudojama ir stresiniame testavime).

Pagrindiniai apribojimai

Nėra galimybės ištestuoti tvarkyklės su visais įrenginiais. Bus naudojami tik du atrinkti fiziniai įrengimai. Darbas su likusiais bus bandomas palaipsniui.

Nuorodos

Lentelė 3. Nuorodos

Dokumentas	Data	Dokumento autorius
Reikalavimų specifikacija	2008 06 16	Algirdas Mockus
Architektūros specifikacija	2008 06 16	Algirdas Mockus
Detalios architektūros specifikacija	2008 06 16	Algirdas Mockus

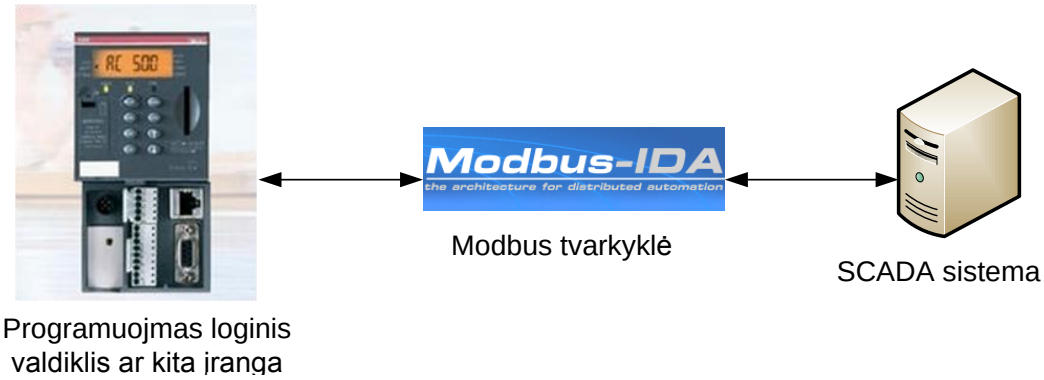
8.3. Testavimo planas

Šiame skyriuje aprašoma testuojama programinė įranga, detalizuojamos testavimo strategijos, apibrėžiami testavimo resursai.

Testuojama programinė įranga

Šiame dokumente aprašomas vizualizacijos (SCADA) sistemų kūrimo programinės įrangos vieno modulio testavimo planas.

SCADA (angl. Supervisory Control And Data Acquisition) vadinamos valdymo ir duomenų surinkimo sistemos, kurios nauojamos pramonėje automatizuotiems procesams vizualizuoti, parametrizuoti bei valdyti. Tokiose sistemose labai svarbų vaidmenį vaidina komunikacijos protokolų tvarkyklės, nes būtent per jas šios sistemos bendrauja su procesus valdančiais įrenginiais (pvz. programuojamais loginiais valdikliais, dažnio keitikliais ir t.t.). Nuo tvarkyklių našumo ir stabilumo proklauso ir visos sistemos darbas.



Pav. 37 MODBUS protokolo panaudojimo pavyzdys

MODBUS - tai nuo 1979 metų gyvuojantis ir bene labiausiai pasaulyje ir lietuvoje paplitęs pramoninis duomenų perdavimo protokolas. Tai iš dalies sąlygojo protokolo paprastumas ir aiškumas. Šiuo metu dažniausiai naudojamos dvi šio protokolo versijos:

- Modbus RTU – skirta perduoti duomenis per nuoseklias sąsajas. Dažniausiai naudojamas RS485 fizinis lygmuo.
- Modbus TCP – skirta perduoti duomenis per TCP/IP tinklus. Ši protokolo versija dirba 7 OSI lygmenyje (taikomajame).

Daugiau informacijos apie MODBUS protokolą galima rasti svetainėje http://www.modbus.org/

Sąsajos

Vartotojo sąsajos protokolo tvarkyklė neturi.

Testavimo strategija

Sekančiuose skyriuose bus apibūdinamos keturios testavimo strategijos, kurias reikės naudoti PĮ moduliui patikrinti.

Vienetų testavimas

Vienetų testavimas naudojamas smulkiausioms PĮ dalims patikrinti: metodams, klasėms, moduliams. Šį metodą galima naudoti viso PĮ išeities kodo kūrimo etapo metu

pradedant nuo pirmųjų metodų. Taigi tai padedai aptikti klaidas jau antyvosiose kodo kūrimo stadijose ir taip palengvina jų pašalinimą. Pagal klasikinį modelį (kurio laikysimės ir mes) šį testavimą atlieka patys programuotojai. Testavimą galima atlikti tiek rankiniu būdu tiek panaudojant įvairius automatizuotus įrankius.

