VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INŽINERINĖS GRAFIKOS KATEDRA

Giedrius Jonauskas

INTERNETINIŲ IR CAD TECHNOLOGIJŲ PANAUDOJIMAS ORGANIZUOJANT PARDAVIMO IR GAMYBOS PROCESUS BALDŲ VERSLE

INTERNET AND CAD TECHNOLOGIES USAGE TO ORGANIZE SALES AND PRODUCTION PROCESSES IN THE FURNITURE BUSINESS

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 62407T104

Inžinerinės ir kompiuterinės grafikos specializacija

Informatikos inžinerijos mokslo kryptis

Vilnius, 2009

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INŽINERINĖS GRAFIKOS KATEDRA

TVIRTINU
Katedros vedėjas

(parašas)
Daiva Makutėnienė
(vardas, pavardė)

(Data)

Giedrius Jonauskas

INTERNETINIŲ IR CAD TECHNOLOGIJŲ PANAUDOJIMAS ORGANIZUOJANT PARDAVIMO IR GAMYBOS PROCESUS BALDŲ VERSLE

INTERNET AND CAD TECHNOLOGIES USAGE TO ORGANIZE SALES AND PRODUCTION PROCESSES IN THE FURNITURE BUSINESS

Baigiamasis magistro darbas

Informacinių technologijų studijų programa, valstybinis kodas 62407T104

Inžinerinės ir kompiuterinės grafikos specializacija

Informatikos inžinerijos mokslo kryptis

Vadovas	doc. dr. Saulius Dereškevičius (Moksl. laipsnis, vardas, pavardė)	(Parašas)	(Data)
Konsultantas	doc. dr. Angelė Kaulakienė (Moksl. laipsnis, vardas, pavardė)	(Parašas)	(Data)

Vilnius, 2009

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
FUNDAMENTINIŲ MOKSLŲ FAKULTETAS
INŽINERINĖS GRAFIKOS KATEDRA

Technologijos mokslų (T000) sritis

Informatikos inžinerijos (07T) mokslo kryptis

Informatikos inžinerijos (07T1) studijų kryptis

Informacinių technologijų studijų programa, valstybinis kodas 62407T104

TVIRTINU
Katedros vedėjas

(parašas)
Daiva Makutėnienė
(vardas, pavardė)
(Data)

BAIGIAMOJO MAGISTRO DARBO
UŽDUOTIS
…….....Nr. .....
Vilnius

Studentui Giedriui Jonauskui

Baigiamojo darbo tema: Internetinių ir CAD technologijų panaudojimas organizuojant pardavimo ir gamybos procesus baldų versle

patvirtinta ..... m. ..... d. dekano potvarkiu Nr. .....

Baigiamojo darbo užbaigimo terminas ..... m. ..... d.

BAIGIAMOJO DARBO UŽDUOTIS:

Aiškinamasis raštas

- Išnagrinėti baldus gaminančios įmonės pardavimo-gamybos verslo procesą.
- Nustatyti šio proceso trūkumus.
- Išnagrinėti ir parinkti technologijas, reikalingas sukurti programinės įrangos sprendimą šiems trūkumams šalinti.
- Pateikti tokios programinės įrangos eksperimentinę sistemą.

Baigiamojo darbo rengimo konsultantai: doc. dr. Angelė Kaulakienė
(Moksl. laipsnis, vardas, pavardė)

Vadovas ..... doc. dr. Saulius Dereškevičius
(Parašas) (Moksl. laipsnis, vardas, pavardė)

Užduotį gavau
……..
(Parašas)
Giedrius Jonauskas
(Vardas, pavardė)
……..…....
(Data)

Vilniaus Gedimino technikos universitetas
Fundamentinių mokslų fakultetas
Inžinerinės grafikos katedra

ISBN ISSN
Egz. sk. ……..
Data ……...-….-….

Magistrantūros studijų **Informacinių technologijų** programos baigiamasis darbas

Pavadinimas	**Internetinių ir CAD technologijų panaudojimas organizuojant pardavimo ir gamybos procesus baldų versle**
Autorius	**Giedrius Jonauskas**
Vadovas	**dr. Saulius Dereškevičius**

Kalba: lietuvių

Anotacija

Baigiamajame magistro darbe ištirti baldų projektavimo ir elektroninės prekybos procesai, pasiūlytas šių procesų optimizavimo technologinis sprendimas. Iškelta duomenų pateikimo klientas-projektuotojui problema. Aptarta duomenų gavimo ir pritaikymo automatizuotam projektavimui problematika ir galimi sprendimo būdai.

Darbe išanalizuotos tinklalapių kūrimo ir kompiuterizuoto projektavimo technologijos, išnagrinėta duomenų mainų tarp *AutoCAD* ir duomenų bazių technologija, apžvelgtas sistemos kūrimo gyvavimo ciklas. Atlikta sistemos reikalavimų analizė, apžvelgti standartai ir programiniai sprendimai. Įgyvendintas baldų e.užsakymo pateikimo ir automatizuoto projektavimo sistemos technologinis sprendimas.

Sukurta baldų e.užsakymo ir automatizuoto projektavimo sistema – tai nauja baldų komplektavimo, konfigūravimo ir užsakymo internetinėje erdvėje galimybė. Šio projekto išskirtinumas ir optimizacijos analizių rezultatai gali būti naudingi įmonėms kuriant panašias taikomąsias sistemas. Išnagrinėjus teorinius ir praktinius internetinių ir auto*CAD* technologijų panaudojimo organizuojant pardavimo ir gamybos procesus baldų versle aspektus, pateiktos baigiamojo darbo išvados ir siūlymai.

Darbą sudaro septynios dalys: įvadas, internetinių technologijų analizė, CAD technologijų analizė, sistemos realizacijos teorinis modelis, sistemos kūrimo realizacija, išvados ir siūlymai, literatūros sąrašas.

Darbo apimtis – 79 p. teksto be priedų, 45 paveikslai., 3 lentelės, 45 bibliografiniai šaltiniai. Atskirai pridedami darbo priedai.

Prasminiai žodžiai

ASP.NET, AJAX, baldai, elektroninis užsakymas, VBA, AutoCAD, automatizuotas projektavimas.

Vilnius Gediminas Technical University
Faculty of Fundamental Sciences
Department of Engineering Graphics

ISBN ISSN
Copies No. ……..
Date ……...-….-….

Master Degree Studies **Information Technologies** study programme Master's Thesis

Title	**Internet and CAD technologies usage to organize sales and production processes in the furniture business**
Author	**Giedrius Jonauskas**
Academic supervisor	**dr. Saulius Dereškevičius**

Thesis language:
Lithuanian

Annotation

Master thesis explores 'customer to CAD designer' data delivery problem. The main goal is to analyze internet and CAD technologies which can be used to improve sales and production processes in the furniture business, disclose suitable technologies, and create two parts experimental system: online furniture configuration website and macro for automated furniture design.

By referencing system development life cycle, existing software quality standards and requirements for system realization were explored and used ASP.NET AJAX and AutoCAD VBA technologies.

The website was created to simplify the processes of customer's furnitures orders, enable online furnitures configuration and deliver order details. AutoCAD VBA macros allows to perform fast design and avoid human mistakes on it. Project oneness and optimization results could be useful for companies to build analogous sistems.

After investigation of theoretical and practical technologies aspects, was produced thesis conclusions and suggestions

Structure: introduction, analysis of internet technologies, analysis of CAD technologies, system development life cycle, realization of system development, conclusions and suggestions, references.

Thesis consist of: 79 p. text without appendixes, 45 pictures, 3 tables, 45 bibliographical entries.

Appendixes included.

Keywords

ASP.NET, AJAX, furniture e.order, VBA, AutoCAD, automated design.

TURINYS

PAVEIKSLŲ IR LENTELIŲ SĄRAŠAS

Paveikslai

Lentelės

SANTRUMPŲ SĄRAŠAS

Ajax	Asinchronizuota *JavaScript* ir *XML* (angl. *Asynchronous JavaScript and XML*)
API	Taikomųjų programų kūrimo sąsaja (angl. *Application Programming Interface*)
ASP	*Active Server Pages*
CAD	Kompiuterizuotas projektavimas (angl. Computer Aided Design)
CAE	Kompiuterinė projektavimo inžinerija (angl. *Computer Aided Engeneering*)
CAM	Kompiuterinė projektavimo gamyba (angl. *Computer Aided Manufacturing*)
CLR	Bendroji kalbos vykdyklė (angl. *Common Language Runtime*)
DB	Duomenų bazė (angl. Database)
CGI	Bendroji tinklų sietuvo sąsaja (angl. *Common Gateway Interface*)
DHTML	Dinaminė hiperteksto žymėjimo kalba (angl. *Dynamic HyperText Markup Language*)
EJB	*Enterprise Java Beans*
FTP	Failų perdavimo protokolas (angl. *File Transfer Protocol*)
HTML	Hiperteksto žymėjimo kalba (angl. *HyperText Markup Language*)
HTTP	Hiperteksto persiuntimo protokolas (angl. *Hyper Text Transfer Protocol*)
IS	Informacinė sitema
IT	Informacinės technologijos
Java EE	Java Enterprise Edition
Java ME	Java Micro Edition
Java SE	Java Standard Edition
JDBC	Java prieiga prie duomenų bazių (angl. *Java Database Connectivity*)
JSP	*JavaServer pages*
JVM	*Java* virtualioji mašina (angl. *Java Virtual Machine*)
LAN	Vietiniai tinklai (angl. *local-area networks*)
MIME	Daugiafunkcis internetinio pašto plėtinys (angl. *Multipurpose Internet Mail Extension*)
PHP	*Personal Home Page*
SQL	Struktūrinė užklausų kalba (angl. *Structured Query Language*)
URL	Universalusis adresas (angl. Uniform Resource Locator)
WAN	Tolimieji tinklai (angl. *wide-area networks*)
XHTML	Išplėsta teksto su nuorodomis žymų kalba (angl. *Extensible Hypertext Markup Language*)

ĮVADAS

Temos aktualumas

Vis labiau plėtojantis ir tobulėjant informacinėms technologijoms ir vis plačiau jas pritaikant praktikoje, nauji iššūkiai pasiekia ir šių technologijų naudotojus - vartotojus ir informacinių sistemų (IS) kūrėjus. Sparti technologijų plėtra neaplenkia ir inžinerinės grafikos. Dar prieš tris dešimtmečius modelių braižymas ir inžineriniai skaičiavimai buvo atliekami ant balto popieriaus lapo, o dabar tam tikslui sukurtos specialios *CAD* sistemos.

Santrumpa *CAD* (angl. *Computer Aided Design*) reiškia kompiuterinį projektavimą. Vyravo nuomonė, kad brėžinių atlikimas kompiuteriu ir yra kompiuterinis projektavimas, o daug kas iki šiol mano, kad *CAD* yra ne kas kita, kaip populiari grafinės aplinkos programa *AutoCAD*.

Iš tikrųjų sąvoka *CAD* yra žymiai platesnė. Šiuolaikinis kompiuterinis projektavimas yra ištisa veiksmų seka prasidedanti nagrinėjamo objekto (detalės, statinio, inžinerinės sistemos) virtualios formos arba modelio sukūrimu, aprašant visus realiam objektui būdingus fizikinius parametrus, apibūdinant jo padėties sąlygas.

Šiandien *CAD* santrumpa jau neatspindi šiuolaikinio kompiuterinio projektavimo, inžinerinių sistemų ir gamybos procesų valdymo galimybių visumos. Atsirado dar smulkesnis suskirstymas: *CAE* (angl. *Computer Aided Engeneering* – liet. kompiuterinė projektavimo inžinerija), *CAM* (angl. *Computer Aided Manufacturing* – liet. kompiuterinė projektavimo gamyba), *AEC* (angl. *Architecture Engeneering Construction* – liet. architektūrinis inžinerinis modeliavimas), *GIS* (angl. *Geograpbic Information Systems* – liet. geografinės informacinės sistemos) ir k.t. [35].

Baldų pramonėje taip pat yra sukurtas nemažas kiekis specializuotų projektavimo sistemų: *Kithechen Draw, AIS 10, PRO 100, Varstotas 100* ir kt. Jos atvėrė plačias galimybes baldų gamintojams ir vartotojams. Dabar jau nebereikia ant popieriaus lapo braižyti baldų modelių ar bandyti įsivaizduoti, kaip pasirinktas baldų komplektas atrodys realybėje. Užtekta nusipirkti ir įdiegti į kompiuterį vieną iš rinkoje siūlomų baldų projektavimo programų ir bandyti pačiam suprojektuoti. O paruoštą projektą pateikti baldų gamintojui. Atrodytų paprasta, tačiau šiam žingsniui vargu ar ryžtųsi menką kompiuterinį raštingumą turintys žmonės.

Kitas esminis trūkumas – didelės laiko sąnaudos. Norėdamas pats modeliuoti baldus, vartotojas turi studijuoti dokumentaciją tam, kad išmoktų dirbti su nauja įranga, arba gaišti laiką ir kreiptis tiesiogiai į gamintoją bei modeliuoti kartu su juo. Greitai besikeičianti verslo aplinka verčia verslą reaguoti į naujas galimybes ir konkurenciją. Kai klientas nori įsigyti baldų komplektą su nedaug baldų, besiskiriančių keliais parametrais, yra tikslinga baldų komplektavimo, konfigūravimo ir užsakymo pateikimo procesus perkelti į internetinę erdvę. Gautą aiškiai suformuotą elektroninį užsakymą gamintojas, pasitelkdamas automatizuoto projektavimo technologijas, keliais mygtukų paspaudimai gali atvaizduoti kompiuterio lange.

Šiame darbe nagrinėjama baldų komplekto konfigūravimo, užsakymo pateikimo bei formavimo procesų perkėlimo galimybės į internetinę erdvę. Apžvelgiamos automatizuoto projektavimo *AutoCAD* aplinkoje galimybės. Pasiūlomas užsakymo formavimo ir modelio projektavimo procesų optimizavimo sprendimas. Pateikiama šio sprendimo nauda ir perspektyvos.

Atlikus technologijų analizę sukuriama žiniatinklyje veikianti baldų konfigūravimo ir e.užsakymo pateikimo bei automatizuoto projektavimo sistema.

Sukurta sistema leidžia sumažinti laiko sąnaudas, vartotojui nebereikia vykti pas baldų gamintojus. Baldų komplektavimą, konfigūravimą ir užsakymo formavimą galima atlikti naudojantis internetu, net neišėjus iš namų. Kita sistemos dalis – *AutoCAD* aplinkoje veikianti automatizuota baldų projektavimo makroprograma keliais mygtukų paspaudimais projektuotoją įgalina nubraižyti kliento pateiktą baldų modelį.

Tikslas ir uždaviniai

Darbo tikslas – išanalizuoti rinkoje esančias žiniatinklio taikomųjų programų, skirtų informacijai apdoroti ir pateikti, kūrimo technologijas ir kompiuterizuotam baldų projektavimui skirtas *CAD* technologijas, atskleisti šių technologijų privalumus ir trūkumus, ištirti produkto projektavimo ir pardavimo procesus baldų versle, pasiūlyti ir įgyvendinti šių procesų optimizavimo technologinį sprendimą.

Šiam tikslui pasiekti iškelti uždaviniai:

- § išanalizuoti serverio (*JAVA .NET* ir *PHP*) ir kliento (*Ajax, JavaScript*) srities žiniatinklio taikomųjų programų kūrimui skirtas technologijas, atlikti šių technologijų lyginamąją analizę;
- § apžvelgti *AutoCAD* programavimo technologijas: *AutoLISP, VBA, ObjectARX*;
- § apžvelgti sistemos kūrimo gyvavimo ciklą;
- § atlikti baldų konfigūravimo, užsakymo pateikimo internetinėje erdvėje ir automatizuoto modelio projektavimo *AutoCAD* aplinkoje sistemos kūrimo galimybių analizę;
- § apibendrinti tyrimų rezultatus ir parengti rekomendacijas sistemos kūrimui;
- § realizuoti žiniatinklio taikomąją programą, skirtą baldų konfigūravimui ir užsakymo formavimui;
- § sukurti automatizuoto baldų projektavimo *AutoCAD VBA* makroprogramą;
- § susieti žiniatinklio taikomąją programą ir *VBA* makroprogramą su bendra duomenų baze;
- § atlikti žiniatinklio taikomosios programos ir *VBA* makroprogramos testavimą.

Mokslinis naujumas

Atsižvelgiant į tai, kad šiuo metu Lietuvos rinkoje praktiškai nėra specializuotų baldų komplektavimo, konfigūravimo ir užsakymo sistemų, orientuotų į interneto vartotojus, darbe atlikta technologijų ir poreikių analizė, apibendrintos realizacijos problemos ir parengti konstruktyvūs šių problemų sprendimo būdai.

Sistema orientuota tiek į interneto, tiek į paprastus vartotojus. Esant poreikiui, sukurta sistema lengvai tobulinama ar papildoma naujais funkciniais moduliais.

Praktinė vertė

Procesą nuo užsakymo formavimo iki suprojektuoto modelio sudaro 5 etapai: baldų komplektavimas, konfigūravimas, užsakymo formavimas, modelio redagavimas (jei reikia) ir modelio projektavimas. Pirmuosius tris etapus (baldų komplektavimą, konfigūravimą ir užsakymo formavimą) labai palengvina šiuolaikinės interaktyviųjų taikomųjų programų kūrimo technologijos, modelio projektavimą – *CAD*. Darbe analizuojamos informacijos apdorojimui ir pateikimui skirtos technologijos, veikiančios tiek iš kliento, tiek iš serverio pusės, ir *CAD* technologijos. Atliekama šių technologijų analizė: technologijos lyginamos, pateikiamos jų ypatybės, privalumai ir trūkumai.

Tyrimo objektas

Tyrimo objektas – baldų projektavimo ir pardavimo procesų optimizavimas; baldų konfigūravimą, e.užsakymo formavimą ir automatizuotą kompiuterinį projektavimą sujungiančios sistemos kūrimo galimybių studija ir šios sistemos realizacijos problemos.

Tyrimo sritis – baldų komplekto konfigūravimas internetinėje erdvėje, automatizuotas projektavimas *AutoCAD* aplinkoje.

Darbo aprobacija

Darbas buvo pristatytas 12 – ojoje VGTU Lietuvos jaunųjų mokslininkų konferencijoje „Mokslas – Lietuvos ateitis“, vykusioje Vilniuje 2009 m. balandžio mėn. 8 d. Pranešimas įtrauktas į konferencijos pranešimų rinkinį.

1. INTERNETINIŲ TECHNOLOGIJŲ ANALIZĖ

Įvadinės pastabos

Šiuolaikinėje pasaulinėje informacinių technologijų bei verslo sprendimų rinkoje IT kompanijos stengiasi kuo suprantamiau ir aiškiau pateikti savo kliento vizijas. Dabar nepakanka tik sukurti programą ar informacinę sistemą, bet reikia stengtis, kad ji būtų informatyvi, suprantama paprastam vartotojui ir interaktyvi [28].

Šiuo metu rinkoje gausu įvairių žiniatinklio taikomosioms programoms kurti ir kompiuteriniam projektavimui įgyvendinti skirtų technologijų, kurių kiekviena pasižymi tam tikrais ypatumais ir galimybėmis. Todėl norint sukurti patrauklią ir vartotojo poreikius tenkinančią baldų komplektavimo, konfigūravimo, e.užsakymo pateikimo ir projektavimo sistemą, svarbu išanalizuoti kiekvienos technologijos ypatybes, privalumus ir trūkumus. Šiame skyriuje bus apžvelgtos populiariausios žiniatinklio taikomųjų programų kūrimui skirtos serverio srities (*JSP, ASP.NET, PHP*) ir kliento srities (*Javascript, Ajax, Java Applet*) programavimo technologijos.

1.1. Žiniatinklio taikomųjų programų kūrimo technologijos

Žiniatinklio taikomoji programa arba tinklalapis – tai tam tikras informacijos šaltinis ar srautas, pateikiamas žiniatinklyje, kurį galima pasiekti pasinaudojus interneto naršykle. Paprastai ši informacija pateikiama puslapiuose teksto su nuorodomis žymų kalba (angl. *HyperText Markup Language – HTML*) arba išplėstine žymų kalba (angl. *Extensible Hypertext Markup Language – XHTML*). Jų paskirtis aprašyti dokumentų struktūrą, o sąsają tarp kelių puslapių sukuria hipertekstas. Žiniatinklio taikomosios programos gali būti saugomi vidiniame kompiuteryje arba serveryje [25].