Mūsų kuriama pramoninio duomenų perdavimo tvarkyklė bus testuojama metodų bei klasių lygmenyje „baltos dėžės“ principu. Kiekviena klasė bus išbandoma atskirai nuo kitų, vėliau atskiros klasės bus apjungiamos ir testuojami jų junginiai. Klasių metodams paduodami sugeneruoti duomenys turi būti parenkami taip, kad užtikrintų maksimalų kodo padengimą

Integravimo testavimas

Mūsų atveju integravimo testavimas bus naudojamas ne atskirų PĮ modulių tarpusavio integracijai patikrinti, bet išbandyti kaip įdiegta tvarkyklė dirba realiame duomenų perdavimo tinkle. Svarbu patikrinti sekančius dalykus:

- Ar tvarkyklė teisingai naudoja komunikacijos kanalus ir yra suderinama su kita juos naudojančia PĮ
- Kaip tvarkyklė reaguoja į tinklu keliaujančius duomenis
- Ar vienu metu dirbančios dvi tokios tvarkyklės nemaišo vien kitai
- Ar tvarkyklė yra suderinama su fizine įranga

Stresinis testavimas

Perkrovos pramoniniuose duomenų perdavimo tinkluose yra labai tikėtinos. Šio testo tikslas yra patikrinti, kaip tvarkyklė elgiasi esant didesniam už numatomą duomenų srautui per ją. Stebėsime ar tvarkykle nepakimba, ar pasibaigus perkrovai gali tęsti darbą, ar negrąžina neteisingų duomenų ir pan.

Našumo testavimas

Našumo testais tikrinsime kuriamos tvarkyklės veikimo greitaveiką normaliomis sąlygomis. Įvertinsime, ar greitaveika tenkina keliamus reikalavimus.

Testavimo resursai

Testavimui bus reikalinga sekanti fizinė ir programinė įranga:

- Kompiuteris (Intel 2-3Ghz, 512MB-2GB RAM)
- Fizinio lygmens keitiklis RS232->RS485
- Ethernet tinklas (pralaidumas 100Mbps
- Programuojamas loginis valdiklis (PLV), palaikantis Modbus RTU ir Modbus TCP protokolus
- NUnit testavimo aplinka
- MS Excel testavimo eigai, rezultatams bei defektams registruoti
- MS Visual Studio integruota PĮ kūrimo aplinka
- Testinė programa, kuri naudos testuojamą tvarkyklę duomenų mainams su išoriniu įrenginiu
- Testavimą atliks vienas programuotojas

Testavimo rezultatai

Testavimo rezultatai bus kaupiami specialiai šiam tikslui suformuotame MS Excel faile.

Testavimo įrankiai ir aplinka

Vienetų testavimui bus naudojamas NUnit paketas. NUnit yra specialiai .NET aplikacijų testavimui skirtas įrankis. Programuojant modulius, iškart rašoma ir testavimo programa, kurioje surašomi testavimo atvejai – t.y. kreipiniai į modulio funkcijas ir laukiami rezultatai. Vėliau testavimo programa panaudojant specialią aplinką (Framework) gali būti įvykdyta. Ši priemonė taip pat leidžia testavimo atvejams suteikti žymes (pvz. „Nebaigta“, „Atkreipti dėmesį“, „Ignoruoti“) ir pan., leidžiantį sekti bendrą testavimo progresą. Vieną kartą aprašius testavimo atvejus juos galima pakartotinai atlikti vėl ir vėl, kas iki minimumo sumažina pakartotinio testavimo kaštus.

Našumo testavimo metu programos stebėjimui realiame laike naudosime našumo analizės įrankį „MS CLR Profiler“. Šis įrankis atvaizduoja vidinę programos objektų krūvą (angl. object heap) bei šiukšlių surinkėjo (angl. garbage collerctor), kuris atsakingas už objektų sunaikinimą, darbą. Įrankis padės nustatyti programos dalis, kuriose neefektyviai naudojama atmintis o tuo pačiu ir procesoriaus resursai.