Serveris – specialios paskirties kompiuteris ar programa, skirta kitų kompiuterių (klientų) aptarnavimui. Pagal paskirtį dažniausiai išskiriamos kelios serverių rūšys: žiniatinklio taikomųjų programų serveriai, failų serveriai (*FTP, SMB*), pašto serveriai (*POP3, SMTP, IMAP*), duomenų bazių serveriai (*MySQL, SQL*), taikomųjų programų serveriai ir pan. Daugeliu atvejų tas pats kompiuteris gali atlikti visas šias funkcijas priklausomai nuo to, kokios serverio programos bus įdiegtos. Serverių naudojimas pagrįstas kliento-serverio (angl. *client-server*) architektūra: kliento programa prisijungia prie serverio, skirto tam tikrai užduočiai, ir siunčia jam užklausas, kurios vykdomos. Dėl tokios architektūros galima lengvai organizuoti daugelio vartotojų bendrą darbą, efektyviau panaudoti skaičiavimo išteklius, padidinti sistemų patikimumą.

Žiniatinklio taikomosios programos gali būti pasiekiamos vietiniais tinklais (angl. *local-area networks – LAN*) privataus pobūdžio intranete (angl. *intranet*) arba internete – tolimaisiais tinklais (angl. *wide-area networks – WAN*). Į žiniatinklio taikomąsias programas kreipiamasi pasinaudojus hiperteksto persiuntimo protokolu (angl. *Hyper Text Transfer Protocol – HTTP*). *HTTP* – tai užklausimo-atsakymo

protokolas, jungiantis klientą ir serverį, leidžiantis naršyklėms siųsti ir gauti informaciją į serverius ir iš jų. Pagrindinės *HTTP* užklausos:

- *GET* – skirta dokumentų gavimui;
- *POST* – duomenų pateikimui (pavyzdžiui, *HTML* dokumentas);
- *HEAD* – informacijos apie dokumentą gavimui (atsakymo antraštės) [5].

URL standartas

Norint savo ekrane išvysti reikiamą tinklalapį, interneto naršyklėje reikia įvesti jo adresą *URL* standartu. Konkretaus informacijos šaltinio (tinklalapio, vieno tinklalapio puslapio) adresui nurodyti internete sukurtas *URL* standartas (angl. *Uniform Recource Locator*), išplečiantis kelio iki failo sąvoką. Adresas sudarytas iš penkių dalių:

1) informacijos perdavimo protokolo pavadinimo;
2) vartotojo identifikacijos dalies – slaptažodžio (nebūtinas);
3) kompiuterio (srities) adreso;
4) jungties numerio (nebūtina);
5) katalogų ir failų vardo (nebūtina).

Dažniausiai standartinis pagrindinis adresas yra sudaromas tik iš 1 ir 3 dalių – tai būtini adreso elementai, norint patekti į pagrindinį (centrinį) serverio puslapį. Slaptažodžių rašant adresą, dažniausiai rašyti nereikia, nes jie įvedami patogesniu būdu – pasitelkus vidines tinklalapio užklausas. Pagal šią numeraciją *URL* adresas schematiškai atrodo taip:

<protokolas://><vartotojas: slaptažodis@><mazgas><:jungtis></kelias>

Ženklai < ir > naudojami sudedamųjų dalių išskyrimui.

Realūs pavyzdžiai:

http://www.google.com

- čia naudojamos tik 1 ir 3 dalys

ftp://ftp.vgtu.lt/pub/software/win2000/pack.zip

- čia naudojamos 1, 3 ir 5 dalys [7].

Žiniatinklio taikomųjų programų skirstymas pagal informacijos pateikimo pobūdį

Pagal turinį ir informacijos pateikimo pobūdį tinklalapiai skirstomi į statinius ir dinaminius. Statiniai tinklalapiai susieti hipertekstu, visas turinys yra suformuotas pačiame tinklalapyje ir pateikiamas vartotojui nemodifikuotas. Visi vartotojai mato tokį patį tinklalapį. Tokio pobūdžio tinklalapių privalumai yra greitas kūrimas ir lengvas jų susiejimas, trūkumai – sunkus palaikymas ir atnaujinimas, kai tinklalapis apima daug puslapių. Statiniai tinklalapiai idealiai tinka mažai besikeičiančiai informacijai pateikti.

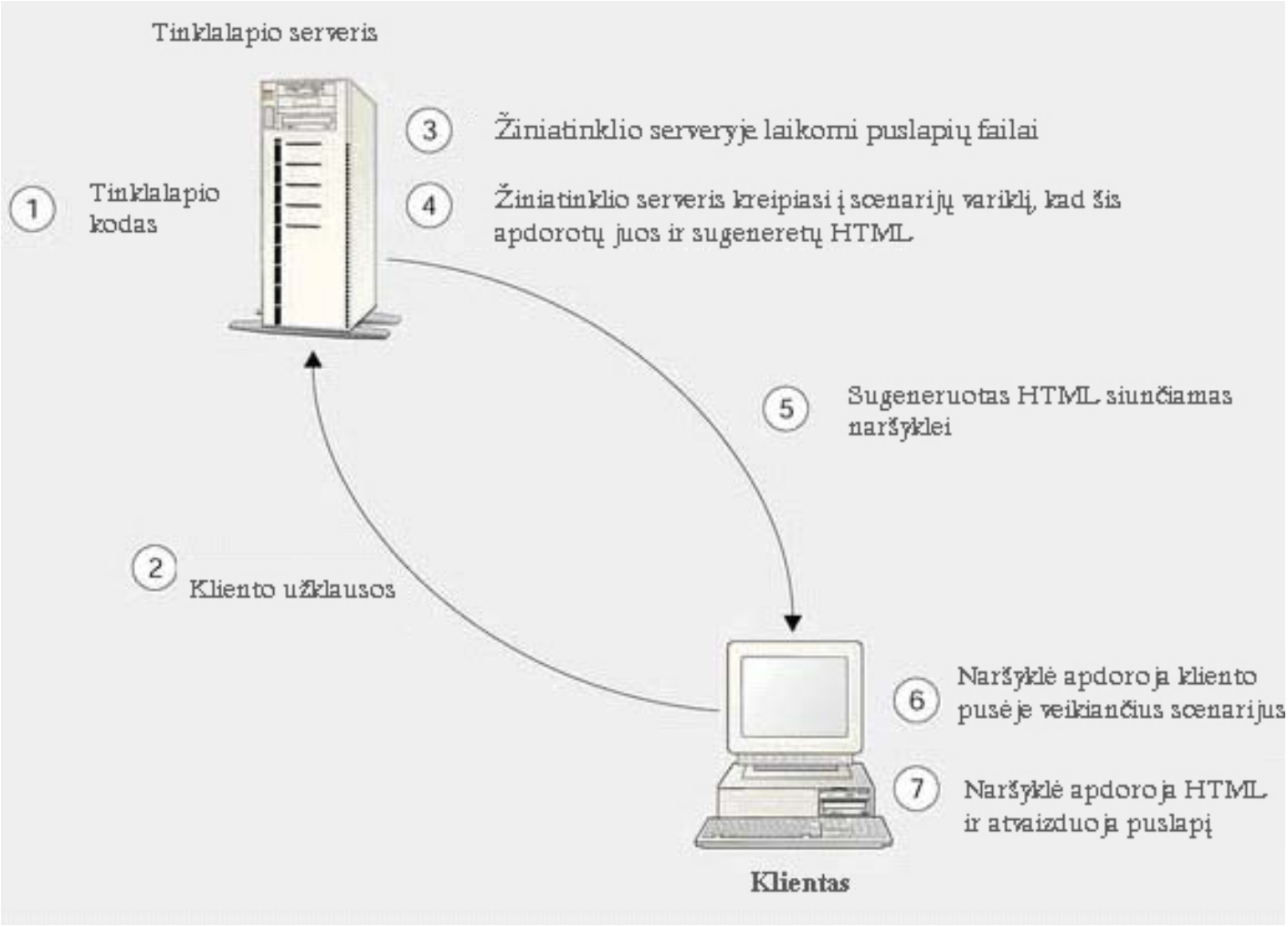
Dinaminiu turiniu pagrįsti tinklalapiai turi kur kas platesnių galimybių. Dažniausiai jų kūrimui naudojamas serverio srities (angl. *server side*), kliento srities (angl. *client-side*) programavimo technologijos, pakopiniai stiliai, *DHTML* kalba ir kitos priemonės. Svarbiausias dinaminių tinklalapių tikslas – kuo geriau pritaikyti jų turinį konkrečiam vartotojui, pavyzdžiui, pateikti paieškos rezultatus, sukurti elektroninės parduotuvės lankytojo krepšelį, įkelti tinklalapį, atitinkantį paskutinio seanso metu buvusias sąlygas [25].

1.2. Serverio srities programavimas

Serverio srities programos (angl. *servlets*) – tai programos, vykdomos žiniatinklio jungties serverio pusėje. Jos dinamiškai išplečia žiniatinklio taikomųjų programų funkcines galimybes. Serverio srities programos yra tarpininkai tarp kliento ir serverio, perimantys daugelį funkcijų tiek iš kliento, tiek iš serverio. Pavyzdžiui, bendraudamos su kliento srities programomis, serverio srities programos gali atlikti jų įgaliotojo serverio (angl. *proxy-server*) vaidmenį: jei kliento programa turi kreiptis į duomenų bazę kitoje mašinoje (kliento srities programoms tai uždrausta), šiuos veiksmus už kliento serverio programą gali atlikti serverio srities programa. Taip pat serverio srities programos gali papildyti serverio galimybes, palaikydamos serveriui nepažįstamą užklausų ir atsakymų pagrindu veikiantį protokolą (pavyzdžiui, *SMTP*, *POP3* ar *FTP*) [24].

1.2.1 *CGI* scenarijai

Bendrojo tinklų sietuvo sąsajos (angl. *Common Gateway Interface – CGI*) technologija skirta dinaminių puslapių kūrimui. *CGI* protokolas apibrėžia, kaip turi bendrauti žiniatinklio serveris ir jo vykdomos programos. Šiuo protokolu paremta kliento-serverio bendravimo schema pateikta 1 paveiksle.



1 pav. *CGI* protokolu paremta kliento-serverio bendravimo schema

1) Žiniatinklio taikomosios programos kodas įdedamas į serverį. 2) Naršyklė siunčia kliento užklausą į serverį. 3) Serveris analizuoja užklausą (apdoroja gautą informaciją). Šiuos veiksmus atlieka serverio dalis – *CGI,* galinti sąveikauti su kitomis programomis serverio kompiuteryje (duomenų bazėmis, elektroninėmis skaičiuoklėmis ir pan.). *CGI* programa gali būti realizuota bet kuria programavimo kalba, tačiau dažniausiai tam naudojama *Perl*. 4) Žiniatinklio serveris kreipiasi į scenarijų variklį, kad šis apdorotų scenarijus ir sugeneruotų *HTML*. 5) Sukuriamas arba kopijuojamas *HTML* dokumentas ir persiunčiamas klientui-naršyklei. Šio *HTTP* atsakymo pradžioje nurodomas atsakymo turinio tipas daugiafunkcinio internetinio pašto plėtinio (angl. *Multipurpose Internet Mail Extension – MIME*) formatu. 6) Naršyklė apdoroja kliento srityje veikiančius scenarijus. 7) Atvaizduojamas tinklalapio turinys.

***CGI* privalumai:**

§ Konkrečios taikymo logikos nereikia tiesiogiai integruoti į žiniatinklio serveryje esančio tinklalapio išeities kodą – dėl kiekvieno pakeitimo nereikia perkompiliuoti ir perkrauti žiniatinklio serverio.

- Taikomosios programos gali būti parašytos bet kuria kalba, įskaitant scenarijų rašymo kalbas. Daugelis *CGI* programų kuriamos būtent scenarijų kalbomis (*perl, php, sh*).
- Dauguma žiniatinklio serverių palaiko *CGI* protokolą, todėl *CGI* programas galima naudoti su skirtingais žiniatinklių serveriais.
- Taikomosios programos gali būti vykdomos kitomis teisėmis nei pats žiniatinklio serveris, taip pagerinamas visos sistemos saugumas.

***CGI* trūkumai:**

- Sparta. *CGI* programos priklauso nuo kompiuterio architektūros. Aptarnaujant kelis klientus jos nėra pakankamai efektyvios, nes kiekvienam sukuriamas atskiras procesas, o ne srautas, serverio resursai yra išnaudojami nepakankamai;
- Vykdant netinkamai aprašytus scenarijus labai apkraunamas serveris.

Žymiai efektyvesnės yra serverio srities programavimo kalbomis kuriamos programos. Jų pranašumai prieš *CGI*:

- Jos vykdomos serverio adresų erdvėje. Norint apdoroti atskirai kiekvieną kliento užklausą, galima sukurti atskirą vykdymo srautą.
- Kai kurios serverio srities programos yra nepriklausomos nuo kompiuterio tipo, *Sun, Netscape, Microsoft* tinklo taikomųjų programų serveriai palaiko *Servlet API*.
- Programos, sukurtos *Java*, gali būti laisvai perkeliamos iš vieno serverio į kitą.
- *Java* ar *.NET* serverio srities programos naudoja turtingas klasių bibliotekas.
- Tam, kad būtų apsaugoti serverio ištekliai, *Java* saugumo mechanizmas palaiko tam tikrus serverio srities programų apribojimus [5].

1.2.2 *Java* technologija

Java – viena iš populiariausių programavimo kalbų. Ja galima kurti tiek savarankiškas autonomiškas programas, tiek tinklu persiunčiamas ir interneto naršyklių vykdomas programas – kliento srities programas. Pagrindinis *Java* pranašumas, palyginus su kitomis programavimo kalbomis, yra jos nepriklausomumas nuo kompiuterių tipo. Nepriklausomumas pasiekiamas apdorojant *Java* programą dviem etapais: pirmiausia ją kompiliuojant į nuo kompiuterio tipo nepriklausomą vadinamąjį, baitinį kodą (angl. *byte code*) virtualiai *Java* mašinai, o paskui šį kodą interpretuojant. Tai lemia ir *Java* trūkumus – sumažėja šios programos našumas.

***Java* kalbos bruožai:**

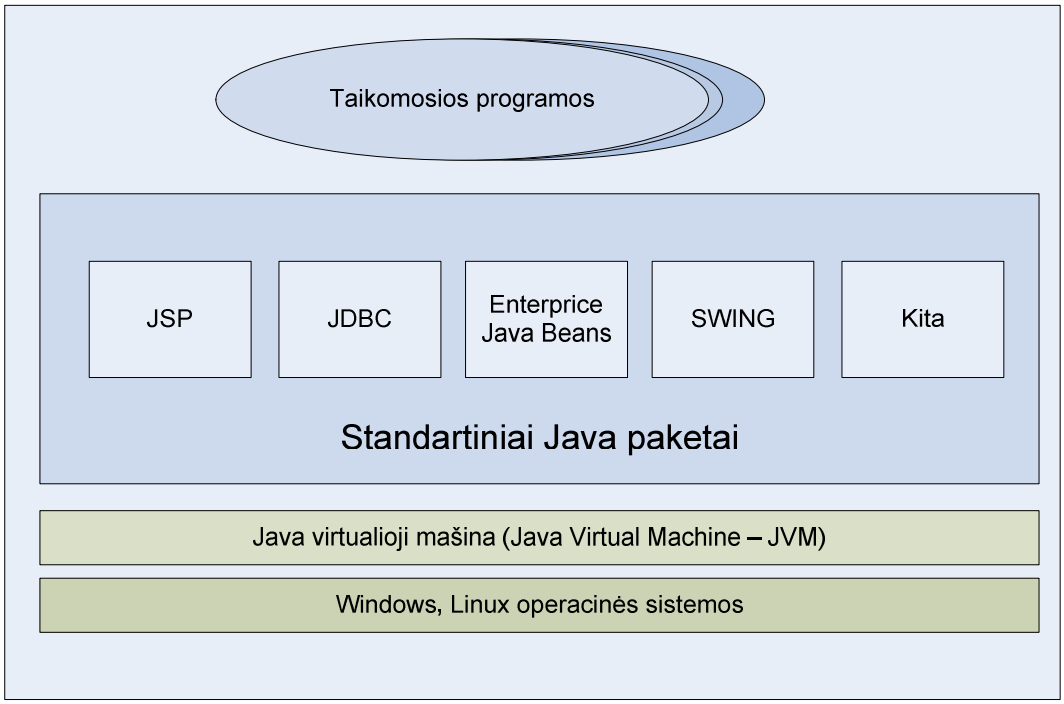
- *Java* – objektinė kalba. Visi *Java* programos duomenys yra objektai, išskyrus tik primityviuosius aritmetinius, simbolinius ir loginius duomenis. Tačiau ir tokiems duomenims yra klasės-kevalai, kad prireikus būtų galima jais operuoti kaip objektais.
- Griežtai tipizuota kalba. Kiekvienas objektas turi griežtai nustatytą tipą.
- Vienkamienis paveldimumas. Daugybinio paveldimumo, kurį palaiko *C++*, *Java* kalboje nėra. Tam tikrą panašumą su analogiška *C++* galimybe užtikrina tik sąsajų klasių (pačių abstrakčiausių klasių) mechanizmas. Visos *Java* klasės paveldi klasės *Object* savybes.
- *Java* pati skiria dinaminę atmintį objektams ir pati ją išlaisvina, kai objektas nebeturi nė vienos į save nukreiptos rodyklės (kitaip – nuorodos). Šiuos veiksmus atlieka mechanizmas šiukšlių rinktuvas (angl. *garbage collector*), tam tikrais momentais peržiūrėdamas programos užimamą atmintį. Pačios rodyklės programuotojui nėra prieinamos. Objektai dinamiškai saugomi atminties srityje. Toks atminties saugojimo mechanizmas yra viena iš *Javos* lėtumo priežasčių.
- *Java* turi patogų objektiškai orientuoto stiliaus mechanizmą, skirtą programos vykdymo klaidoms apdoroti. Visos tokios klaidos gali būti apdorotos pačios programos.
- *Java* – daugiasrautė (angl. *multi-threaded*) kalba, leidžianti rašyti programas, vienu metu vykdančias kelis srautus toje pačioje programos adreso erdvėje. Turi srautų sinchronizavimo ir tarpsrautinio bendravimo galimybės.
- Saugumas. *Java* kliento srities programos neturi prieities prie kliento kompiuterio atminties [6].

Java platforma vadinama susijusių programų grupė, sukurta *Sun Microsystems* ir leidžianti kurti bei vykdyti *Java* kalba parašytas programas. Ši platforma nėra pritaikyta tik vienam konkrečiam procesoriui ar operacinei sistemai. Greičiau tai yra vykdymo variklis, vadinamas virtualiąja mašina, ir kompiliatorius su standartinėmis bibliotekomis, kurios įgyvendinamos įvairioje aparatinėje įrangoje ir operacinėse sistemose, todėl *Java* jose visose gali veikti vienodai [16]. Yra kelios platformos rūšys:

- *Java Micro Edition (Java ME)* – apibrėžianti kelis skirtingus bibliotekų rinkinius įrenginiams, kurie yra pakankamai riboti, kad suprastų visus Java bibliotekų rinkinius.
- *Java Standard Edition (Java SE)* – bendriems tikslams naudojama kompiuteriuose, serveriuose ir kituose panašiuose įrenginiuose.
- *Java Enterprise Edition (Java EE)* – tai *Java SE* su papildomomis taikomųjų programų sąsajomis (angl. *application interface – API*), kurios naudojamos įmonės daugiapakopėms kliento serverių taikomosioms programoms.