8.4. Testavimo procedūra

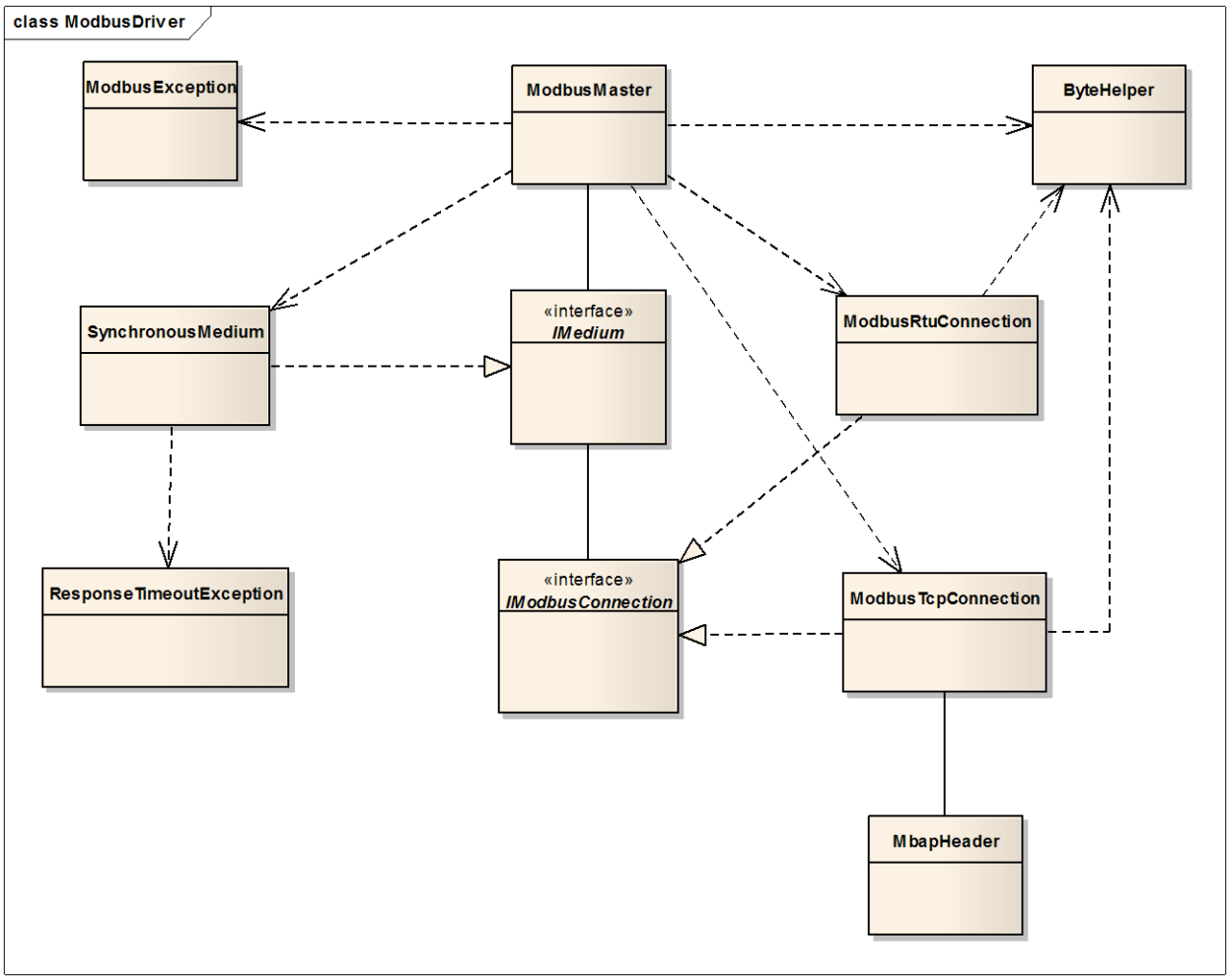
Šiame skyriuje toliau bus detalizuojamos naudojamos testavimo procedūros.

Testuojama programinė įranga

Bendrąjį testuojamos programos aprašymas galima rasti skyriuje 0.