J2EE platforma

J2EE (angl. *Java 2 Platform Enterprise Edition*) – tai platforma, skirta naujos kartos daugialyčių žiniatinklio taikomųjų programų kūrimui *Java* kalba. Platforma yra pagrįsta moduliniais komponentais, vykdomais programų serveryje. *J2EE* sistema yra kelių lygmenų. Vartotojui artimiausias lygmuo yra interneto naršyklė, veikianti vartotojo kompiuteryje.



2 pav. *Java* struktūra

Interneto komponentų lygmuo

Pagrindiniai *J2EE* modelio elementai yra internetas ir naršyklė. Būtent jie suteikia vartotojui sąsają dirbti su informacinėmis įmonių sistemomis. Šis metodas pranašus tuo, kad naršyklė yra paprasta programinė įranga, veikianti visose operacinėse sistemose, nereikalaujanti daug išteklių ir turinti kitų privalumus. Bendrauti su naršyklėmis *J2EE* apibrėžia serverio srities programas ir *JSP* specifikaciją. Būtent šie komponentai, naudojami kartu, leidžia atskirti veiklos logiką nuo vaizdinės dalies, taip supaprastinant interneto sistemos kūrimą ir palaikymą. Serverio srities programos – tai *Java* klasės, paveldinčios tam tikrą abstrakčią klasę ir realizuojančios metodus, sukuriančius interneto puslapį. *Java* klasės kodo pavyzdys:

```
public class Servletas extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
    throws ServletException, IOException
  {
    PrintWriter out = response.getWriter();
    out.println(„<html><head><title>Labas</title></head></html>“);
  }
  }
```

1.2.2.1 *JSP* technologija

JSP (angl. *JavaServer Pages*) – *Java* platformos technologija, leidžianti dinamiškai generuoti *HTML, XML, DHTML, XHTML* tipo puslapius. Ši technologija paremta serverio srities programų klasėmis ir yra *J2EE SDK* standartinė dalis. *JSP* technologiją palaiko dauguma tinklo serverių, palaikančių pačią *Java*. *JSP* puslapiai kompiliuojami į serverio srities programų klases [6, 12]. Ši technologija suteikia galimybę į statinį puslapį įterpti *Java* kalba parašytus kodo fragmentus aprašant juos specialiais sakiniais, pavyzdžiui:

```
<html>
   <head></head>
   <body>
       <jsp:include page="mycommon.jsp" >
           <jsp:param name="extraparam" value="myvalue" />
       </jsp:include>
       name:<%=request.getParameter("extraparam")%>
   </body>
</html>
```

JSP konteineriai atpažįsta šiuos sakinius ir naudoja juose įdėtą kodą serverio srities programai ar jo daliai generuoti, taip pat kodas gali kviesti *JavaBeans* komponentus. *JSP* yra paprastesnis puslapių kūrimo įrankis nei serverio srities *CGI* scenarijų programos: pakanka vienintelio dokumento dinaminiam puslapiui aprašyti.

JSP puslapio failai turi plėtinius *jsp* arba *jspx*. Plėtiniai nurodo tinklo serveriui, kad tam tikras failo dalis turi kompiliuoti *JSP* kompiliatorius. Kompiliatorius interpretuoja *JSP* sakinius ir generuoja atitinkamą kodą – serverio srities programos kodą. Tokie puslapių failai yra kompiliuojami pirmąkart suaktyvinus puslapį, vėliau sukompiliuotas tekstas saugomas serverio adreso erdvėje. Todėl kitos to paties puslapio užklausos atliekamos greitai, o pats puslapis faktiškai yra statinis *HTML* puslapis. Pakeitus *JSP* failo sakinius, failas automatiškai perkompiliuojamas ir perkraunamas į serverio atmintį.

JSP puslapyje keičiasi įprastine *HTML* kalba parašyti fragmentai ir *Java* sekcijos. *J2EE* serveris vykdo puslapyje esantį *Java* kodą, kuris iš anksto priskiria reikiamas reikšmes specialia *JSP* direktyva deklaruotiems kintamiesiems.

Serverio srities programų ir *JSP* puslapių didžiausias pranašumas prieš *CGI* yra tai, kad kiekvieną kartą gavus užklausą nereikia kurti naujo proceso. *JSP* serveriai kiekvienai užklausai apdoroti naudoja tik atskirą giją, kurios kūrimui ir inicializavimui reikia mažiau išteklių nei naujo proceso kūrimui.

Jei sistema taisyklingai programuojama, joje *JSP* puslapio kodas turi būti atsakingas tik už paties dokumento generavimą. Kiti veiksmai (sąveikavimas su duomenų bazėmis, sudėtingi skaičiavimo algoritmai, sprendimų priėmimas bei kiti panašūs algoritmai) turi būti koduojami aukštesniu *Enterprise Java Beans* (*EJB*) lygmeniu [12, 24].

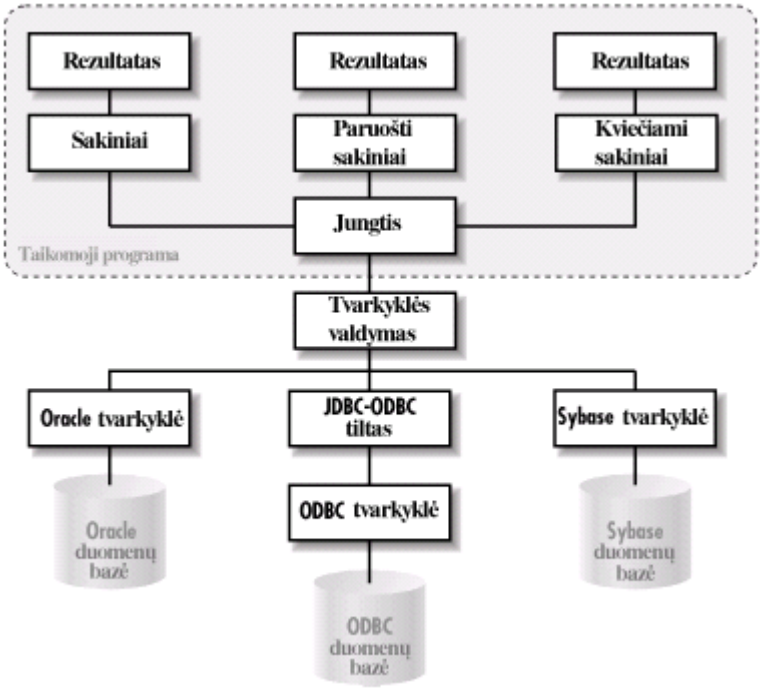
JSP serveriai

Komerciniai serveriai – *IBM Websphere*, *BEA Weblogic*, atvirojo kodo serveriai – *Jboss*, *Apache Tomcat*, *Jetty*. Populiariausias ir našiausias serveris yra *Apache Tomcat* (*Java* kalba įgyvendintas daugiaplatformis savarankiškas *HTTP* žiniatinklio serveris, palaikantis serverio srities programas ir *JSP* puslapius). *Apache Tomcat* serveris gali veikti įvairiose operacinėse sistemose: *Linux*, *Windows*, *Solaris* ir kt. *JSP* puslapius į serverio srities programas kompiliuoja *Jasper* kompiliatorius. Vėliausia *Apache Tomcat* versija – 6.0.18, palaiko serverio srities programų 2.5 ir *JSP* 2.1 specifikacijas [42].

Diegiant *Java* serverio srities programas, reikia paleisti serverį, serverio domeną, paskui įdiegti *J2EE* taikomąsias programas iš *jar* failų tam tikruose aplankuose. Serverio srities programos turi būti suarchyvuotos į *jar* failų tipą *war* (angl. *Web Application Archive* – liet. žiniatinklio programos archyvą). Suarchyvuotos žiniatinklio programos gali būti laikinos arba pastovios (t. y. veiks, kol veiks pats serveris) – tai priklausys nuo jų įdiegimo serveryje [18, 24].

1.2.2.2 *JDBC* prieiga prie duomenų bazių

Svarbus kuriamų sistemų aspektas yra galimybė susieti jas su duomenų bazėmis. *Java* aplinkoje tai įgalina *Java* prieigos prie duomenų bazės sąsaja (angl. *Java Database Connectivity* - *JDBC*). Ji skirta realizuoti *SQL* sakinius *Java* programose. Tai gali būti užklausų vykdymas, duomenų įterpimas, atnaujinimas ir t.t.



3 pav. *JDBC* prieiga prie duomenų bazių [21]

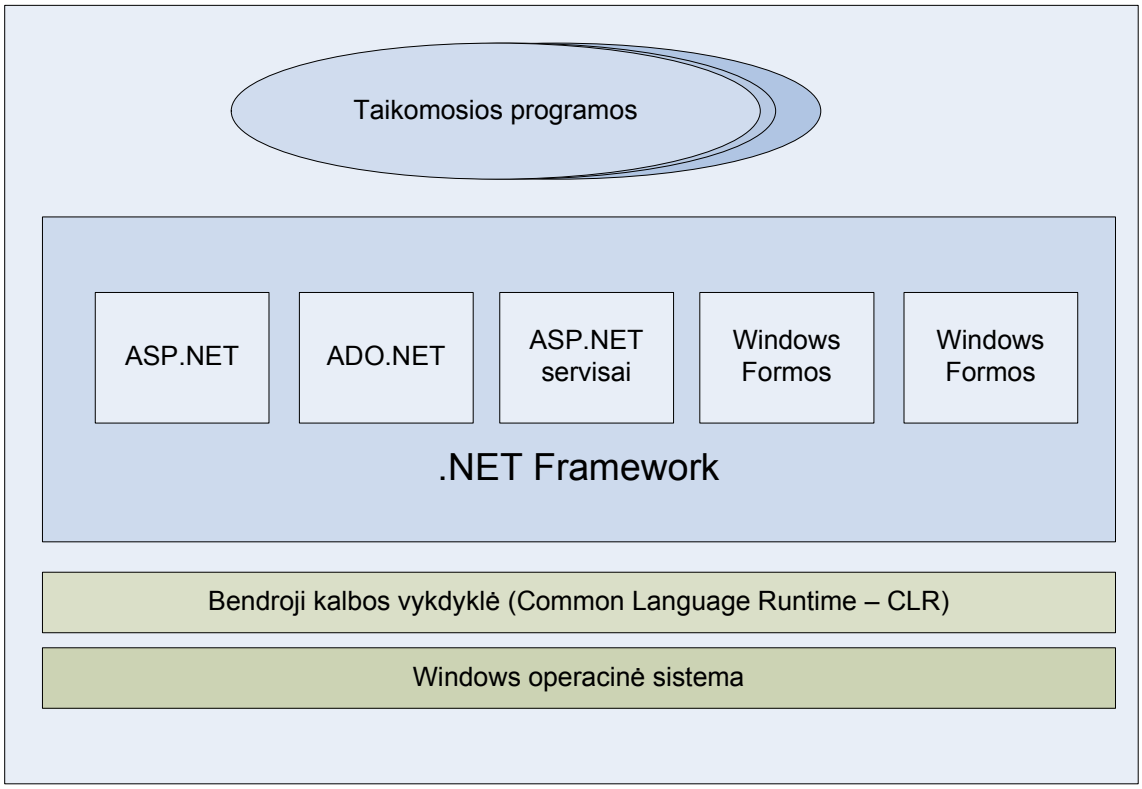
1.2.3 *.NET* technologija

Microsoft kompanijos 2002 m. pradėta plėtoti technologija, apimanti bendrąją kalbos vykdyklę (angl. *Common Language Runtime – CLR*) ir *.NET* platformą. *.NET* platforma sujungia į

vieną paketą visas technologijas reikalingas *Windows Desktop*, žiniatinklio paslaugoms ir taikomosioms programoms kurti. Ji turi daugybę jau paruoštų bibliotekų, kurias sudaro didelis objektinis standartinių klasių ir kitų tipų, parašytų *.NET* kalbomis, rinkinys. Nors iki *.NET* atsiradimo naudotą komponentų objekto modelio (angl. *Component Object Model – COM*) technologiją pamažu baigia pakeisti *.NET*, *COM* dar kartais naudojama dėl nebaigtos visiškos *.NET* integracijos į trečiųjų šalių taikomąsias programas ar esamų geresnių *COM* nei *.NET* savybių.

Taip pat *.NET* technologijos sąvoka apima integruotąją kūrimo aplinką (angl. *Integrated Development Environment – IDE*), skirtą greitam *.NET* programų kūrimui *Windows* aplinkoje. Struktūrinė *.NET* schema pateikta 4 paveiksle [30, 45].

.



4 pav. *.NET* struktūra

***ASP* apžvalga**

Prieš *ASP.NET* atsiradimą dinaminių žiniatinklio taikomųjų programų kūrimui buvo naudojama aktyvių serverio puslapių (angl. *Active Server Pages – ASP*) technologija, kurioje *ASP* puslapių funkcionalumo kūrimą įgalina aktyvusis scenarijų vykdymo variklis (angl. *Active Scripting Engine*) palaikomas *COM*. *ASP* technologija paremti puslapiai turėjo būti parašyti scenarijų kalba, dažniausiai *VBSrcipt*. Kitos scenarijų kalbos (pavyzdžiui, *JScript*, *PerlScript*) taip pat buvo galimos, tik jos turėjo atitikti *Microsoft* aktyvių scenarijų standartą ir turėjo būti aprašytos kalbų aprašymo srityje (*@Language*

direktyvoje) ar nurodytos `<script language="language" runat="server">` sakinyje. *VBScript* prisijungimo prie duomenų bazės pavyzdys:

```
<html>
<body>
<%
        Set oConn = Server.CreateObject("ADODB.Connection")
        oConn.Open "DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("DB.mdb")
        Set rsUsers = Server.CreateObject("ADODB.Recordset")
        rsUsers.Open "SELECT * FROM Users", oConn
%><script language="language" runat="server">
</body>
</html>
```

1.2.3.1 *ASP.NET*

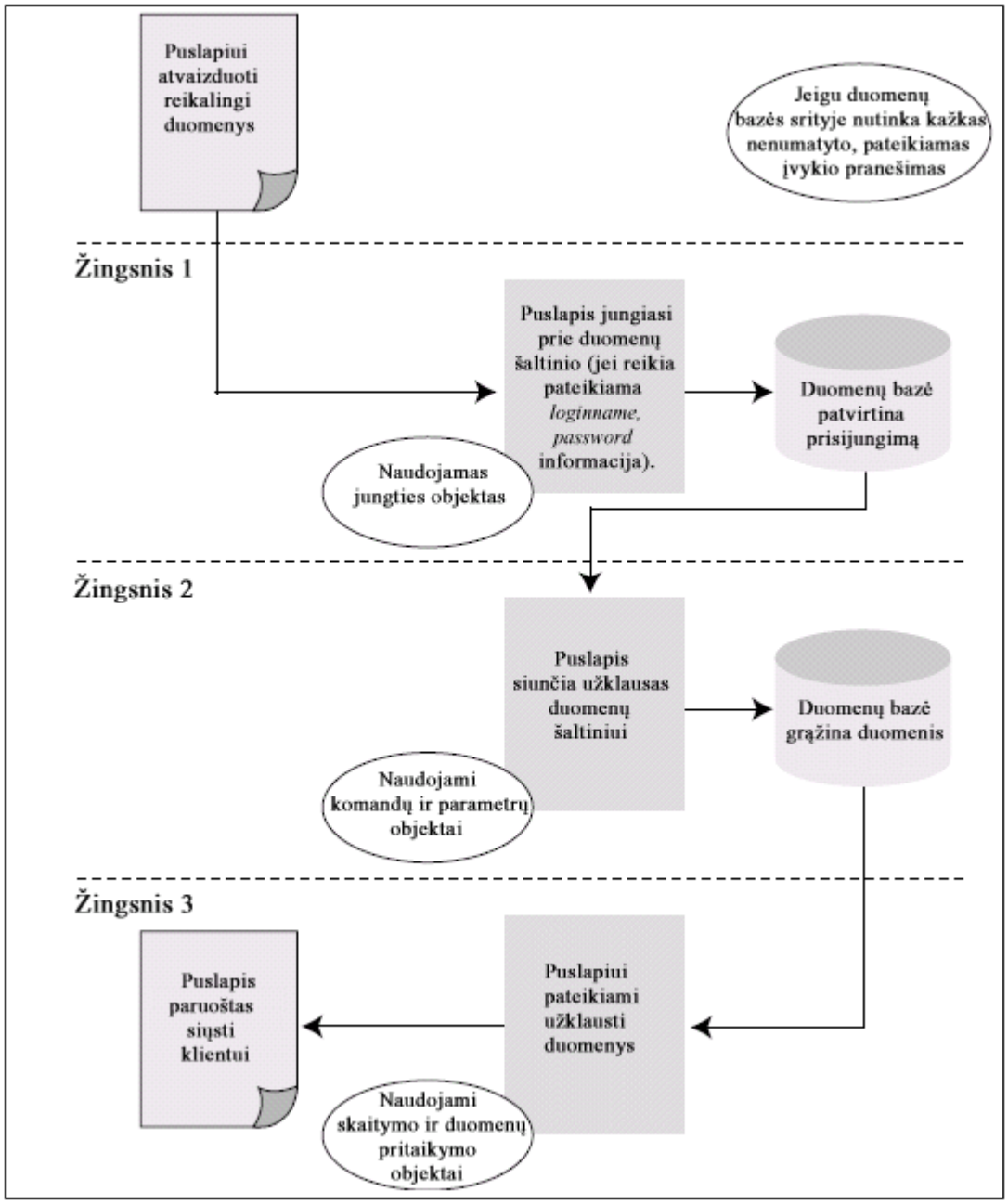
Tobulėjant technologijoms buvo sukurta *ASP.NET*. Ji yra svarbi *.NET* dalis, skirta dinaminių internetinių tinklalapių, žiniatinklio konstrukcijų ar žiniatinklio paslaugų (angl. *WEB services*) kūrimui. *ASP.NET* privalumai lyginant su *ASP*:

- § Didesnis programavimo kalbų pasirinkimo spektras.
- § Našumas: kiekvieną kartą užklausos metu serveris *ASP* puslapio kodą turi perskaityti, įkelti į atmintį, apdoroti ir įvykdyti (interpretuojamas kodas), o *ASP.NET* yra kompiliuojamas, užtenka vieną kartą sukompiliuoti kodą į binarinius failus, kurie vėliau paleidžiami labai greitai neatliekant anksčiau paminėtų procesų.
- § Aiški įvykio (*Load, Click, Change* ar kt.) kontrolė. Visi *ASP.NET* objektai (*EditField, Button* ir kt.) žiniatinklio puslapyje gali būti susieti su įvykiais, kuriuos apdoroja *ASP.NET* kodas, taip puslapio programavimas tampa paprastesnis ir labiau organizuotas.
- § Įdiegiant naują ar kompiliuojant esamą kodą nereikia serverio perkrovimo, *ASP.NET* tiesiog nukreipia visas naujas užklausas į naująjį kodą [13, 23].

ASP.NET puslapio failą sudaro *HTML* arba *XHTML* žymos, kuriose pateikiamas visas reikiamas statinis ir dinaminis žiniatinklio taikomosios programos turinys. Kodas, kuris veikia serveryje ir atlieka puslapio funkcijas, gali būti pateiktas ir pačiame puslapyje rašant jį tarp figūrinių skliaustų *<% -- dinaminis kodas -- %>* panašiai kaip tai daroma *JSP* ir *PHP*. Tačiau šiai praktikai nepritaria *Microsoft* korporacija. Ji rekomenduoja naudoti paslėpto kodo modelį, kai kodas yra įdedamas į atskirą failą ar į specifiškai sukurtą scenarijų žymą. Paslėpto kodo failai yra tipiškai pavadinami *puslapis.aspx.cs* ar *puslapis.aspx.vb*, priklausomai nuo to kuria kalba programuojama (*C# – .cs, VB – .vb*). Serverio srities kodo atskyrimas nuo *HTML* sluoksnio palengvina darbą komandoje, programuotojai gali keisti serverio srities kodą, neįtakodami *HTML* [37].

1.2.3.2 *ADO.NET* prieiga prie duomenų

ADO.NET – tai duomenų mainų tarp *.NET* programų ir duomenų šaltinio technologija. Duomenų šaltinis gali būti ne tik duomenys iš skirtingų duomenų bazių, bet ir tekstiniai, *Excel spreadsheet* ar *XML* failai. Sąveika tarp žiniatinklio taikomosios programos ir duomenų šaltinio pavaizduota 5 paveiksle.