Paveikslėlyje **Error! Reference source not found.**“ pateikiama principinė kuriamos tvarkyklės klasių diagrama. Pagrindinės klasės:

- **ModbusMaster** – fasadinė klasė, kuri tiesiogiai bus naudojama kituose komponentuose.
- **IModbusConnection** – interfeisas, kurį realizuoja klasės, atsakingos už duomenų perdavimą tinklu (ModbusRtuConnection ir ModbusTcpConnection). Šios dvi klasės yra vienos svarbiausių ir daro didžiausią įtaką PĮ našumui.
- **IMedium** – Šį interfeisą realizuojančios klasės yra tarpininkai tarp fasadinės klasės ir komunikacijos lygmens. Jose bus sudaromos paketų eilės.
- **ByteHelper** – Šioje klasėje realizuoti įvairūs pagalbiniai metodai darbui su baitų masyvais: kopijavimas, perkėlimas, CRC skaičiavimas.



Pav. 38 Principinė modulio struktūra

Testavimo procedūros

Šiame skyriuje detaliau apžvelgsime kiekvieną iš testavimo metodų, nusakysime kaip jis turi būti panaudotas, kas turi būti testuojama ir kokie rezultatai turėtų būti gaunami.

Vienetų testavimas

Vienetų testavimo tikslas – atskirai ištestuoti kiekvieną klasę ir visus jų metodus. Svarbiausių klasių vienetų testavimo gairės:

- **ByteHelper**. Šios klasės paskirtis – supaprastinti veiksmus su baitų masyvais. Šioje klasėje taip pat realizuotas kontrolinės sumos (CRC) skaičiavimo algoritmas. Šios klasės metodai turi būti testuojami paduodant įvairaus ilgio atsitiktinai sugeneruotus baitų masyvus. Svarbu patikrinti kraštutinius

atvejus: paduodamas masyvas yra NULL, masyvo ilgis lygus 0, masyvas turi vieną elementą, masyvo ilgis lygus 512000 (apie 512KB).

- **SynchronousMedium**. Ši klasė realizuoja interfeisą IMedium. Joje yra sudaromos FIFO tipo paketų eilės. Vėliau paketai vienas po kito nuosekliai yra perduodami ryšio lygmeniui tam, kad būtų išsiųsti į įrenginius. Naujas paketas nesiunčiamas tol, kol negaunamas atsakymas į prieš tai išsiųstą. Šios sąlygos išimtis – atsakymo laiko limito pasibaigimas (ResponseTimeout). Svarbu patikrinti, ar paketų eilės yra teisingai sudaromos, jei į šią klasę kreipiasi daugiau nei viena gija, ar teisingai skaičiuojamas atsakymo laiko limitas.

- **ModbusRtuConnection**. Ši klasė realizuoja interfeisą IModbusConnection ir yra atsakinga už nuoseklios duomenų perdavimo linijos valdymą (kompiuterio COM sąsaja). Svarbu patikrinti: ar visi interfeiso IModbusConnection metodai yra realizuoti; kaip metodai elgiasi, jei nepavyksta prisijungti prie išorinio įrenginio; ar yra užtikrinamas teisingas darbas jei šią klasę vienu metu bando naudoti kelios gijos; ar teisingai paskaičiuojamas paketo CRC suma.

- **ModbusTcpConnection**. Ši klasė realizuoja interfeisą IModbusConnection ir yra atsakinga už duomenų perdavimą naudojant TCP/IP paketus. Svarbu patikrinti: ar visi interfeiso IModbusConnection metodai yra realizuoti; kaip metodai elgiasi, jei nepavyksta prisijungti prie išorinio įrenginio; ar yra užtikrinamas teisingas darbas jei šią klasę vienu metu bando naudoti kelios gijos; ar teisingai nustatomas paketo ilgis.

- **ModbusMater**. Tai fasadinė klasė, kurioje realizuojamas užklausų formavimas ir vertimas į baitų masyvus, kurie yra perduodami vienai iš IModbusConnection interfeisą realizuojančių klasių. Testuojant šios klasės metodus būtina patikrinti, ar suformuojamų užklausų formatas atitinka Modbus standartą; ar užklausos teisingai formuojamos esant įvairiems metodų parametrų rinkiniams; ar kelios gijos gali naudoti ta patį šios klasės objektą vienu metu.

Integravimo testavimas

Tvarkyklę įdiegsime tikrą darbo aplinką imituojančioje testavimo aplinkoje ir tikrinsime, ar:

- Tvarkyklė netrikto kitų sistemų darbo. Ši tvarkyklė operacinėje sistemoje veiks lygiagrečiai kartu su kitomis tvarkyklėmis bei programomis. Būtina patikrinti, ar tvarkyklė netrikdys kitų pas klientą veikiančių sistemų darbo. Testuojama realią darbo aplinką atkartojančioje testavimo aplinkoje paleidžiant bandomąją programą, kuri naudos kuriamą tvarkyklę menamiems duomenims nuskaitinėti ir įrašinėti į išorinį įrenginį. Stebimas kitų tuo pačiu metu veikiančių programų darbas, operacinės sistemos įvykių žurnalas.

- Teisingai naudoja sistemos resursus. Tvarkyklė naudos operacinės sistemos nuoseklius prievadus (COM port) bei TCP/IP steką. Svarbu įsitikinti, kad tvarkyklei baigus darbą šie resursai būtų atlaisvinti ir jais galėtų naudotis kitos programos. Taip pat svarbu, kad centrinis procesorius nebūtų perkraunamas dėl „amžino ciklo“ buvimo tvarkyklėje.

- Neperkrauna tinklo. Turi būti patikrinta, ar tvarkyklė siunčia tik būtinus duomenis ir vieną kartą. Kaip ji elgiasi, jei nesulaukia atsakymo į užklausą. Ar nepradeda „bombarduoti“ tinklo pakartotinėmis užklausomis.

- Neiškraipo tinklu keliaujančių duomenų. Tvarkyklė turi atskirti, ar tinklu keliaujantys duomenys yra skirti jai ar ne. Ji neturi interpretuoti ir siųsti atsakymų į duomenis, kuriuos siunčia kitos sistemos.

- Vienu metu gali veikti kelios tokios tvarkyklės. Tikrinama, ar kelios to paties tipo tvarkyklės gali vienu metu apklausinėti ta pačią išorinę įrangą (jei tai nėra nuoseklaus duomenų perdavimo prievadas).

- Tvarkyklė suderinama su fizine įranga. Tikrinama, ar išorinis įrenginys atpažįsta tvarkyklės siunčiamas užklausas, ar tvarkyklė teisingai interpretuoja jo atsakymą. Testavimas vykdomas naudojant testinę programą, kuri naudoja testuojamą tvarkyklę menamų duomenų nuskaitymui ir įrašymui.

Stresinis testavimas

Vykdydami stresinį testavimą maksimaliai apkrausime tvarkyklę ir stebėsime, kaip ji elgiasi, kokiuos duomenis siunčia, ar spėja apdoroti visas užklausas.

Šiam testavimui naudosime testinę programą, kuri kas 10 ms naudodama tvarkyklę siųs 100KB dydžio užklausas įrenginiui ir gaus tokio paties dydžio atsakymus.

Našumo testavimas

Tikrinsime, ar tvarkyklė tenkina numatytus našumo reikalavimus. Našumu šiuo atveju vadinamas vidutinio ilgio užklausų ir atsakymų skaičius perduodamas tinklu per vieną sekundę. Vidutinis užklausos ilgis yra 100KB (Modbus standartas apibrėžia max. užklausos ilgį lygų 256KB). Perduodamos užklausos ir atsakymų formatai privalo atitikti standartą ir būti teisingi.

Tvarkyklė buvo projektuojama 5 užklausoms per vieną sekundę. Testavimui naudosime testinę programą, kuri naudodama testuojamą protokolo tvarkyklę apklausinės išorinį įrenginį minimaliu našumu 48h. Testinė programa taip pat tikrins, ar visos užklausos yra teisingos.

9. REZULTATŲ ĮVERTINIMAS

Vizualizacijos sistemos prototipas buvo sukurtas naudojant vien tik Microsoft Visual Studio programų kūrimo priemones. Naudojant standartinę vizualizacijų kūrimo sistemą analogišką vizualizaciją pavyktų sukurti apie 4 ar 5 kartus greičiau. Taip yra todėl, kad pastarosios sistemos yra specializuotos kurti būtent vizualizacijoms ir turi daugybę darbą paspartinančių įrankių, nereikalauja išeities kodo rašymo. Patobulinus Visual Studio sistemą ją būtų galima pritaikyti našiam vizualizacijų kūrimui. Patobulinimus galima įdiegti naudojant standartinius šios sistemos išplėtimo mechanizmus: įskiepius ir VSPackage mechanizmą.