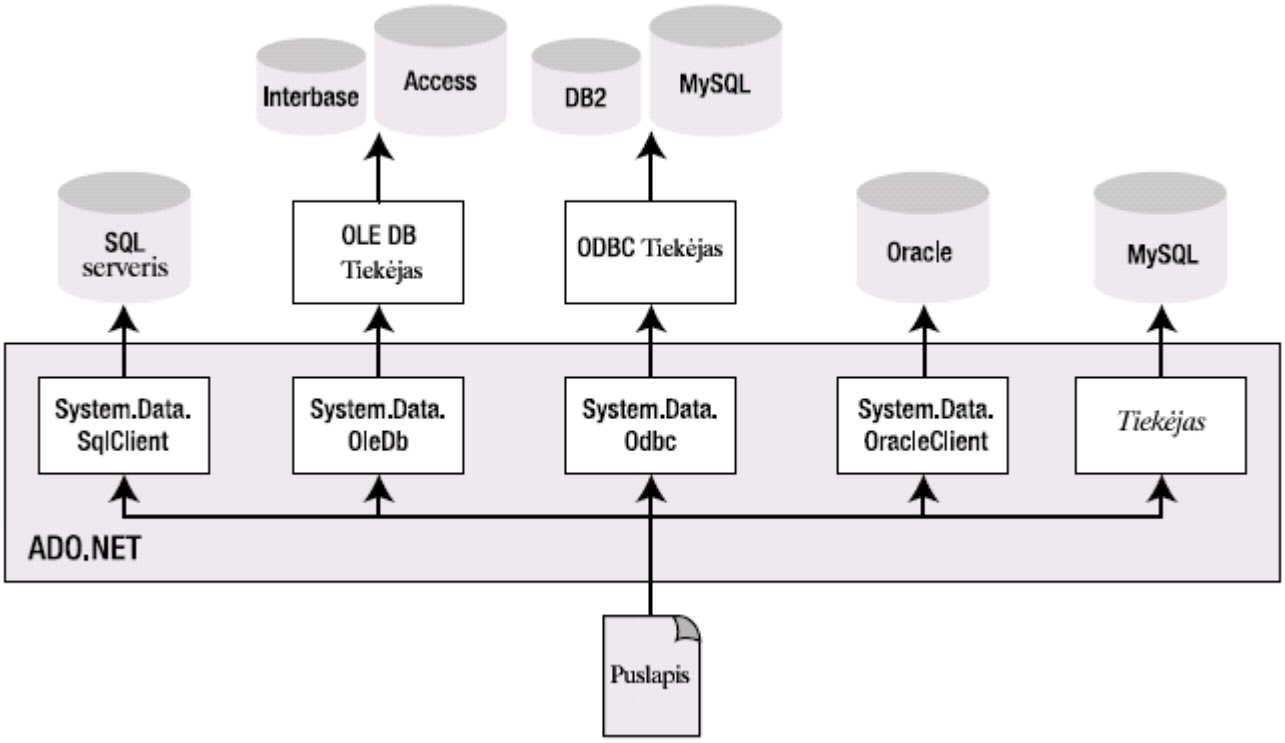
5 pav. Sąveika tarp žiniatinklio taikomosios programos ir duomenų bazės

Įvairiems duomenų šaltiniams pasiekti *ADO.NET* leidžia naudoti atitinkamus duomenų tiekėjus (žr. 6 paveikslą):

- *System.Data.Odbc* – *.NET* duomenų tiekėjas *ODBC* duomenų bazėms;
- *System.Data.OleDb* – *.NET* duomenų tiekėjas *OLE DB* duomenų bazėms (*Microsoft Access*);
- *System.Data.OracleClient* – *.NET* duomenų tiekėjas *Oracle* duomenų bazėms;

§ System.Data.SqlClient is the – *.NET* duomenų tiekėjas *SQL* serveriui;

§ System.Data.SqlServerCe – *.NET* duomenų tiekėjas *SQL ServerCE*.



6 pav. Duomenų tiekėjai [23]

System.Data – tai viena svarbiausių vardų sričių *.NET* platformos klasių bibliotekoje. Šios vardų srities tipai ir sudaro objektiškai orientuotą bibliotekų grupę – *ADO.NET*. Sąveika su duomenų šaltiniu yra paremta nuo tiekėjo nepriklausoma objektų grupe:

§ *SqlConnection* objektas skirtas prisijungimo prie duomenų šaltinio valdymui.

§ *SqlCommand* objektas – bendravimui su duomenų šaltiniu, į jį galima siųsti komandas.

§ *SqlDataReader* objektas – duomenų skaitymui.

§ *SqlDataAdapter* objektas – jei norima dirbti su duomenimis atsijungus nuo duomenų bazės.

1.2.3.3 Visual Web Developer 2008 IDE

Microsoft korporacijos sukurta *Visual Web Developer 2008 IDE* programavimo aplinka, leidžia naudoti *C#, VisualBasic*, C++ ir kt. programavimo kalbas. *IDE* pasižymi patogia grafine vartotojo sąsaja, grafiniu programos struktūros vaizdavimu. Čia įdiegtos automatinio teksto generavimo ir pabaigimo funkcijos, kurios leidžia sistemų kūrėjams greitai rašyti programinį kodą. *IDE* turi integruotą kompiliatorių ir galimybę derinti programas. 2008 m. išleidusi *Visual Web Developer 2008, Microsoft* korporacija planuoja 2010 m. birželio mėn. išleisti naują šios aplinkos versiją. Naujoje *IDE* didžiausias dėmesys toliau bus skirtas internetinių technologijų plėtojimui. Bus įdiegtas aukšto našumo *Javascript*

IntelliSense variklis; paruoštas visiškas *Silverlight*™ technologijos palaikymas, įgyvendintas sistemų kūrimo principas *„vieno mygtuko paspaudimas“*, įgalinantis patogiai ir greitai kurti žiniatinklio taikomąsias programas, taip pat turėtų supaprastėti taikomųjų programų konfigūravimo mechanizmas [23, 43].

1.2.4 *ASP.NET* ir *JSP* technologijų palyginimas

Išanalizavus *JSP* ir *ASP.NET* technologijas, pastebėta, kad jos turi daug panašumų. Abi suprojektuotos patogiam interaktyvių tinklalapių kūrimui, abi leidžia programuotojams skaidyti programavimo logiką nuo paruošto puslapio, kuris naudoja iškviečiamus komponentus, iki tokio, kuris pats išsikviečia komponentus. Tiek *JSP*, tiek *ASP.NET* programuotojams leidžia atskirti turinio generavimą iš sluoksnių, nuo pačio puslapio. *ASP.NET* palaiko *.NET* modelį, *JSP – JavaBeans*™ arba *JSP* žymas. Naudojant *JSP* yra galimybė laisvai praplėsti *JSP* žymų panaudojimą, pasitelkiant žymų bibliotekas (pavyzdžiui, naudojant *XML* žymas sumažėja priklausomumas nuo scenarijų rašymo).

Technologijos turi ir skirtumų, kurie gali būti esminiai rodikliai pasirenkant, vieną iš jų. Vienas iš esminių technologijų skirtumų yra tai, kad *JSP* suprojektuota taip, kad ji būtų nepriklausoma nuo naudojamos operacinės sistemos (OS), tuo tarpu *ASP.NET* realų suderinamumą turi tik su *Microsoft* šeimos operacinėmis sistemomis.

1 lentelė. *ASP.NET* ir *JSP* žiniatinklio serveriai ir operacinės sistemos

	ASP.NET	***JSP***
Žiniatinklio serveriai	*Internet Information Services (IIS), Personal Web Server*	*TomCat, JBoss, Blazix, WebLogic, WebSphere* ir kt.
OS	*Windows Server 2003*	*Solaris* ™, *Microsoft Windows, Mac OS, Linux* ir kitos *UNIX* šeimos.

Suprasdama šį trūkumą 2008 m. *Microsoft* korporacija sukūrė įrankį, leidžiantį *ASP.NET* realizuoti *Linux* terpėje (iki tol buvo naudojamas *Novell* kompanijos remiamas atvirojo kodo – *Mono http://www.mono-project.com*). Dėl *Linux* naudojamo kitokio skaičių konvertavimo į eilutę algoritmo (angl. *a high-performance algorithm*) *ASP.NET* taikomųjų programų vykdymas *Linux* platformoje žymiai spartesnis (*Linux* platformoje *.NET* skaičių konvertavimas į eilutę yra nuo 40 iki 260% greitesnis) [11, 19].

Svarbus *Java* ir *.NET* skirtumas yra tai, kad *Java* atveju galutinį *HTML*, kuris bus pateiktas klientui, nustato pateikčių pakopos programuotojas, o *.NET* atveju – *Visual Studio* valdiklis. Dėl tokios

Java savybės atsiranda trys problemos. Pirma, pateikčių pakopoje reikia daug kodo, nes kiekvienai „lengvų“ klientų[1] sistemai (angl. *thin client system*) reikia skirtingo kodo kelio. Antra, labai sunku išbandyti kodą kiekvienai įmanomai „lengvų“ klientų sistemai. Trečia, sunku įtraukti naujus „lengvus“ klientus į esamą taikomąją programą, nes norint tai atlikti, reikia išnagrinėti ir pakeisti didelį pateikčių pakopos logikos kiekį. *.NET* platformos atveju reikia parašyti nuo įrenginio nepriklausomą kodą, kuris sąveikautų su vaizdiniais valdikliais. Kokį *HTML* pateikti, priklausomai nuo kliento įrenginio pajėgumų sprendžia valdiklis, o ne programuotojas [21].

Lyginant pagal brandumą, *.NET* platforma yra brandesnė už *Java EE*. Ji turi daugiau bibliotekų, paruoštų šablonų, yra labiau išbaigta. Kadangi *.NET* trimis metais senesnė, spėjo užsitarnauti stambių kompanijų nuolankumą. Dėl savo stabilumo ir paprastumo *ASP.NET* yra lengviau įsisavinama, praleidžiama mažiau laiko programuojant. *.NET* technologija paremtos taikomosios programos veikia našiau nei *J2EE*. *.NET* – daug kalbų palaikanti technologija, tai programuotojams suteikia daugiau galimybių. Lyginant pagal *IDE* įvairovę, abi pusės jų turi panašų kiekį:

- Microsoft: *Visual Studio 2005, Visual Studio 2008 Pro, Visual WEB Developer 2008, Visual Studio 6.0*.
- Java: *Eclipse, NetBeans IDE, JCreator* ir t.t.

1.2.5 *PHP* technologija

PHP (angl. *Personal Home Page* – liet. asmeninis namų puslapis) yra plačiai paplitusi dinaminė interpretuojama programavimo kalba, pradėta plėtoti 1994 m. Rasmuso Lerdorfo. Pradžioje ši kalba buvo kaip *Perl* kalbos plėtinys, skirtas asmeninei svetainei kurti. Kalbos autorius sukurtą priemonių rinkinį pavadino *PHP/FI* arba „Asmeninis puslapis / formų interpretatorius“. Šis priemonių rinkinys leido lengvai valdyti pateiktas formas, vykdyti prisijungimo ir sekimo funkcijas. Pradinę *PHP* versiją sudarė interpretuojama scenarijų kalba, kurią galima įterpti į *HTML* puslapius, ir analizatorius, kuris, pateikus puslapiui užklausą, apdoroja įterptus scenarijus. Griežtai laikantis standartinio *HTTP* užklausos atsako ciklo, galima lengvai kurti scenarijus, juos derinti, išdėstyti ir palaikyti.

Dėl *PHP* modulio populiarumo, atvirojo kodo ir bendruomenės entuziazmo 1997 m. *PHP* buvo perrašyta suformuojant *PHP 3* pagrindą. Svarbiausias pakeitimas trečiojoje *PHP* versijoje – tai architektūra, leidžianti kūrėjams nesudėtingai tobulinti *PHP*. Šios kalbos sintaksė panaši į daugelį struktūrinių kalbų, ypač į *C* ir *Perl*.

---

[1]Lengvo kliento modelyje visų programų vykdymas ir duomenų valdymas yra vykdomas serveryje. Klientas yra atsakingas tik už atvaizdavimo programinės įrangos veikimą. Pagrindinis trūkumas yra tas, kad labai apkraunamas serveris ir tinklas. Sunkaus kliento modelyje serveris yra atsakingas tik už duomenų valdymą. Kliento programinė įranga įgyvendina taikymus ir sąveiką su sistemos vartotoju. Sudėtingesnis nei lengvo kliento modelis, ypač valdyme. Naujos programos versijos turi būti įdiegtos visuose klientuose.

PHP 3 taip pat naudoja scenarijus, įterptus į *HTML* dokumentą. Prieš pateikiant *Apache* serveriui failus su plėtiniu *.php,* jie apdorojami parengiamajame *PHP* procesoriuje, kur kiekvieno įterpto scenarijaus vietoje įterpiama atitinkama išvestis. Vartotojams, mokantiems *HTML* ir besinaudojantiems *PHP*, atsirado galimybė atlikti paprastas užduotis, pavyzdžiui, pakeisti kintamąjį ar atlikti sudėtingus veiksmus (prisijungti prie duomenų bazės). Dėl interpretuojamos *PHP* prigimties, scenarijų nereikėjo kompiliuoti. Panašiai kaip naršyklė kiekvieną kartą peržiūrint puslapį pateikia *HTML* kodą, *Apache PHP* modulis pateikia serverio scenarijus. *HTML* naudojantiems kūrėjams buvo pažįstamas keitimo/atnaujinimo ciklas, todėl *PHP* tapo populiari tarp programuotojų. Būdama savo populiarumo viršūnėje, *PHP 3* buvo įdiegta 10-yje procentų visų interneto serverių.

Toliau sekusios *PHP 4* versijos kūrimo tikslas buvo optimizuoti pagrindinį kodą taip, kad didelės ir sudėtingos taikomosios programos galėtų veikti našiau ir padidinti pagrindinės kalbos realizacijos ir interpretuojamos scenarijų kalbos, naudojamos interneto kūrėjų, skaidymo į modulius ir perkeliamumo galimybę. Svarbus *PHP 4* atnaujinimas – tai ribotos objektinio programavimo OOP galimybės. Nors kitų kalbų objektinio programavimo galimybės daug didesnės, tačiau *PHP 4* leido kūrėjams, mažai dirbusiems su kitomis kalbomis, suprasti objektinio programavimo svarbą ir galimybes.

2004-aisias išleistos *PHP* versijos pagrindinis bruožas – objektinis programavimas. *PHP 5* vadinama subrendusi programavimo kalba, kurią galima lyginti su *Java* ir *Microsoft VB, C#* kalbomis [10].

***PHP* privalumai:**

- *PHP* yra visiškai nemokamas;
- *PHP* nepriklausoma nuo platformos (veikia įvairiose operacinėse sistemose: *Win, *nix, MacOS, Solaris, HP-UX, AIX* ir k.t.);
- *PHP* – atvirojo kodo projektas, todėl jį kuria didelė grupė žmonių, iškilusios klaidos ar neatitikimai yra greitai ištaisomi;
- *PHP* sparčiai plečiasi, dėl savo nepriklausomumo nuo platformos, ji gali veiki ant daugelio internetinių serverių: *Apache, IIS, PWS, OmniHTTP, BadBlue* ir k.t.;
- pasižymi dideliu greičiu serverio pusėje, taip pat dirbant su duomenų bazėmis;
- nedideliuose projektuose *PHP* galima įterpti į savo *HTML* kodą;
- platus paruoštų šablonų spektras (*HotScripts, FreeScripts, PHPClasses.upperdesign.com* ir k.t.).

***PHP* trūkumai:**

- *PHP* yra interpretuojama kalba, mažesnis našumas nei kompiliuojamų kalbų;
- matomas *PHP* kodas;
- *PHP-GTK* programos šaltinis vis dar negali būti koduojamas [10].

PHP kalba yra atviro kodo, nesudėtinga ir gana lanksti. Ji veikia daugelyje operacinių sistemų, palaiko nemažai reliacinių duomenų bazių bei veikia su dauguma interneto serverių (*CGI, FastCGI, ISAPI* ir kt.) protokolų.

1.3. Kliento srities programavimas

Vienas iš kliento srities programavimo būdų yra plėtinių modulių naršyklėms (angl. *plug-ins*) rašymas. Plėtinys suteikia naršyklei papildomų funkcijų. Plėtiniai atvėrė kelią naujoms kliento srities scenarijų (angl. *script*) kalbomis atsirasti. Scenarijų kalbos dažniausiai naudojamos grafinėms vartotojo sąsajoms tobulinti. Kliento srities programa įterpiama į *HTML* puslapį, o ją apdorojantis modulis automatiškai suaktyvinamas peržiūrint puslapį.

Labiausiai paplitusi scenarijų, kalba palaikoma tiek *Internet Explorer*, tiek *FireFox* naršyklių, yra *JavaScript*. Kalba sukurta Brendano Eicho *Netscape* kompanijoje ir pavadinta *Mocha*, vėliau pervadinta į *JavaScript*. Vienas iš argumentų pervadinant kalbą buvo sintaksinis panašumas su *Java* kalba. Paskutinė *JavaScript* versija (1.8. *JavaScript*) yra skirta papildyti statinius *HTML* puslapius, išplečiant juos dinaminiu funkcionalumu (anketų parametrų tikrinimas, naujų langų atidarymas, suskleidžiamos hierarchinės struktūros rodymas, išsiskleidžiantis meniu ir daug kitų interaktyvumo formų). *JavaScript* kalba remiasi kelios pagrindinės tinklalapių kūrimo metodologijos – *DHTML* (dinaminis *HTML*), *AJAX*, *SPA*. Kadangi *JavaScript* yra interpretuojama kalba, scenarijai vykdomi nenaudojant kompiliavimo, jie įterpiami tiesiai į *HTML* puslapius [8, 38].

VBScript – tai *Visual Basic* kalbos atmaina, dažniausiai naudojama žiniatinklio taikomosiose programose. Tai pat kaip ir *JavaScript*, ji yra interpretuojamoji kalba. *VBscript* kodą *HTML* puslapiuose vykdo *Internet Explorer* naršyklė [15].

***JavaSript* ir *VBscript* palyginimas**

Panašumai:

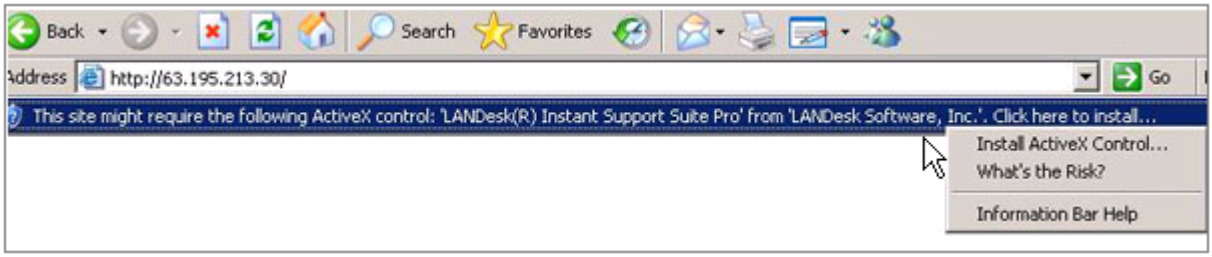
- Kalbos paprastos ir lengvai išmokstamos.
- Dažniausiai naudojamos žiniatinklio taikomųjų programų papildymui.
- Abi gali būti naudojamos vietoj *CGI* , nes jų scenarijai vykdomi kliento srityje.

Skirtumai:

- *JavaScript* yra nepriklausoma nuo platformos technologija, jos scenarijai gali būti vykdomi skirtingose naršyklėse, o *VBscript* veikia tik *Microsfot IE* naršyklėse.
- *JavaScript* turi keletą kalbos versijų. Dėl šios priežasties skirtinga *JavaScript* versija parašytą kodą įvairios vartotojų naršyklės ir jų versijos kitaip traktuoja, reikia daugiau testavimo.
- *JavaScript* yra jautri didžiosioms ir mažosioms raidėms (angl. *case-sensitive*).