Vizualizacijų kūrimo našumui padidinti reikalingi sekantys bendrieji Visual Studio sistemos patobulinimai:

- Specializuotas projekto šablonas

- Patogesnis ir daugiau funkcijų turintis XAML (vektorinės grafikos) redaktorius. Turi būti galimybė piešti įvairius grafinius primityvus, atlikti jų

transformacijas ir pan. Dabartinis vartotojo sąsajos redaktorius priraikytas kurti tik programų vartotojo sąsajas dėstant lange standartinius elementus.

- Animacijos kūrimo vartotojo sąsaja
- Išorinių įrenginių ir iš jų nuskaitomų duomenų konfigūravimo vartotojo sąsaja
- Įvairūs vedliai ir dialogo langai, kurie leistų susieti vartotojo sąsajos elementus su duomenimis, priskirti jiems įvairius standartinius veiksmus (pvz. kito lango atidarymas, programos uždarymas ir pan.)

Kita vertus su taip patobulinta Visual Studio sistema galėtų dirbti ir programavimo neišmanantys asmenys, nes daugumą funkcionalumo būtų galima pasiekti per vartotojo sąsają.

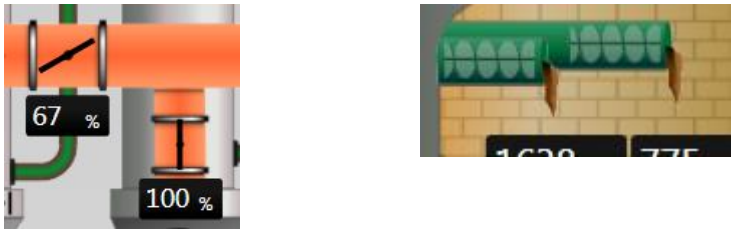
WPF karkaso savybių panaudojimas leido ženkliai sumažinti išeities kodo kiekį ir padaryti vartotojo sąsają patrauklesnę. Dažniausiai buvo naudojamos šios savybės:

- Objektų dydžio transformacijos. WPF vartotojo sąsaja pasižymi kompozicijos principu. Kiekvienas grafinis objektas ar jų grupė turi tėvinį objektą, kuriam priklauso. Jei tėvinio objekto dydis yra keičiamas, lengvai galima padaryti taip, kad automatiškai keistųsi ir jo vaikų dydžiai. Ši savybė labai naudinga kuomet reikia keisti sudėtingų objektų (pvz. vizualizacijos lango) dydį.

Pav. 39 Vizualizacijos prisitaikymo prie lango dydžio keitimo pavyzdys

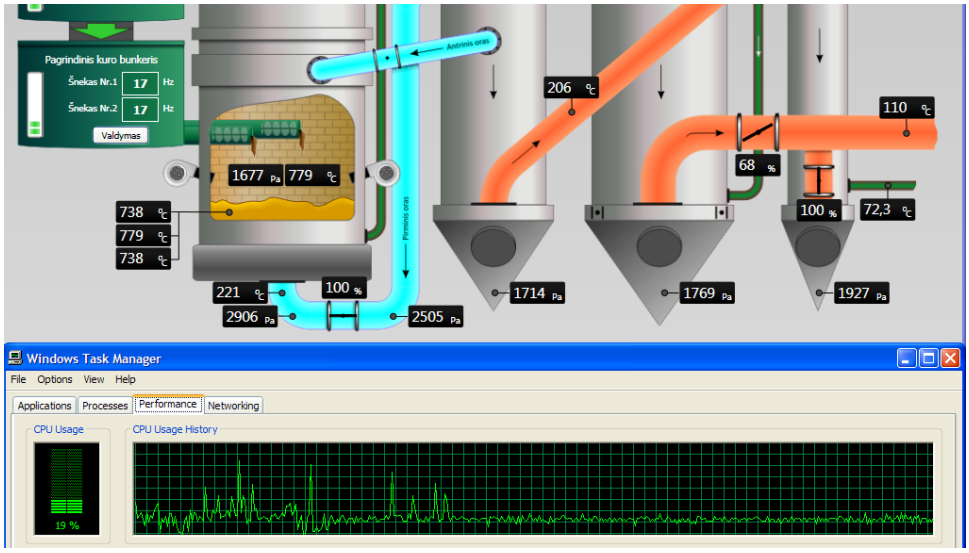
- Duomenų surišimas (angl. data binding). Matavimo prietaisų indikatoriai ekrane su duomenų nuskaitymo lygmeniu buvo susiejami naudojant standartinius WPF duomenų surišimo mechanizmus. Tai labai palengvino darbą, nes vidiniai duomenų surišimo mechanizmai patys pasirūpina įspėjimų apie pasikeitusius duomenis perdavimu tarp skirtingų programos lygių.