Prie kliento srities programavimo technologijų priskiriamos *Java* kalba parašytos programos – apletai (angl. *applets*). Apletai yra nedidelės programos, vykdomos naršyklės viduje, įkeliamos į internetinį puslapį kaip jo dalis. Apletą suaktyvinus, vykdoma programa. Šių programų patraukli savybė yra nepriklausomumas nuo kompiuterio platformos. Programos kuriamos ir apdorojamos dviem etapais. Pirmiausia apleto kodas kompiliuojamas į baitinį kodą, taip jis tampa nepriklausomas nuo kompiuterio architektūros (platformos). Kai norima peržiūrėti, jį interpretuoja virtuali *Java* mašina, kurią palaiko dauguma naršyklių. Jei kliento srities programą sudaro keli moduliai (klasės), tai jo siuntimo tinklu pagreitinimui visi moduliai suglaudinami į vieną *jar* failą. Saugumo sumetimais kliento programos turi griežtus apribojimus darbui su kliento kompiuterio ištekliais: pavyzdžiui, tokia programa neturi teisių skaityti kliento kompiuterio kietojo disko ir į jį įrašyti.

Kita kliento srities programavimo technologija yra *Microsoft ActiveX*, leidžianti prijungti ir vykdyti programinius modulius žiniatinklyje. Šią technologiją palaiko *Internet Explorer* ir *FireFox* naršyklės. *ActiveX* yra savarankiški programiniai komponentai, kuriami remiantis *COM* technologija. Komponentai gali būti vykdomi tik kitų taikomųjų programų konteinerių viduje. Tokių konteinerių pavyzdžiai yra *MS Visual Basic, MS Access, MS Internet Explorer*. Iš komponentų gali būti sudaromos kitos taikomosios programos. Pagrindinė problema, naudojant *ActiveX* komponentus *HTML* dokumentuose, yra kliento kompiuterio saugumo užtikrinimas. Jei *Java* kliento srities programos dėl savo prigimties negali padaryti žalos klientui, tai *ActiveX* komponentai turi prieigą prie kliento kompiuterio atminties. Saugumą stengiamasi užtikrinti naudojant skaitmeninius komponentų parašus. Diegiant naują komponentą į kompiuterį, naršyklė dialogo lange užklausia vartotojo, ar komponentą reikia įkelti ar ne (7 paveikslas). Lange nurodoma, kas sukūrė komponentą, kas jį platina, kas nustatė komponento autentiškumą (viena tokių organizacijų, išduodančių autentiškumo sertifikatus, yra *VeriSign*).



7 pav. Naujo komponento įdiegimas

1.4. *Ajax* technologija

Šiuo metu statinis žiniatinklio dizainas naudojamas žymiai rečiau. Puslapių perkrovimas jau nėra priimtinas šiuolaikinėms žiniatinklo taikomosioms programoms, kuriose yra taikomos *JavaScript* ir *XML* technologijos.

Dar prieš kelis metus žiniatinklio taikomosios programos neturėjo pranašumo prieš įprastinę programinę įrangą, nes nepasižymėjo nei savo vartotojų sąsaja, nei interaktyvumu [28].

2005 m. pristatyta Ajax (angl. *Asynchronous JavaScript and XML* – liet. asinchronizuota *JavaScript* ir *XML*) metodologija – tai grupė, tarpusavyje susietų internetinių programų kūrimo technikų, naudojamų interaktyvių internetinių tinklalapių kūrimui. Ši technologija leidžia kurti programas labiau praturtintas animacija. Naudojant *Ajax*, tinklalapiai gali gauti duomenis iš serverio asinchroniškai, t.y. nebūtinas viso puslapio perkrovimas norit gauti vienos puslapio dalies duomenis. Duomenų gavimui naudojmas *XMLHttpRequest* objektas. J. J. Garrett šią technologiją apibūdina kaip tarpininką tarp vartotojo ir serverio [17].

Iš šiuolaikinių žiniatinklio taikomųjų programų, naudojančių *Ajax* principą, galima išskirti *Google Suggest*, *Google Maps* ar *Gmail*. Tai interaktyvios funkcionalios programos, kurios pilnavertiškai sąveikauja su vartotoju, informacija ekrane beveik akimirksniu yra atnaujinama, todėl nereikia laukti puslapio perkrovimo.

Ajax veikimo principas

Standartinėje internetinėje taikomojoje programoje už informacijos apdorojimą yra atsakingas serveris, o naršyklė atsako tik už sąveiką su vartotoju, užklausų perdavimą ir *HTML* išvedimą. Serveris vykdo reikalingą duomenų apdorojimą – gauna duomenis, apdoroja skaitmenis, sąveikauja su įvairiomis paveldėtomis sistemomis ir tik po to klientui pateikia *HTML* puslapį. Šis modelis paveldėtas iš pradinio žiniatinklio kaip hipertekstinės aplinkos taikymo modelis. Toks modelis yra techniškai prasmingas, bet neužtikrina efektyvios sąveikos su vartotoju. Kol serveris atlieka savo darbą, vartotojui tenka laukti.

Ajax pašalina *start-stop-start-stop* tipo sąveiką, nes tarp vartotojo ir serverio atsiranda dar vienas tarpininkas – *Ajax* varytuvė. Šis tarpinis lygmuo nustato, kurios užklausos gali būti apdorotos iš karto, o kurioms reikia kreipimosi į serverį. Vietoje to, kad naršyklė užkrautų puslapį sesijos pradžioje, ji užkrauna *Ajax* varytuvę, kuri yra parašyta *JavaScript* kalba ir paslėpta į nematomą rėmelį. Ši varytuvė atsako už vartotojo sąveikos formavimą ir sąveiką su serveriu vartotojo vardu. *Ajax* varytuvė leidžia asinchroniškai sąveikauti su vartotoju, nepriklausomai nuo sąveikos su serveriu. Dėl to vartotojui daugiau nereikia stebėti tuščio naršyklės lango, kol serveris atlieka darbą.

Kiekvienas vartotojos veiksmas, kurį įprastai atlieka *HTTP* užklausa, *Ajax* modelyje pereina į *Ajax* varytuvės *JavaScript* iškvietimo formą. Kiekvienas atsakas į vartotojo veiksmą (duomenų patikrinimą, duomenų redagavimą atmintyje ar kt.) atliekamas savarankiškai. Jeigu atsakui reikia informacijos iš serverio (pavyzdžiui, turi būti užkrautas papildomas sąsajos kodas, duomenų perdavimas apdorojimui ar

naujų duomenų gavyba), varytuvė atlieka užklausas asinchroniškai naudodama *XML,* nenutraukdama vartotojo sąveikos su taikomąja programa [17, 28, 41].

1.5. Kliento ir serverio srities programavimo technologijų palyginimas

2 lentelė. Kliento ir serverio srities programavimo technologijų palyginimas

Technologija	Privalumai	Trūkumai
Kliento srities	§ reaguojant į kiekvieną kliento užklausą nesikreipiama į serverį, taip sumažinamas tinklo apkrovimas ir sutaupomos energijos sąnaudos; § dėl spartaus taikomųjų programų veikimo greitai reaguojama į vartotojo užklausas [38]; § jei naršyklė nepalaiko *JavaScript* ar *VBscript,* gali būti naudojamos kitos alternatyvos (pavyzdžiui, *HTML*); § galima naudoti paruoštus nemokamus šablonus (*hotscript.com, javascript.com*) [8].	§ ne visos naršyklės palaiko scenarijų kalbas; § skirtingos naršyklės ir jų versijos, skirtingai traktuoja scenarijus (reikia daugiau testavimo); § kai nenaudojami paruošti scenarijų šablonai, o kuriami savi, jiems sukurti reikia nemažai laiko; § saugumas (kenkėjiškų programų įterpimas); § kliento srities taikomųjų programų kodas ne kompiliuojamas, o interpretuojamas, dėl to jis gali būti matomas pašaliniams; § iš kliento srityje veikiančios taikomosios programos negalima prieiti prie failų, direktorijų ar duomenų bazių [15].
Serverio srities	§ paslėptas kodas; § prieiga prie duomenų bazių; § serverio srityje veikiantis kodas yra nepriklausomas nuo naršyklės [41].	§ lėtesnės nei kliento srityje veikiančios taikomosios programos, jos reaguoja į kiekvieną vartotojo užklausą [41].

1.6. Išvados

Atlikus analizę nustatyta, kad kiekviena technologija turi privalumų ir trūkumų. Žiniatinklio taikomosios programos *baldų e.užsakymai* kūrimui geriausia būtų naudoti abi, ir kliento srities ir serverio srities programavimo technologijas. Nesudėtingų funkcijų ir dizaino elementų apdorojimą pavedant kliento mašinai, verslo veiklos logiką ir taisykles – serveriui.

Kliento srities funkcijoms realizuoti tinkamiausios yra *JavaScript* ir *Ajax* technologijos. *JavaScript* scenarijų kūrimas yra paprastesnis, jie veikia sparčiau nei *Java* apletai. *Ajax* technologija yra naudinga dėl asinchronizuoto veikimo, didesnio efektyvumo, galimybės kurti animuotus ir interaktyvius puslapio elementus. Kadangi *Ajax* nauja technologija, programuotojai vis dar kuria naujus arba tobulina esamus valdymo šablonus, kurie lengvai integruojami į žiniatinklio taikomąsias programas.

Serverio srities programos daliai realizuoti tinkamiausia yra *ASP.NET* technologija. *ASP.NET* programų kūrimui reikalingas laiko sąnaudas galima sumažinti, naudojant *Microsoft* korporacijos siūlomas, vartotojui draugiškas ir paprastas programavimo aplinkas (*IDE*). Šių *IDE* privalumai: grafinės sąsajos kūrimui galima naudoti „paimk ir padėk“ funkciją, mažesnė klaidų įvėlimo tikimybė, nes naudojamas kokybiškas komponentų kodas sukurtas *Micorsoft* projektuotojų. Statistiškai *ASP.NET* technologijos įsisavinimui reikia mažiau laiko nei *JSP*.

Už kitas technologijas *ASP.NET* taip pat pranašesnė dėl realizuoto objektinio programavimo, saugumo, stabilumo, paprastesnio puslapių palaikymo, dinamiškumo ir daugiagijo modelio. *ASP.NET* yra daug kalbų palaikanti technologija. Iš *ASP.NET* palaikomų kalbų tinklalapio kūrimui tinkamesnė *Visual Basic*, nes ji brandesnė ir spartesnė nei C# kalba. *Visual Basic* galima naudoti programuojant ir *AutoCAD* aplinkoje.

2. *CAD* TECHNOLOGIJŲ APŽVALGA

Įvadinė dalis

Šiame technologijų amžiuje sunkiai įsivaizduojamas pastatų, tiltų, mašinų ar kitų inžinerinių gaminių projektavimas be šiam tikslui skirtų automatizuotų projektavimo sistemų. Sukurtos galingos projektavimo sistemos žymiai palengvina architektų ir inžinierių darbą. Sistemų kūrimas tai nevienadienis procesas, jų plėtojimui ir palaikymui reikia milžiniškų išteklių tiek finansine, tiek technologine prasme. Sudėtingėjant inžineriniams sprendimams, nauji reikalavimai keliami ir kompiuterinio projektavimo sistemoms. Sukurta viena produkto versija jau po kelerių metų tampa nebeaktuali, taigi sistemų kūrėjams tenka nuolat taikytis prie besikeičiančių sąlygų.

2.1. Projektavimo procesų plėtra

Stebint projektavimo proceso plėtrą per paskutinius 20–25 m. galima išskirti keturis pagrindinius etapus. Pirmas etapas – tai rankinis projektavimas, itin sudėtingas ir ilgai trunkantis procesas. Šio proceso problemos ir trūkumai gana akivaizdūs: brėžinių netikslumas, nedidelis darbo greitis, prastos kopijavimo galimybės, klaidos ir kt. Antras etapas prasidėjo atsiradus programinei įrangai *AutoCAD*. Ši naujovė rinkoje labai supaprastino projektuotojų darbą, nes dingo tokios problemos kaip brėžinių rankinis taisymas, vizualinis netikslumas, kopijavimo problemiškumas. Programinė įranga suteikė projektuotojams galimybę paprasčiau ir greičiau atlikti savo darbus bei lengvai taisyti jų netikslumus, neperdarant brėžinio iš naujo. Atsiradus objektinėms projektavimo programoms, prasidėjo trečiasis etapas – trimatis objekto pateikimas. *AutoCAD* paprasčiausiai imituodavo brėžinio lapą ir braižymo įrankius. Siekiant brėžinyje pateikti itin sudėtingas sistemas ir prietaisus, jų vaizdavimas tapdavo labai sudėtingas. Programinės įrangos kūrėjai pradėjo ieškoti naujų projektavimo technologijų, kurios leistų, pavyzdžiui, sukurti sieną iš vos vienos nubrėžtos linijos, atspindinčios ir visus jos geometrinius, techninius, fizinius parametrus, o žiūrint į objektą būtų matomas visas jo trimatis vaizdas (angl. *three-dimensional – 3D*). Šie poreikiai inicijavo objektinės projektavimo technologijos atsiradimą. Dėl objektinės projektavimo technologijos nubrėžus liniją iš karto gaunamas konkretus pasirinktas objektas, jei tai statybinis – siena, perdanga, langas, jei baldų – lentynos, stalčiai. Lyginant *CAD* ir objektinės technologijos projektavimo procesus, svarbu paminėti, kad naudojantis pastaruoju kompiuterio ekrane sukurtame brėžinyje galima turėti ne tik grafinę, bet ir kitą informaciją apie brėžinio elementus – jų spalvą, svorį, tamprumo modulį, kainą, gamintoją ir t. t. Taip galima lengvai atlikti daugumą skaičiavimų, o keičiant parametrus daryti įtakos ir objekto geometriniam vaizdui. Sukaupus visą šią informaciją vienoje sistemoje, mažėja klaidų ir daugėja variantinio projektavimo galimybių [34].

Naujausias projektavimo etapas yra modeliavimas, dar labiau keičiantis mąstymą ir svarbiausius darbo proceso etapus. Kompiuterinis modeliavimas pirmiausia atsirado mechanikoje, aviacijoje, kosmoso pramonėje. Šiose srityse reikėjo tobulo tikslumo ir galimybės patikrinti daugybę kuriamo prietaiso

parametrų. Pirmieji modeliavimo bandymai projektuojant statinius buvo pritaikyti objektinio projektavimo technologijomis pagrįstose programose. Tačiau pagrindinis jų trūkumas buvo informacijos tarp objektų nebuvimas ir negalėjimas visiškai įvertinti projektuotojo darbo proceso. Modeliavimo procesas leidžia daugiau dėmesio skirti kūrybiniam darbui, kiekvienos srities specialistui optimaliai spręsti jam aktualius uždavinius ir problemas. Baldų modelis pirmiausia reiškia, kad visa informacija saugoma vienoje vietoje. Projektavimo erdvėje modeliavimas leidžia ne tik geriau įsivaizduoti kuriamą objektą, bet ir sumažinti projektavimo klaidų skaičių iki minimumo arba visiškai jų išvengti [34].

Viena iš kompanijų, siūlančių modernias kompiuterinio projektavimo technologijas, yra *Bentley Systems, Inc*. Įmonės specializacija – kompleksinių grafinių ir informacinių technologijų plėtra, programines įrangos, skirtos kompiuteriniam projektavimui, kūrimas. *Bentley* programiniai produktai nuo pat atsiradimo tobulinami grafinės aplinkos *MicroStation* bazėje. Tai profesionali grafinė sistema, skirta trimačiams ir dvimačiams objektams modeliuoti, braižyti, vizualizuoti, analizuoti, duomenų bazėms tvarkyti. Universali, atvira grafinė aplinka *MicroStation* praverčia visur, kur tik reikia grafinio duomenų apdorojimo ir valdymo, ryšio tarp grafinės ir atribucinės informacijos, lygiagretaus *CAD* ir *GIS* uždavinių sprendimo.

Šiuo metu *Bentley* yra vienas didžiausių pasaulyje automatizuoto projektavimo sistemų gamintojų vientisos ir integruotos *MicroStation* programines įrangos aplinkoje. Skirtingai nuo kitų kompanijų, kuriančių produkciją šiai rinkai, *Bentley* siūlo visą programinės įrangos produktų spektrą [33].

2.2. *CAD* programinis pritaikymas

Svarbus *CAD* sistemų privalumas yra tai, kad vartotojai programiniu būdu gali jas pritaikyti pagal savo poreikius.

3 lentelė. *CAD* programinės įrangos pritaikymas [44]

***CAD* programinė įranga**	**Pritaikymo sąsaja**	**Programavimo kalbos**
AutoCAD	*AutoLISP, VisualLISP, DCL* *ObjectARX* *AutoCAD VBA* *AutoCAD.Net*	*AutoLISP, VisualLISP, DCL* *Visual C++ 6.0, Visual C++ .NET* *Visual Basic 6.0, VBA* *Visual Basic .NET, Visual C# .NET,* *Visual C++ 6.0, Visual C++ .NET*

3 lentelės tęsinys

SolidWorks	*SolidWorks API* *eDrawings API* *FeatureWorks API* *PDMWorks Workgroup API* *PhotoWorks API* *SolidWorks Routing API* *SolidWorks Toolbox Browser API* *SolidWorks Utilities API* *SolidWorks Document Manager API* *SolidWorks macro files*	*Visual Basic 6.0* *VBA* *Visual Basic .NET* *Visual C# .NET* *Visual C++ 6.0* *Visual C++.NET*
SolidEdge	*SolidEdge API* *SolidEdge VBA*	*VB, VB.Net, C#.Net*
Pro/ENGINEER	*Pro/TOOLKIT*	*C*
	J-Link	*Java*
Autodesk Inventor	*Inventor API*	*VB, VB.Net, C#.Net*
Microstation	*MicroStation BASIC, JMDL, MDL (MicroStation Development Language),*	*VBA, Visual Basic C/C++, C#*

2.3. *AutoCAD* programinė įranga

AutoCAD – universali kompiuterinio projektavimo programinė įranga, skirta ne tik baldų, mechaninių detalių projektavimui, bet ji gali būti pritaikoma ir kuriant pastatų architektūrą, kraštovaizdžius ar žemėlapius. Pasaulinei rinkai ji buvo pristatyta prieš dvidešimt dvejus, o Lietuvoje pasirodė prieš aštuoniolika metų. Jos populiarumą lėmė ilgas gyvavimas rinkoje ir kaina. *AutoCAD* paskirtis yra palengvinti projektavimą, automatizuoti, susisteminti projektavimo etapus, projektinės dokumentacijos (brėžinių) ruošimą, modeliuoti sudėtingas dvimates ir trimates erdvines konstrukcijas [34].

Panaudojant programinį pritaikymą žymiai praplečiamos *AutoCAD* galimybės: galima automatizuoti tipinių užduočių (objektų brėžimo ar redagavimo) atlikimą, automatizuotai braižant sumažėja klaidų tikimybė, atsiranda galimybė verslo objekto modelio (angl. *Business Object Model – BOM*) informaciją susieti su duomenų bazėmis [32].

2.3.1 *AutoLISP* technologija

VisualLISP – specialiai sukurta darbui *AutoCAD* aplinkoje ir seniausiai joje naudojama programavimo kalba. *AutoLISP* – *VisualLISP* kalbos modifikacija, veikianti *AutoCAD* terpėje ir skirta praplėsti bei papildyti *AutoCAD* galimybes. *AutoLISP* turi nuosavą programavimo aplinką, kuri apima kompiliatorių, derinimo įrankį ir kitas papildomas priemones darbo produktyvumui pagerinti. Panaudojant *AutoLISP* programavimo kalbos priemones, galima kurti naujas komandas, jų kombinacijas, makroprogramas. *AutoLISP* veikia kaip interpretatorius, jis įvertina programos kodą, o po to pateikia atsakymą. *AutoLISP* skirtas paprastiems, nesudėtingiems uždaviniams spręsti, pavyzdžiui, parametrizuotų figūrų braižymui [44]. Linijos braižymo *AutoLISP* kodo pavyzdys:

```
(setq pt1 '(1 1) pt2 '(1 5))
(command "line" pt1 pt2 "")
```

Privalumai:

- Net ir neturint gilių programavimo žinių, *AutoLISP* kalbą lengva išmokti ir naudoti.
- Kalba buvo specialiai kurta *AutoCAD* komandų realizavimui, todėl turi platų *AutoCAD* funkcijų pasirinkimą.
- *AutoLISP* kalba sukurta programa gali būti naudojama tiesiogiai kaip *AutoCAD* aplinka arba kaip nuosava *AutoCAD* komanda.
- *AutoLISP* programų moduliai gali būti rašomi ir derinami atskirai.
- *AutoLISP* gali sąveikauti su grafinio objekto duomenimis (gali gauti informaciją arba ją keisti).