- Animacija. Užsklandų padėčių atvaizdavimui, sraigtinių transporterių darbo indikacijai buvo panaudotos WPF animacijos galimybės.



Pav. 40 WPF animacijos mechanizmai buvo panaudoti pavaizduoti užsklandų bei sraigtinių transporterių judėjimui

Daugiausiai klausimų ir abejonių sukėlė WPF technologijos greitaveika. Prototipo bandymų metu nuskaitant duomenis iš įrenginio kiekvieną sekundę ir įjungus animacijas procesoriaus apkrovimas nusistovėdavo apie 20%. Įvertinant tai, kad šis prototipas nėra pilnai funkcionuojanti vizualizacijos sistema, realiose sistemose greitaveika gali būti vienas iš šios technologijos panaudojimą stabdančių veiksnių. Reikėtų atlikti detalesnius tyrimus tam, kad nustatytai programos vietas, kurios labiausiai įtakoja darbo spartą. Tuo atveju būtų galima ieškoti programos kodo optimizavimo būdų. Kita vertus WPF yra dar ganėtinai jauna technologija ir Microsoft kompanija ją nuolat tobulina, optimizuoja jos greitaveiką, todėl tikėtina, kad sistema su naujesnėmis WPF versijomis veiks greičiau.



Pav. 41 Centrinio procesoriaus apkrovimas veikiant visoms animacijoms ir vykstant duomenų nuskaitymui iš nutolusių įrengimų

10.IŠVADOS

- Vizualizacijų sistemų kūrimui galima pakartotinai panaudoti taikomųjų programų kūrimo metodiką ir jau sukurtas priemones (pvz. Microsoft Visual Studio integruotą programų kūrimo aplinką). Rezultatas yra paprastesnis produkto palaikymas ir tobulinimas, didesnis lankstumas.
- Tam, kad vizualizacijų kūrimas su Microsoft Visual Studio sistema būtų efektyvesnis ir prieinamas IT išsilavinimo neturintiems asmenims, reikia sukurti papildomus įskiepius ir sistemos funkcionalumo praplėtimus. Tai turėtų leisti panaudoti bazinį ir dažniausiai pasikartojantį funkcionalumą naudojant tik grafinę vartotojo sąsają (dialogo langus, vedlius ir pan).
- Kadangi vizualizacijos kuriamos standartinėmis taikomųjų programų kūrimo priemonėmis, nebereikia naudoti „skriptavimo“ (angl. scripting) kalbų norint praplėsti sistemos funkcionalumą. Galima naudoti dideles galimybes turinčias programavimo kalbas (C#, C++, VB.NET), objektiškai orientuotą programavimo metodologiją bei jau .NET karkasui sukurtus komponentus.
- Modernioms vizualizacijos sistemoms grafinė vartotojo sąsaja turi būti vektorinė. Tai leidžia išsaugoti vaizdo kokybę esant skirtingoms vaizdo pateikimo priemonėms bei jų charakteristikoms ir taip pat palengvina vaizdinės informacijos manipuliacijas programos darbo metu.
- Tarp šiuo metu naudojamų vektorinės grafikos aprašymo standartų XAML kalba (viena iš WPF karkaso dalių) turi daugiausiai funkcionalumo, yra išbaigta

ir komerciškai palaikoma todėl labiausiai tinka vizualizacijoms kurti. Jos išskirtinis bruožas yra tai, kad ji yra tinkama aprašyti ne tik vektorinę grafiką, bet ir grafinę vartotojo sąsają bei dokumentus. Galima sudaryti ir visų šių tipų bendrą kompoziciją.

- Šios XAML kalbos savybės yra pačios naudingiausios kuriant vizualizacijas: galimybė maišyti vektorinės grafikos ir vartotojo sąsajos aprašus; vartotojo sąsajos elementų sudarymas kompozicijos principu; animacija; duomenų surišimas (angl. data binding); vartotojo sąsajos temų palaikymas.
- Pirminiai sukurto prototipo bandymai parodė, kad WPF sistema yra imli sistemos resursams. Tai ypač išryškėja naudojant animaciją. WPF kaip technologija yra pakankamai jauna, kompanija Microsoft ją tobulina ir didina jos greitaveiką. Kita vertus reikėtų atlikti papildomus bandymus ir ištirti galimus išeities kodo optimizacijos būdus.