Trūkumai:

- Sudėtingus uždavinius sunku būtų realizuoti, nes tai scenarijų vykdymo kalba (nėra objektiškai orientuota).
- *AutoLISP* priklauso *AutoCAD* aplinkai, todėl negali būti vykdomas atskirai.

2.3.2 *VBA* technologija

Visual Basic – objektiškai orientuota programavimo kalba. Ji ganėtinai paprasta, sukurta specialiai tiems, kurie nesudėtingomis priemonėmis siekia kurti didesnes ir sudėtingesnes taikomąsias programas *Windows* aplinkoje. Kalba valdo objektus, su kuriais atliekami įvairūs veiksmai. Programos kodas pateikiamas procedūrose (paprogramėse), į kurias pagrindinė programa kreipiasi ir iškviečia atskirai. Procedūras galima rašyti formos, modulio ir klasės moduliuose [39].

Lyginant su *LISP*, *Visual Basic* turi platesnių galimybių *AutoCAD* taikomųjų programų kūrimui. *VB* dialektas – *VBA* (angl. *Visual Basic for Application*) kalba kartu su redaktoriumi yra integruota *Microsoft Office* programose ir grafinėje sistemoje *AutoCAD* (nuo *R14.01* versijos). Turėdamos *VBA* įdiegtį šios programos per objektus gali komunikuoti viena su kita [4].

Microsoft Office programos *VBA* projektus, programas, makroprogramas laiko specifiniame dokumente, *AutoCAD* tam tikslui turi atskirą projekto failą su *.dvb* plėtiniu. Dėl to *AutoCAD VBA* projektai gali būti atidaromi ar uždaromi net braižymo vykdymo metu.

2.3.3 *ObjectARX* technologija

ObjectARX® (angl. *AutoCAD Runtime Extension*) – tai kompiliuojamosios kalbos programavimo aplinka, skirta *AutoCAD* taikomųjų programų kūrimui. *ObjectARX* programavimo aplinkos dinamiškai susietų bibliotekų (angl. *dynamic link libraries – DDL*) failai vykdomi toje pačioje adreso erdvėje kaip ir pati *AutoCAD* programa. *DLL* failai tiesiogiai operuoja su *AutoCAD* kodu ir pagrindinėmis duomenų struktūromis. Išnaudodamos *AutoCAD* atvirosios architektūros privalumus, bibliotekos užsitikrina tiesioginį priėjimą prie *AutoCAD* duomenų bazių struktūrų, grafinės sistemos, braižymo variklio, darbo metu išplėčia *AutoCAD* klases. Bibliotekose aprašytos naujos komandos veikia lygiai taip pat kaip *AutoCAD* komandos. *ObjectARX* bibliotekas galima naudoti kartu su kitomis *AutoCAD* programavimo sąsajomis (*AutoLISP* ar *VBA*) [2].

Lyginant visas tris *AutoCAD* programinio pritaikymo technologijas, *ObjectARX* pati sudėtingiausia. *ObjectARX* programavimui dažniausi naudojama *C++* programavimo kalba. Nesudėtingai figūrai nubraižyti *AutoLISP* ir *VBA* užtenka 2-3 kodo eilučių, norint tai realizuoti *ObjectARX*, gali prireikti iki 100 kodo eilučių. Taip yra todėl, kad kuriant programas, pagrįstas *ObjectARX* technologija, yra naudojmas šakninis *AutoCAD* kodas. *ObjectARX* technologija paremtos programos kūrimas užima daug laiko, reikia išmanyti *AutoCAD* architektūrą, suprasti, kokie objektai ją valdo, kaip vyksta procesai *AutoCAD* kodo lygmenyje [45].

2.4. *KitchenDraw*

KitchenDraw – tai specializuota programa, skirta baldų ir interjero projektavimui. Programa suteikia galimybę sudėlioti spinteles, stalus, kėdes ir kitus virtuvės ar biuro baldus pasirinktoje erdvėje, pateikti matmenis automatiniu ar rankiniu būdu. Programa taip pat turi galimybę sudaryti sąmatas ir specifikacijas, atlikti pardavimų ir užsakymų apskaitą. Visi reikalingi elementai pasirenkami iš *KitchenDraw* bibliotekų ar katalogų, kuriuos galima atsisiųsti iš oficialaus *KitchenDraw* puslapio. Programa leidžia lengvai pasirinkti kuriamos erdvės aukštį, ilgį, plotį, sienų tekstūras (yra galimybė sienų tekstūrą pasirinkti pagal *JPG* ar *BMP* paveikslėlius). Jei programoje nėra pageidaujamo baldo, jį galima

susikurti *KitchenDraw* aplinkoje arba importuoti iš *AutoCAD, Autodesk Inventor, 3ds Max* programų. Baldams taip pat galima priskirti savo tekstūras, įkeltas tiesiog nusifotografavus turimą tekstūros pavyzdį [27].

***KitchenDraw* privalumai:**

- Programą lengva naudoti: baldus ir įrenginius iš bibliotekų galima sudėlioti „paimk ir padėk“ principu.
- Galima greitai redaguoti pasirinktus elementus.
- Dėl aukštos kokybės vizualizacijos bet kuriuo projektavimo metu galima peržiūrėti, kaip atrodo kuriamo projekto fotorealistinis vaizdas. Tokiu būdu lengviau įsivaizduoti, kaip viskas atrodys realybėje [27].

***KitchenDraw* trūkumai:**

- Nėra galimybės kurti papildomų programinių modulių, makroprogramų.
- Programa neturi sąsajos su duomenų bazėmis.
- Yra ribotos galimybės projektuotojui pačiam projektuoti baldus.
- Trečiųjų šalių (*AutoCAD* ar *Autodesk Inventor*) programose sukurtų baldų importavimas į *KitchenDraw* yra neefektyvus ir ekonomiškai nenaudingas.

2.5. Išvados

Apžvelgtos dvi skirtingos projektavimui skirtos technologijos: specializuota baldų projektavimo programa *KitchemDraw* ir universalus projektavimo įrankis *AutoCAD*. *KitchemDraw* programa yra labai ribotų galimybių, joje galima modeliuoti baldų komplektus tik iš programoje esamų baldų, prireikus naujų, siūloma kurti trečiųjų šalių produktuose ir integruoti juos, o tai apsunkina darbo procesą.

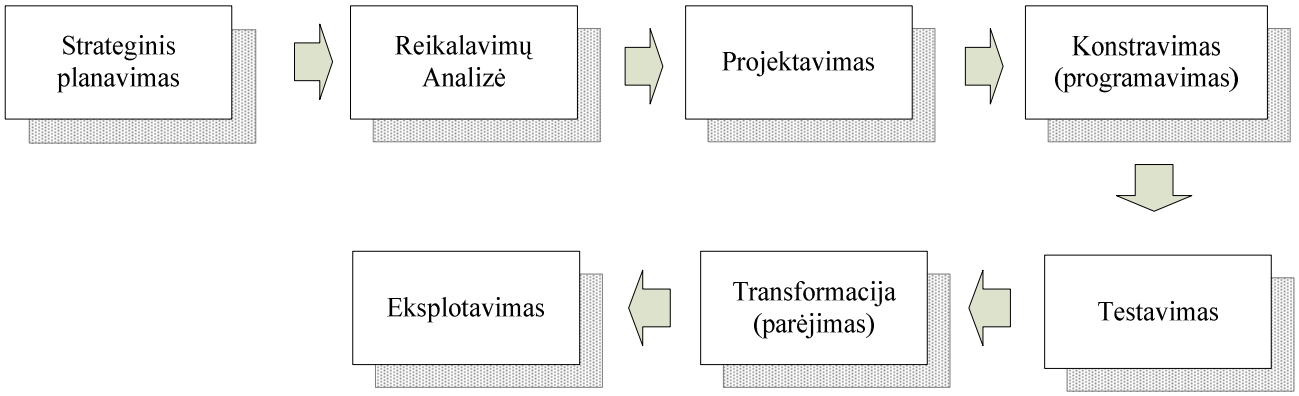
AutoCAD – universalus ir galingas įrankis, puikiai tinkantis stambiems projektavimo uždaviniams spręsti. Jis palengvina projektavimą, leidžia automatizuoti ir susisteminti projektavimo etapus, palengvina projektinės dokumentacijos ruošimą.

Automatizuoto baldų projektavimo makroprogramos kūrimui tinkamesnė yra *VBA* nei *AutoLISP* technologija, nes *VBA* galima realizuoti objektinį programavimą, veiklos logiką galima skaidyti į specialius klasių ar formų modulius. *VBA* programose formų kūrimas ir palaikymas daug paprastesnis, galima naudoti *COM* bibliotekas, integruoti su kitomis programomis (pavyzdžiui, *MS Excel*). *VBA* ir *AutoCAD* bendravimas vyksta per *ActiveX* automatizuotą sąsają, dėl to *VBA* programos veikia sparčiau nei *AutoLISP*.

3. SISTEMOS REALIZACIJOS TEORINIS MODELIS

3.1. Sistemos kūrimo gyvavimo ciklas

Krioklio tipo sistemos kūrimo gyvavimo ciklas (žr. 8 paveikslą) yra iš anksto nustatytas sistemų kūrimo būdas, kuris prasideda strateginiu planavimu, toliau seka reikalavimų analizė, projektavimas, konstravimas, testavimas ir įgyvendinimas (eksploatavimas). Konkretūs etapai gali skirtis priklausomai nuo pasirinktos metodologijos.



8 pav. Sistemos kūrimo gyvavimo ciklas

3.2. Reikalavimų analizė

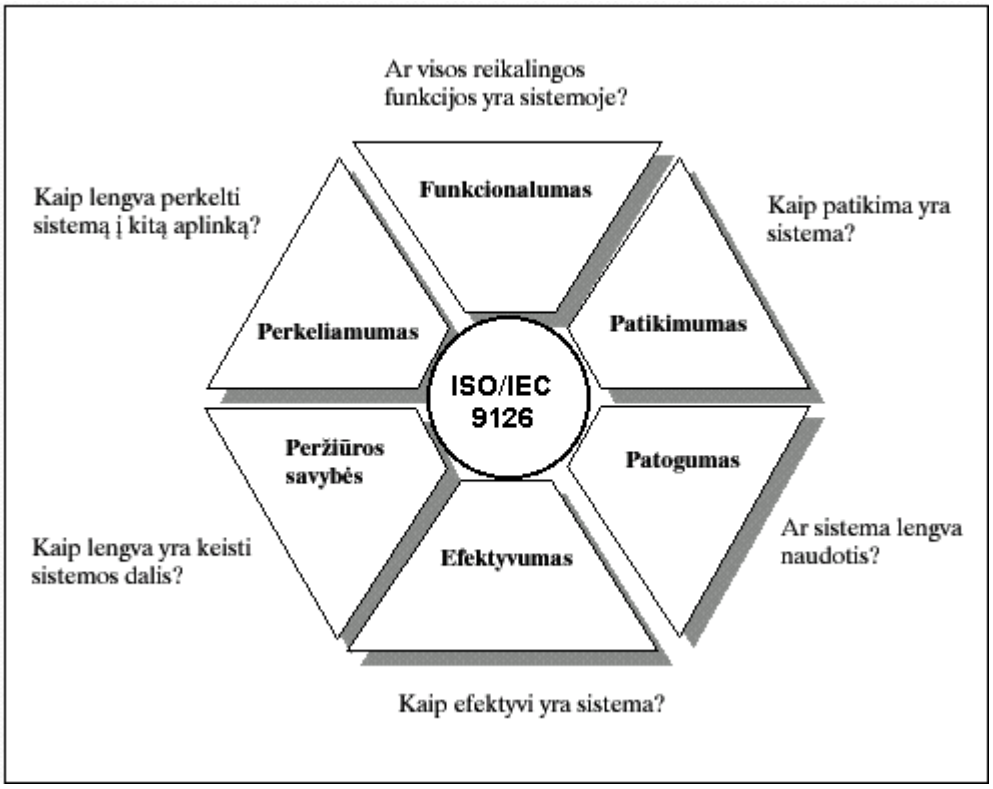
Reikalavimas – funkcionalumas arba sąlyga, kurią turi įgyvendinti sistema. FURPS – tai reikalavimų modelis, apibrėžiantis tai, ką vartotojas gali pasakyti apie sistemą:

- funkcionalumas (angl. *Functionality*),
- patogumas (angl. *Usability*),
- patikimumas (angl. *Reliability*),
- efektyvumas (angl. *Efficiency*),
- priežiūros savybės (angl. *Maintainability*),
- perkeliamumas į kitą aplinką (tiek organizacine, tiek technine, tiek programine prasme) (angl. *Portability*).

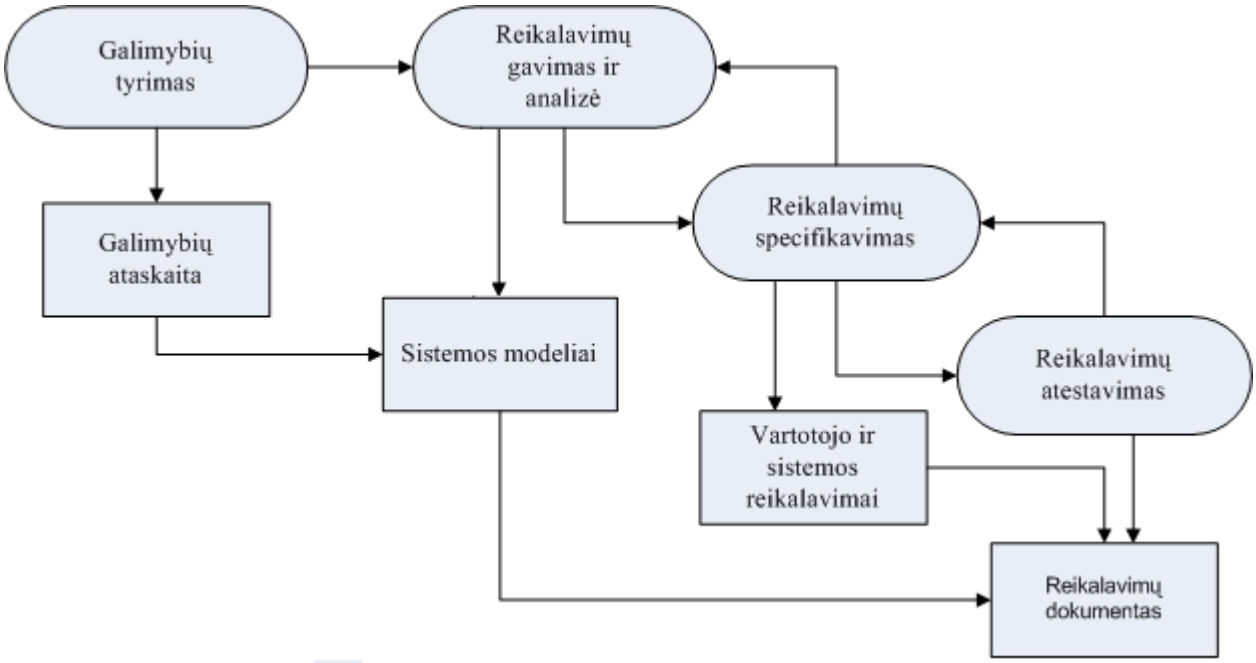
Sistemos kokybės kriterijai

Norint, kad sistema būtų kokybiška ir pripažinta rinkoje, projektuojant reikia taikyti pasaulyje žinomus ir jau apibrėžtus standartus. Pasaulyje žinomiausi ir taikomi daugelyje sričių yra ISO standartai. Tarptautinų standartų organizacija (angl. *International Organization for Standardization* - ISO) yra pati

didžiausia pasaulyje standartų kūrėja ir vystytoja. ISO naudojamas 157 šalyse. Sistemos kokybei yra taikomas ISO 9126 standartas. Standartas pateikia 6 kokybės charakteristikas (žr. 9 paveikslą) [22].



9 pav. Sistemos kokybės kriterijai



10 pav. Reikalavimų analizės etapai

Galimybių tyrimai nustato, ar verta kurti siūlomą sistemą. Trumpas koncentruotas tyrimas tikrina:

- § ar sistema atitinka organizacijos tikslus;
- § ar sistema gali būti sukurta naudojant turimas technologijas ir vidinį biudžetą;
- § ar nauja sistema bus suderinama su jau naudojamomis sistemomis.

Reikalavimų gavimas ir analizė – tai pagrindinis sistemos kūrimo etapas. Atliekant reikalavimų analizę reikia atsakyti į tokius klausimus: koks pagrindinis tikslas (lūkesčiai), kokie sistemos komponentai, kas ir ką joje turi daryti, o ne kaip daryti. Norint sužinoti apie taikymo sritį, paslaugas, kurias turėtų teikti sistema, ir sistemos darbo apribojimus, reikalavimų gavimo ir analizės metu kreipiamasi į su užsakovais dirbantį techninį personalą ir į pačius užsakovus (galutinius vartotojus, vadybininkus, palaikymo/aptarnavimo inžinierius, srities ekspertus ir t.t.) [20, 40].

Reikalavimų analizės problemos:

- § Suinteresuoti asmenys dažnai nežino tikslių savo pageidavimų.
- § Užsakovai reikalavimus išreiškia savo terminais.
- § Atliekant analizę nepakankamai įtraukiami užsakovai.
- § Skirtingi suinteresuoti asmenys gali kelti tarpusavyje nesuderinamus reikalavimus (neefektyvios komunikacijos priemonės).
- § Analizės metu besikeičiantys ar atsirandantys nauji užsakovo reikalavimai apsunkina darbo procesą.

3.3. Projektavimas

Projektavimas – tai sprendimų seka, kuri lemia, koks bus naujas produktas. Projektavimo metu atsižvelgiama į produkto kūrimui naudojamas technologijas ir į jų kainas.

Projektavimas apima:

- § objektų diagramų kūrimą ir detalizavimą;
- § komponentų diagramų kūrimą;
- § diegimo plano sudarymą;
- § vartotojo sąsajos ir prototipų projektavimą;
- § testų kūrimą;
- § dokumentacijos kūrimą.

Projektavimo fazėje turi būti atsižvelgta ne vien į įgyvendinimą, bet ir į kitas programinės įrangos gyvavimo fazes, t.y. programų keitimą, priežiūrą, diegimą ir kitas su jomis susijusias veiklas.

Šiuolaikinių informacinių sistemų projektavimui dažniausiai naudojama unifikuota modeliavimo kalba – *UML* (angl. *Unified Modeling Language*), sukurta remianti *OMG* (angl. *Object Management Group)* specifikacija. Kalba skirta aprašyti kuriamą sistemą ar jos dalį norimame abstrakcijos lygmenyje. *UML* yra kalba, todėl egzistuoja griežtos grafinių elementų panaudojimo taisyklės. *UML* modelis aprašo, ką sistema turi daryti, bet jis nepasako kaip realizuoti tą sistemą. *UML* komponentai susideda iš įvairių grafinių elementų, kurie jungiami į diagramas. Diagramų tikslas – įvairiais pjūviais aprašyti kuriamą sistemą, t.y. sudaryti sistemos modelį [9, 36].

***UML* diagramų tipai**

Panaudos atvejų (angl. *use case*) *UML* diagrama, aprašo, ką projektuojama sistema gali atlikti, kartu aprašydama ir išorinius sistemos veikėjus (aktorius). Kuriant dideles sistemas ši diagrama skirstoma į posistemes. Pagrindinis šios diagramos elementas – panaudos atvejis (užduotis), aprašantis aibę panašių sąveikos scenarijų. Kiekvienas panaudos atvejis turi vieną pagrindinį ir keletą šalutinių scenarijų, kurie aprašomi dinaminėmis *UML* diagramomis (sekų, bendradarbiavimo, veiklos).