LITERATŪROS SĄRAŠAS

[1] SCADA [žiūrėta 2007 11 10], prieiga internete http://en.wikipedia.org/wiki/SCADA

[2] Steven D. Garbrecht, The Benefits of Component Object-Based SCADA and Supervisory System Application Development [žiūrėta 2007 11 10], prieiga internete http://us.wonderware.com

[3] OPC standartai [žiūrėta 2007 08 20], prieiga internete http://www.opcfoundation.org

[4] Samuel C. Sciacca; Wayne R., Advanced SCADA Concepts. *IEEE kompiuterinė duomenų bazė.* 1995 [žiūrėta 2007 11 10]. Prieiga per internetą: <http://ieeexplore.ieee.org/Xplore/guesthome.jsp>

[5] Thomas Hadlich, Providing device integration with OPC UA. *IEEE kompiuterinė duomenų bazė.* 2006 [žiūrėta 2007 11 10]. Prieiga per internetą: <http://ieeexplore.ieee.org/Xplore/guesthome.jsp>

[6] Microsoft® Windows® Software Development Kit (SDK) for Windows Vista. Prieiga internete http://msdn.micfosoft.com

[7] FactoryLink SCADA solutions [žiūrėta 2007 11 10]. Prieiga internete http://www.inotek.com/Catalog/usdata1cn.html

[8] Monitor Pro 7. Getting Started Guide. Prieiga internete http://www.schneider-electric.com

[9] Vijeo Citect SCADA User Guide. Prieiga internete http://www.citect.com

[10] Jack Greenfield; Keith Short et al., Software Factories-Assembling Applications with Patterns, Models, Frameworks, and Tools. John Wiley & Sons ©, 2004, 666 puslapių

[11] Chris Sells; Ian Griffiths, Programming WPF, Second edition. O'Reilly Media, 2007, 821 puslapių

[12] R. Fan; L. Cheded; O. Toker, Designing a SCADA system powered by Java and XML. *IEEE kompiuterinė duomenų bazė.* 2005 [žiūrėta 2007 11 10]. Prieiga per internetą: <http://ieeexplore.ieee.org/Xplore/guesthome.jsp>

[13] Chen Qizhi; Qian Qingquan, The research of UNIX platform for SCADA. *IEEE kompiuterinė duomenų bazė.* 2000 [žiūrėta 2007 11 10]. Prieiga per internetą: <http://ieeexplore.ieee.org/Xplore/guesthome.jsp>

[14] Microsoft® Visual Studio 2005 Software Development Kit (SDK). Prieiga internete http://msdn.micfosoft.com

[15] Wu Sitao; Qian Qingquan, Using device driver software in SCADA systems. *IEEE kompiuterinė duomenų bazė.* 2000 [žiūrėta 2007 11 10]. Prieiga per internetą: <http://ieeexplore.ieee.org/Xplore/guesthome.jsp>

[16] Gamma Erich; Richard Helm; Ralph Johnson; John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995

[17] SVG-Based User Interface Framework [žiūrėta 2007 11 10]. Prieiga internete http://www.svgopen.org/2004/papers/SPARK/#helpers_sparkhelperdecorator

[18] SVG for SCADA Applications [žiūrėta 2007 11 10]. Prieiga internete http://www.svgopen.org/2004/papers/SVGforSCADA/

[19] Chapter 2. History: A Brief History of User Interfaces [žiūrėta 2009 05 20]. Prieiga internete http://www.catb.org/~esr/writings/taouu/html/ch02.html

[20] Jeremy Reimer, A History of the GUI. Ars Technica. 2005 [žiūrėta 2009 05 20]. Prieiga internete http://arstechnica.com/articles/paedia/gui.ars

[21] Kresimir Matkovic; Reinhard Sainitzer; M. Eduard Groller, Process Visualization with Levels of Detail. IEEE kompiuterinė duomenų bazė. 2002 [žiūrėta 2007 11 10]. Prieiga per internetą: http://ieeexplore.ieee.org/Xplore/guesthome.jsp

[22] Visualization (computer graphics) [Žiūrėta 2009 05 20]. Prieiga internete http://en.wikipedia.org/wiki/Visualization_(computer_graphics)