Būsenų (angl. *state*) diagrama vaizduoja svarbiausias verslo ar veiklos sistemos būsenas ir tų būsenos kitimą toje pačioje sistemoje. Būsenos diagrama atvaizduoja verslo ir operatyvinius sistemos komponentų srautus. Ji parodo visapusišką kontrolės srautą. Veiksmo būsena – tai tam tikras veiksmas, kurį atlieka objektas. Pradinė būsena yra būsena, kurioje prasideda aprašoma veikla. Galinė būsena yra viena iš galimų veiklos pabaigos būsenų.

Sekų (angl. *sequence*) diagramos iliustruoja objektų, jų būsenų, veiksmų lygiagretų išsidėstymą laiko atžvilgiu. Objektai gali būti žmonės, kompiuteriai, procesai, organizaciniai vienetai, įmonės ir t.t. Dažniausiai sekų diagrama vaizduoja objektų vienas kitam siunčiamų pranešimų seką. Pranešimų siuntimo laikas ir tvarka yra griežtai apibrėžti.

Klasių (angl. *class*) diagrama skirsto objektus į kategorijas. Klasė aprašo objektų, kurie turi panašius atributus ir vienodą elgseną ar metodus, grupę. Klasių diagramoje specifikuojamas sistemos duomenų modelis, objektų klasės, jų atributai, su jomis atliekamos operacijos ir ryšiai tarp klasių [31].

***UML* privalumai:**

- Kuriama sistema yra profesionaliai projektuojama ir dokumentuojama dar prieš realizaciją, tiksliai žinoma, ko siekiama.
- Lengva nustatyti projektavimo klaidas.
- Geras sistemos projektas užtikrina teisingą ir efektyvų realizavimą.
- *UML* leidžia aprašyti sistemą norimu detalumu ir pjūviais.

- Atlikti sistemos modifikacijas yra žymiai lengviau turint sistemos *UML* dokumentaciją.
- *UML* leidžia žmonėms iš šalies greitai suvokti sistemą.
- Visuotinai pripažintas standartas [9, 36].

3.4. Sistemos konstravimas

Sistemos konstravimas apima:

- programinės įrangos vieneto (PĮV) ir duomenų bazės (DB) kūrimą ir dokumentavimą;
- testų ir testinių duomenų kiekvienam PĮV ir DB kūrimą ir dokumentavimą;
- testavimą;
- vartotojo vadovo kūrimą;
- PĮ kodo ir testų rezultatų įvertinimą [20].

Testavimas

Programinės įrangos testavimas – tai empirinis techninis tyrimas, atliekamas norint sužinoti, ar programinė įranga atitinka jai keliamus reikalavimus. Testavimo metu naudojami iš anksto paruošti testavimo scenarijai ir sąlygos.

Juodosios dėžės (angl. *Black-box*) testavimas yra vienas praktiškiausių ir dažniausiai naudojamų testavimo tipų. Pagrindinis šio testavimo bruožas yra tai, kad atliekamas testavimas nesigilinant, kas yra programos viduje, tiesiog susikoncentruojant į programos testavimą iš vartotojo pusės. Pagrindiniai šio tipo testavimo uždaviniai yra įsitikinti, kad sistema veikia pagal reikalavimus ir kad sistema pateisina vartotojo lūkesčius [40].

Netinkamą sistemos veikimą apibūdinančios sąvokos:

- gedimas (angl. *defect*) – kodo klaida, kai neteisingai vykdomi reikalavimai;
- klaida (angl. *error*) – netikslumas, nenumatytas atvejis;
- riktas (angl. *bug*) – tai bet kokia netinkamo ar nenumatyto sistemos veikimo priežastis;
- triktis (angl. *failure*) – sistemos ar komponento nesugebėjimas atlikti reikalaujamą užduotį;
- trūkumas (angl. *fault*) – situacija, būsena ar įvykis, kai sistema veikia nenumatytai ar netinkamai.

3.5. Transformacija ir eksploatavimas

Transformacija reiškia sistemos elementų integravimą su technine įranga ir kitomis sistemomis. Nauja sistema tampa organizacinės struktūros dalimi. Transformacija apima mokymus, naujos organizacinės struktūros apibrėžimą ir vaidmenis. Eksploatavimo procesas aprašo organizacijos, eksploatuojančios sistemą realioje aplinkoje (realiems vartotojams), veiksmus. Sistema eksploatuojama realioje aplinkoje taip, kaip nustatyta vartotojo vadove. Į eksploatavimo sąvoką įeina pagalba vartotojams, konsultavimas, laikini sprendimai [40].

4. EKSPERIMENTINĖS SISTEMOS KŪRIMO REALIZACIJA

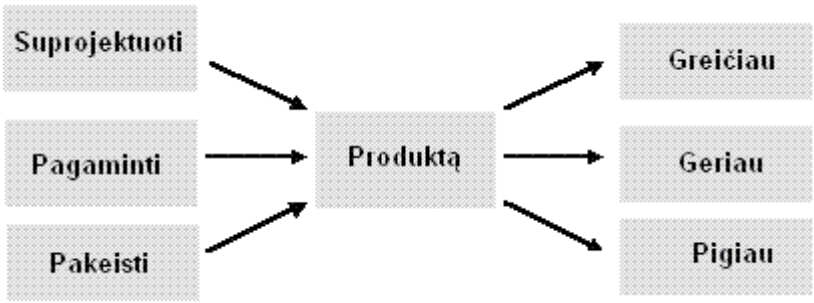
Įvadas

Šiame skyriuje atliekama baldų projektavimo ir pardavimo procesų analizė. Pagal gautas išvadas siūloma optimizuoti šiuos procesus, sukuriant e.užsakymo formavimo ir automatizuoto projektavimo sistemą. Sistema kuriama remiantis atlikta technologijų analize ir trečiame skyriuje nagrinėtu sistemos kūrimo gyvavimo ciklu. Kiekvienas šio ciklo etapas detaliai aprašomas, pateikiamos verslo procesų, sistemos panaudos atvejų, klasių diagramos.

4.1. Sistemos reikalavimų analizė

Kiekviena įmonė turi savo strategiją ir tikslus. Pagrindinis įmonės tikslas – optimaliomis sąnaudomis pasiekti maksimalų pelną. Gaminant produktą įmonei yra svarbūs tokie faktoriai, kaip minimalus brokas, trumpi, bet realūs užsakymo vykdymo laikotarpiai, minimali atsargų apimtis.

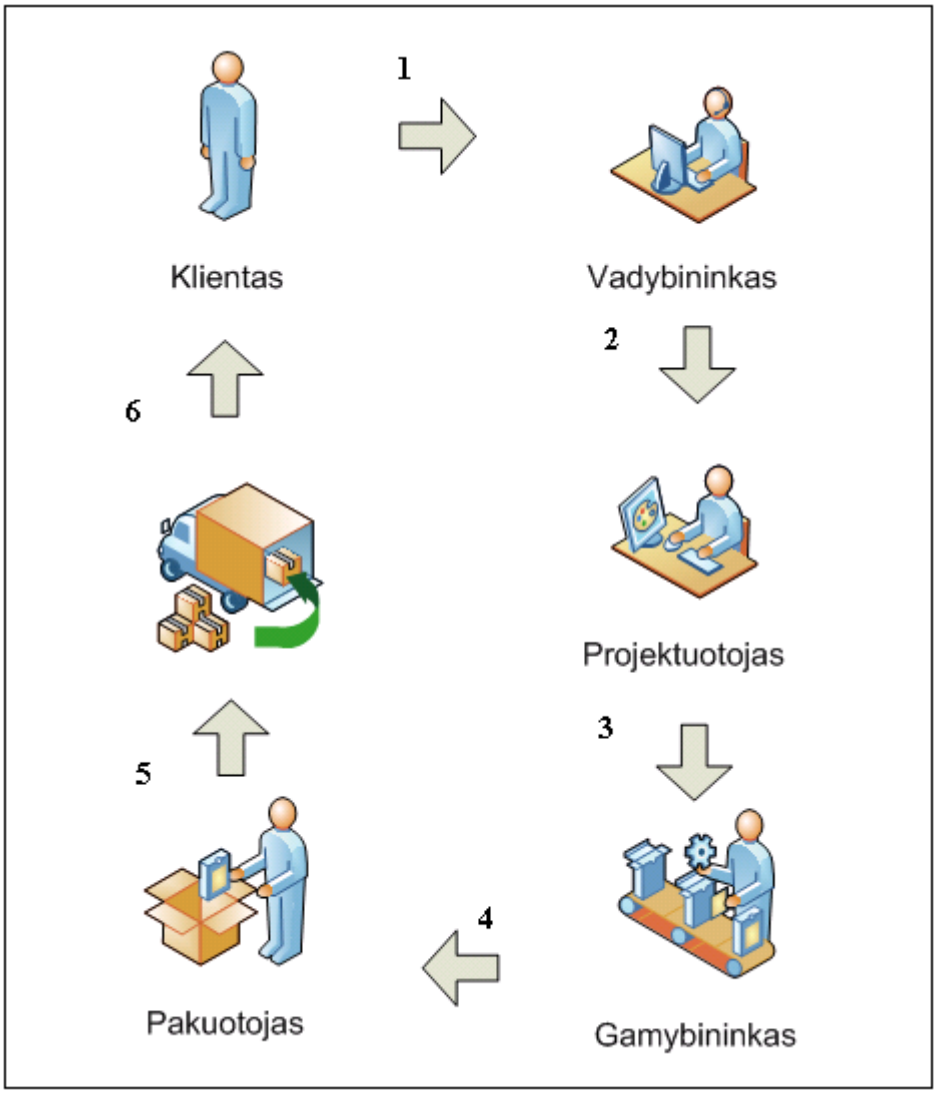
Pagrindiniu strategijos kintamuoju tampa produktas. Suprojektavus, pagaminus, pakeitus produktą, reikia siekti, kad procesas vyktų greičiau, geriau ir pigiau [1, 26].



11 pav. Produktas ir siekiamybės

4.1.1 Baldų verslo procesai

Baldų gamybos verslas apima ne tik gamybos ir projektavimo, bet ir daug įvairių verslo veiklos procesų. Tai ir užsakymų priėmimas, ir tiekimo užsakymai, rezervavimas, pristatymas klientui ir daugelis smulkių procesų, sudarančių vientisą verslo grandinę. 12 paveiksle pateikta nenutrūkstama grandinė nuo užsakymo pateikimo iki baldų pristatymo klientui.



12 pav. Baldų verslo procesų diagrama

1) *Informacijos pateikimas*. Klientas kreipiasi į vadybininką (gamintoją), kad šis pagal jo pateiktus reikalavimus pagamintų baldų komplektą.

2) *Užsakymo formavimas*. Vadybininkas, išanalizavęs gautą informaciją, suskaičiuoja kainą, suformuoja užsakymą ir jį perduoda projektuotojui.

3) *Baldų projektavimas*. Turėdamas aiškius reikalavimus projektuotojas suprojektuoja baldų modelį, modelio eskizą perduoda gamybininkui.

4) *Baldų gamyba*. Gamybininkas pagal projekto eskizus išpjauna plokštes, paruošia baldų komplekto detales ir jas perduoda pakuotojui.

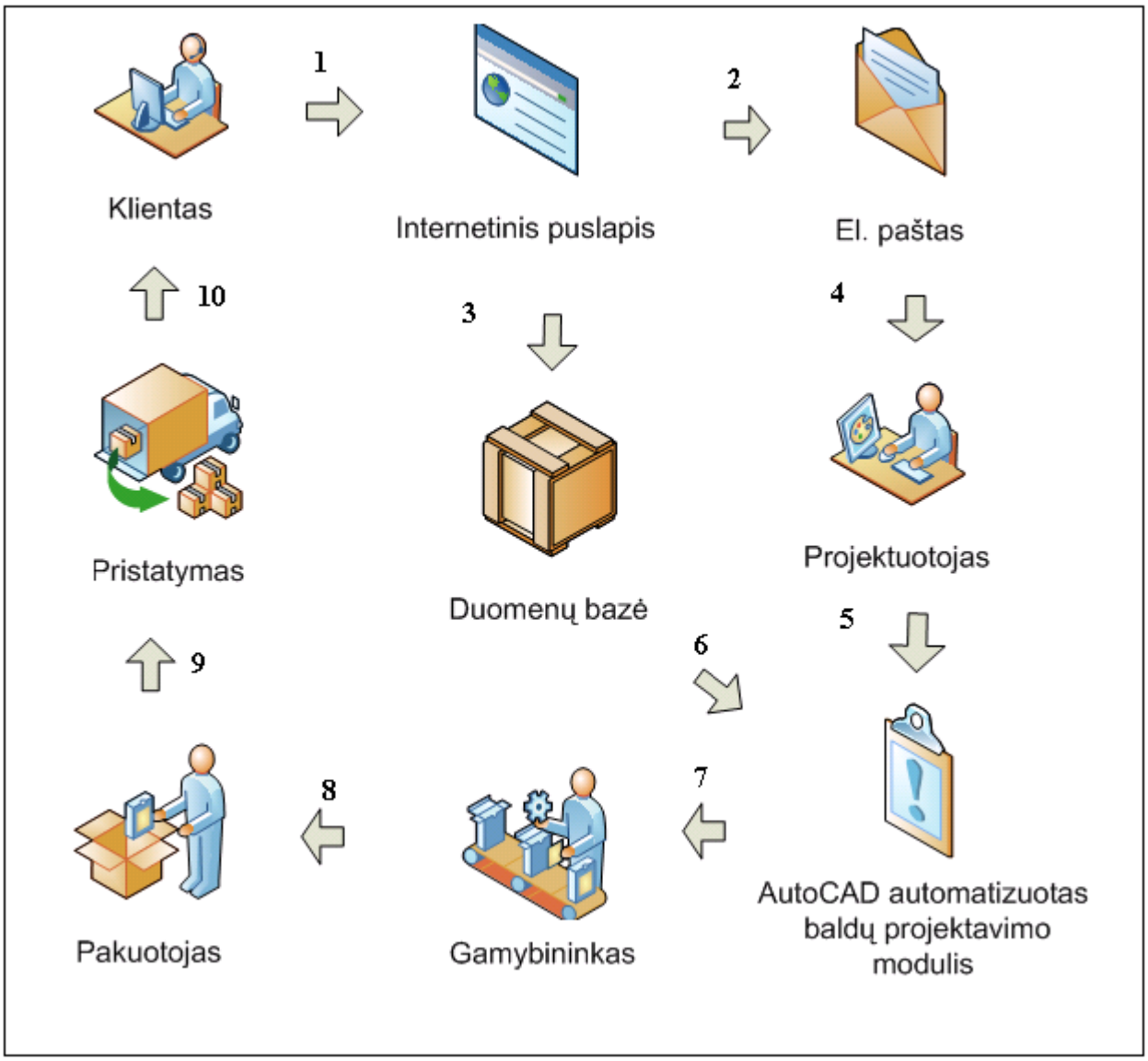
5) *Pakavimas*. Paruoštos baldų detalės supakuojamos ir perduodamos transportavimui.

6) *Transportavimas.* Baldai pristatomi klientui.

4.1.2 Verslo procesų optimizavimas

Iš 12 paveiksle pateiktos baldų verslo procesų diagramos matyti, kad kai kurie procesai nėra efektyviai organizuoti, pavyzdžiui, kliento poreikius nusakančios informacijos surinkimui ir analizavimui,

dokumentavimui ir užsakymo formavimui reikia didelių laiko išteklių. Norint optimizuoti šiuos procesus, tikslingiau būtų naudoti kompiuterizuotą sistemą. Tinkamai paruošta, žiniatinklyje veikianti užsakymų priėmimo ir formavimo sistema galėtų atlikti vadybininko darbus. Kita veikla, kurią būtina optimizuoti – projektavimas. Rankinis projektavimas turėtų būti pakeistas automatizuotu. Gavus kliento užsakymą, projektuotojui nebereikėtų kiekvieną kartą atlikti projektavimo darbų. Juos atliktų automatizuoto projektavimo sistema. Optimizuotos veiklos procesų diagrama pateikiama 13 paveiksle.



13 pav. Optimizuotos veiklos procesų diagrama

1) *Informacijos pateikimas*. Klientas, prisijungęs prie internetinio tinklalapio, išsirenka baldus, atlieka jų konfigūraciją, pasirenka fasado plokštės tipą, nurodo matmenis, užpildo elektroninę užsakymo formą.

2) *Užsakymo formavimas*. Gavęs signalą apie pateiktą informaciją, tinklalapio serveris suformuoja elektroninį pranešimą, parduoda jį pašto serveriui, kad šis nusiųstų pranešimą projektuotojui.

3) *Duomenų išsaugojimas*. Kliento pateikta užsakymo informacija išsaugoma duomenų bazėje.

4) *Elektroninis pranešimas*. Pašto serveris nusiunčia elektroninį pranešimą projektuotojui.

5) *Pasiruošimas užsakymo įgyvendinimui*. Gavęs pranešimą projektuotojas *AutoCAD* programoje iškviečia automatizuoto baldų projektavimo makroprogramą.

6) *Duomenų įkrovimas į makroprogramą*. Projektuotojas įkrauna duomenis ir pasiruošia užsakymo įgyvendinimui (atsiverčia užsakymą pagal elektroniniame pranešime gautą numerį).

7) *Automatizuotas projektavimas*. Projektuotojas paspaudžia braižymo mygtuką ir *VBA* makroprograma nubraižo projekto modelį. Modelio eskizai perduodami gamybininkui.

8) *Baldų gamyba*. Gamybininkas pagal projekto eskizus paruošia baldų komplekto detales ir jas perduoda pakuotojui.

9) *Pakavimas*. Paruoštos baldų detalės supakuojamos ir perduodamos transportavimui.

10) *Transportavimas.* Baldų detalės pristatomos klientui.

Tokiu būdu savo veiklą optimizavusi baldus gaminanti įmonė gautų nemažai ekonominės naudos:

- § Gamintojui būtų pateikta tiksli kliento poreikių informacija.
- § Nebereikėtų tiesioginio kliento ir gamintojo bendravimo.
- § Būtų sumažintos komunikavimo, užsakymo formavimo ir projektavimo laiko sąnaudos.
- § Projektuojant automatizuotai būtų išvengiama žmogiškojo faktoriaus klaidų.
- § Nebereikėtų vadybininko paslaugų.
- § Sumažėtų projektuotojo darbo krūvis.
- § Atsiradus e.užsakymo pateikimo galimybei padaugėtų klientų.
- § Sumažėtų užsakymo įvykdymo laikas.
- § Padidėtų baldų įmonės produktyvumas.

4.2. Kokybės reikalavimai kuriamai sistemai

Sistemos, skirtos baldų e.užsakymo formavimui ir automatizuotam projektavimui, kūrimas ir diegimas yra vienas iš svarbių ir atsakingų įmonės veiklos optimizavimo elementų.

Sistemai keliami suprantamumo, patogumo, funkcionalumo, patikimumo ir lankstumo reikalavimai [1, 14]. Ypač reikia išskirti baldų komplektavimo ir konfigūravimo internetinėje erdvėje pakankamą funkcionalumą, paprastą duomenų (informacijos) įvedimą ir duomenų modifikavimą, t. y. tas programinės įrangos savybes, kurios svarbios klientui pateikiant e.užsakymą.

Sistemos naudotojams-projektuotojams svarbu, kad sistema leistų saugiai kaupti ir saugoti duomenis (baldų e.užsakymus). Tai aktualu duomenų analizei ir veiklos planavimui. *VBA AutoCAD* makroprograma turi pasižymėti lankstumu, t. y. turi leisti projektuotojams patiems kurti naujus baldų projektavimo modulius, skirtus naujai iškilusiems projektavimo uždaviniams spręsti. Makroprogramoje turi būti modifikavimo ir išplėtimo galimybės.

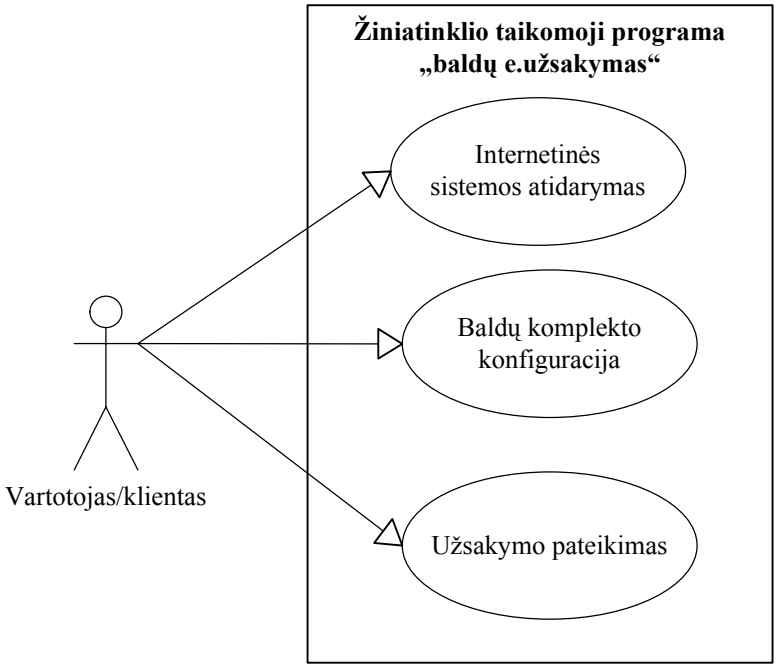
4.3. Sistemos projektavimas

Projektavimo metu buvo apibrėžta sistemos architektūra, komponentai, jų sąsajos ir kitos sistemos charakteristikos. Kadangi sistema yra dviejų dalių, suformuojamos ir pateikiamos atskiros žiniatinklio taikomosios programos *baldų e.užsakymas* ir *AutoCAD VBA* makroprogramos panaudos atvejų ir klasių diagramos. Taip pat pateikiama bendra sistemos naudojimo sekų diagrama.

4.3.1 *Baldų e.užsakymas* panaudos atvejai ir specifikacijos

Žiniatinklio taikomąją programą *baldų e.užsakymas* apibūdina panaudos atvejų diagrama (žr. 14 paveikslą), kurią sudaro trys panaudos atvejai:

- *baldų e.užsakymas* programos atidarymas;
- baldų komplekto konfigūracija ir detalizavimas;
- užsakymo pateikimas.



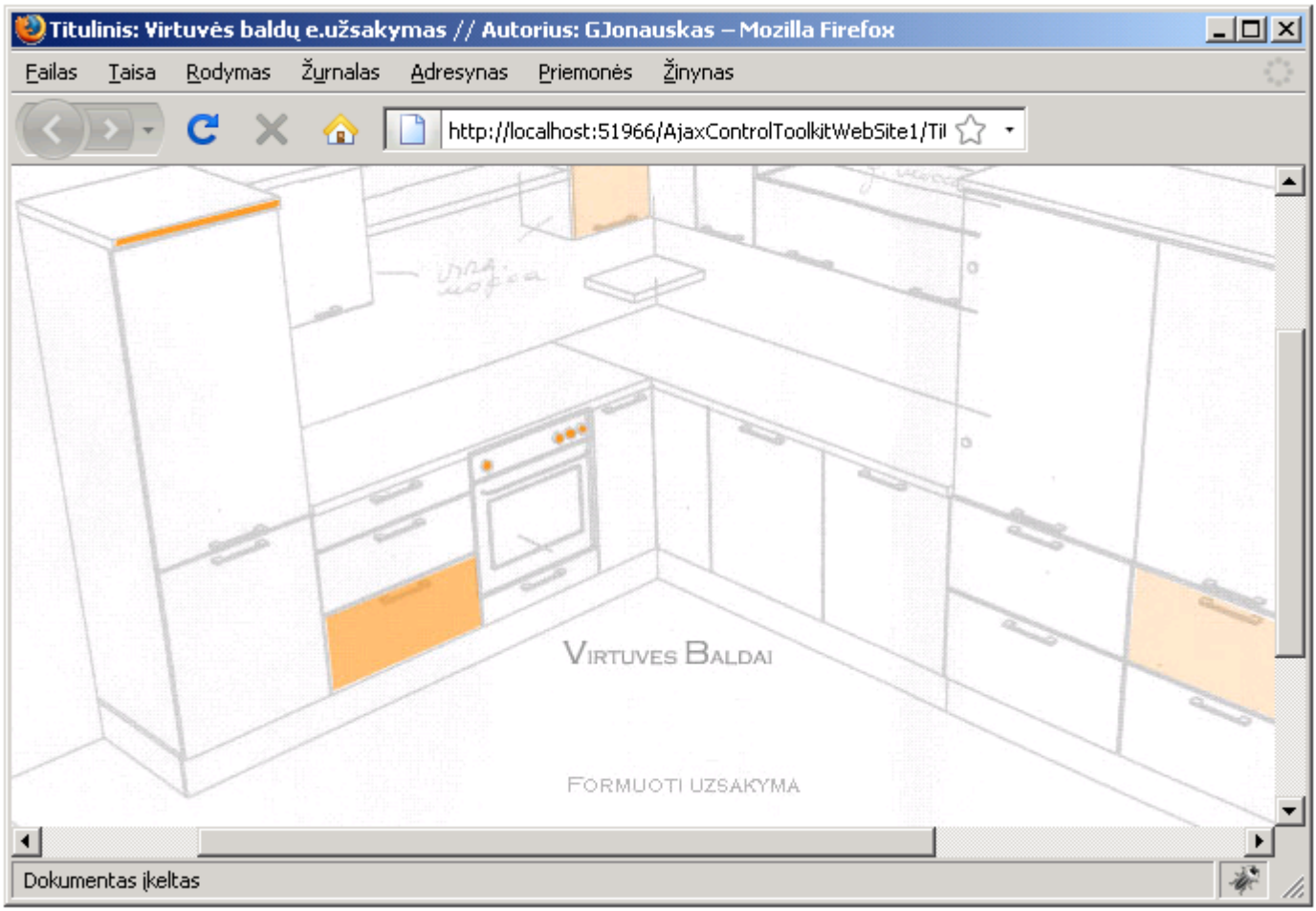
14 pav. *Baldų e.užsakymas* panaudos atvejų diagrama

Panaudos atvejis nr.: UC101

Panaudos atvejis: *Baldų e.užsakymas sistemos atidarymas*

Aktoriai: Vartotojas/klientas

Aprašymas: Vartotojas atidaro internetinę naršyklę ir įveda *URL* adresą. Šią užklausą naršyklė siunčia į serverį, šis apdoroja ir siunčia atsakymą atgal į naršyklę. Gavusi atsakymą naršyklė atvaizduoja titulinį langą *Virtuvės baldų e.užsakymas* (žr. 15 paveikslą).



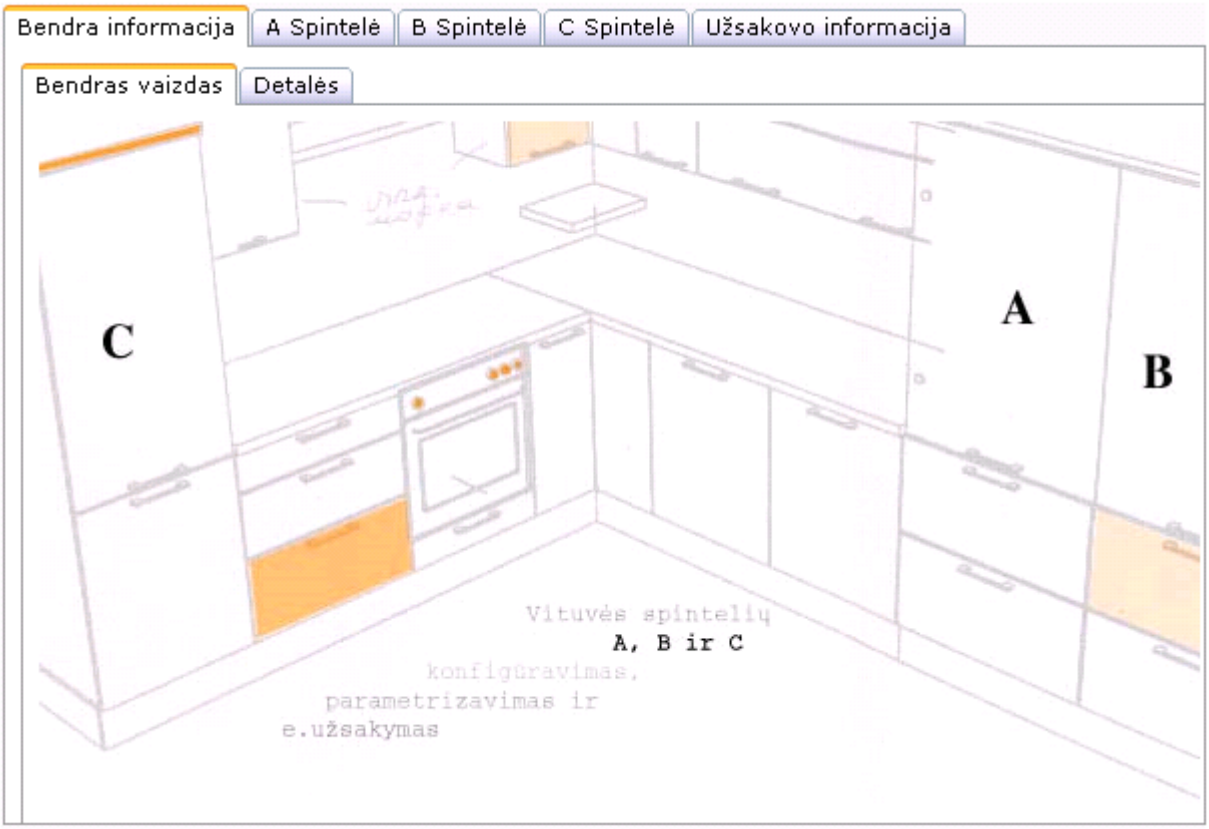
15 pav. Pirmas langas

Panaudos atvejis nr.: UC102

Panaudos atvejis: *Baldų komplekto konfigūracija ir detalizavimas*

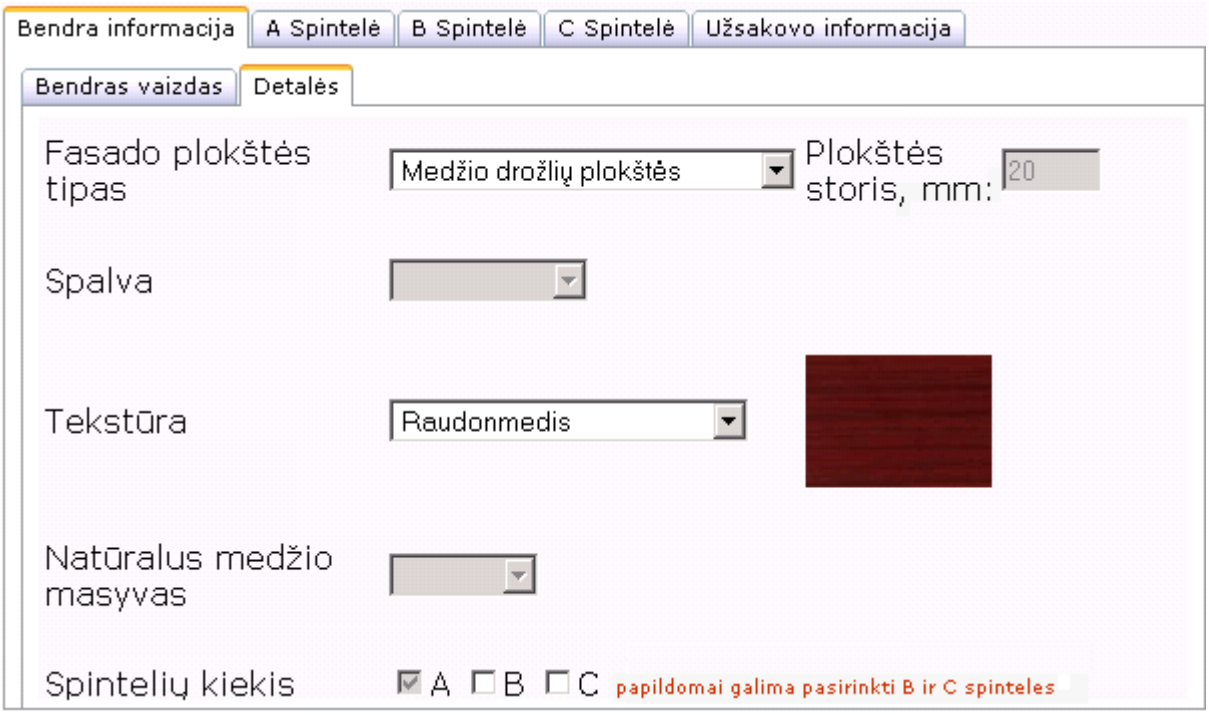
Aktoriai: Vartotojas/klientas

Aprašymas: Pirmame lange klientas spusteli nuorodą *Formuoti užsakymą*, atverčiamas langas *Virtuvės baldų konfigūravimo sistema* su daugiapakopiu *Ajax* kortelių konteineriu (žr. 16 paveikslą). Konteinerį sudaro aukštesnės pakopos kortelės: *Bendra informacija*, *A Spintelė*, *B Spintelė*, *C Spintelė*, *Užsakovo informacija*. Pagal nutylėjimą atverčiamas kortelės *Bendras vaizdas* langas. Šiame lange klientas mato, kurias virtuvės baldų komplekto dalis galima konfigūruoti ir užsakyti (spintelės pažymėtos raidėmis: „A“, „B“ ir „C“).



16 pav. Kortelės *Bendras vaizdas* langas

Kortelės *Detalės* (17 paveikslas) lange klientas turi detalizuoti spintelės fasado plokštės tipą. Galimos parinktys: laminuota medžio drožlių plokštė, natūralios medienos plokštė, medžio dulkių plokštė ir t.t.



17 pav. Kortelės *Detalės* langas

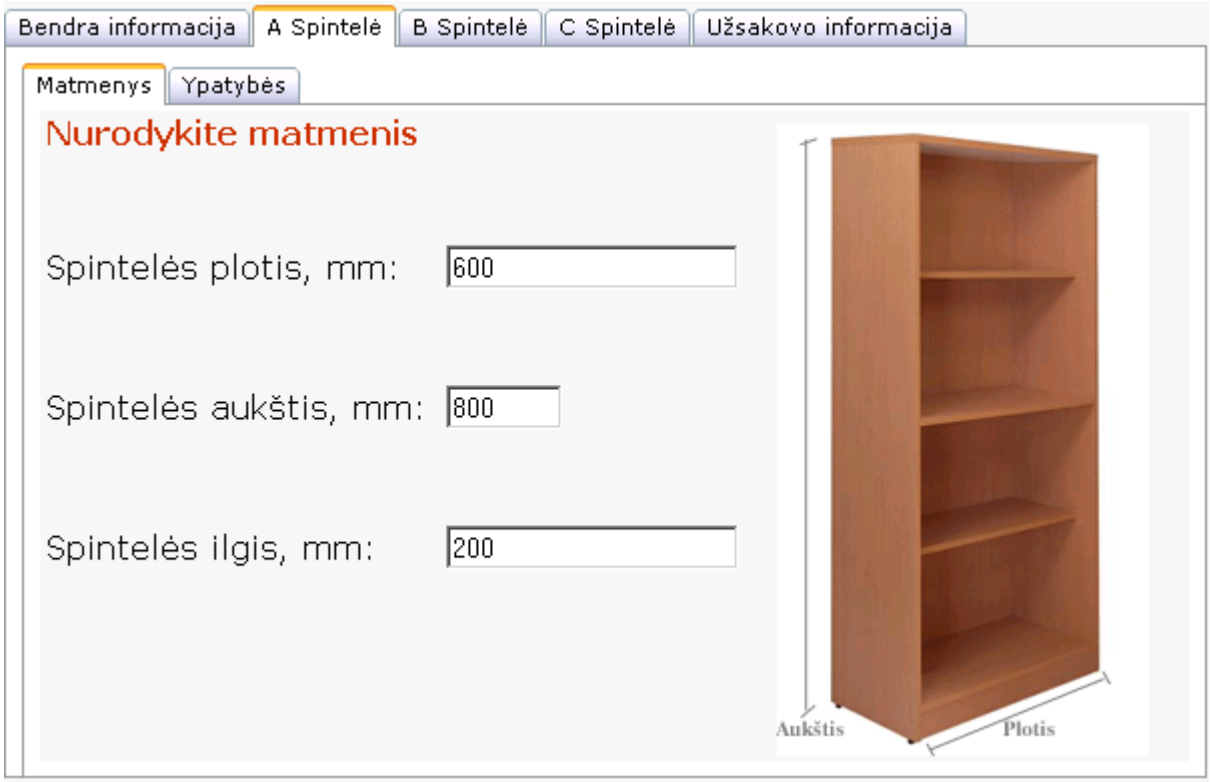
Spalva, *Tekstūra* ir *Natūralus medžio masyvas* laukeliai yra aktyvūs arba neaktyvūs, priklausomai nuo pasirinktos fasado medžiagos tipo. Pasirinkus fasado plokštės tipą *Medžio drožlių plokštė (nedažyta)* aktyvuojamas laukelis *Spalva*. Renkantis spalvas, šalia vaizduojamas spalvos pavyzdys (18 paveikslas).

18 pav. Spalvos pavyzdys

Pasirinkus fasado plokštės tipą *Natūralus medžio masyvas* aktyvuotame parinkties laukelyje *Natūralus medžio masyvas*, galima pasirinkti medienos tipą (ąžuolas, beržas).

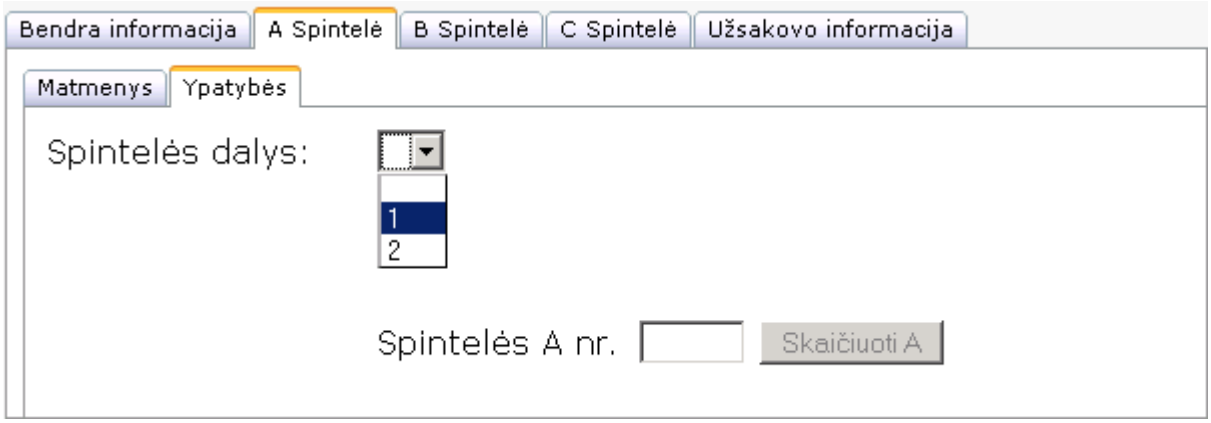
Plokštės storis įvedimo laukelis yra neaktyvus ir turi reikšmę 20 mm. Jis aktyvuojamas, kai pasirenkamas fasado plokštės tipas *Medžio dulkių plokštė*. Plokštės storį galima keisti (20 – 30 mm). Parinktis *Spintelių kiekis* leidžia papildomai pasirinkti „B“ ir „C“ spinteles.

Kortelės *Matmenys* lange (19 paveikslas) turi būti nurodomi spintelės matmenys: plotis, aukštis ir ilgis. Laukai turi įvesties kontrolę (neskaitinių simbolių įvesti neleidžiama).



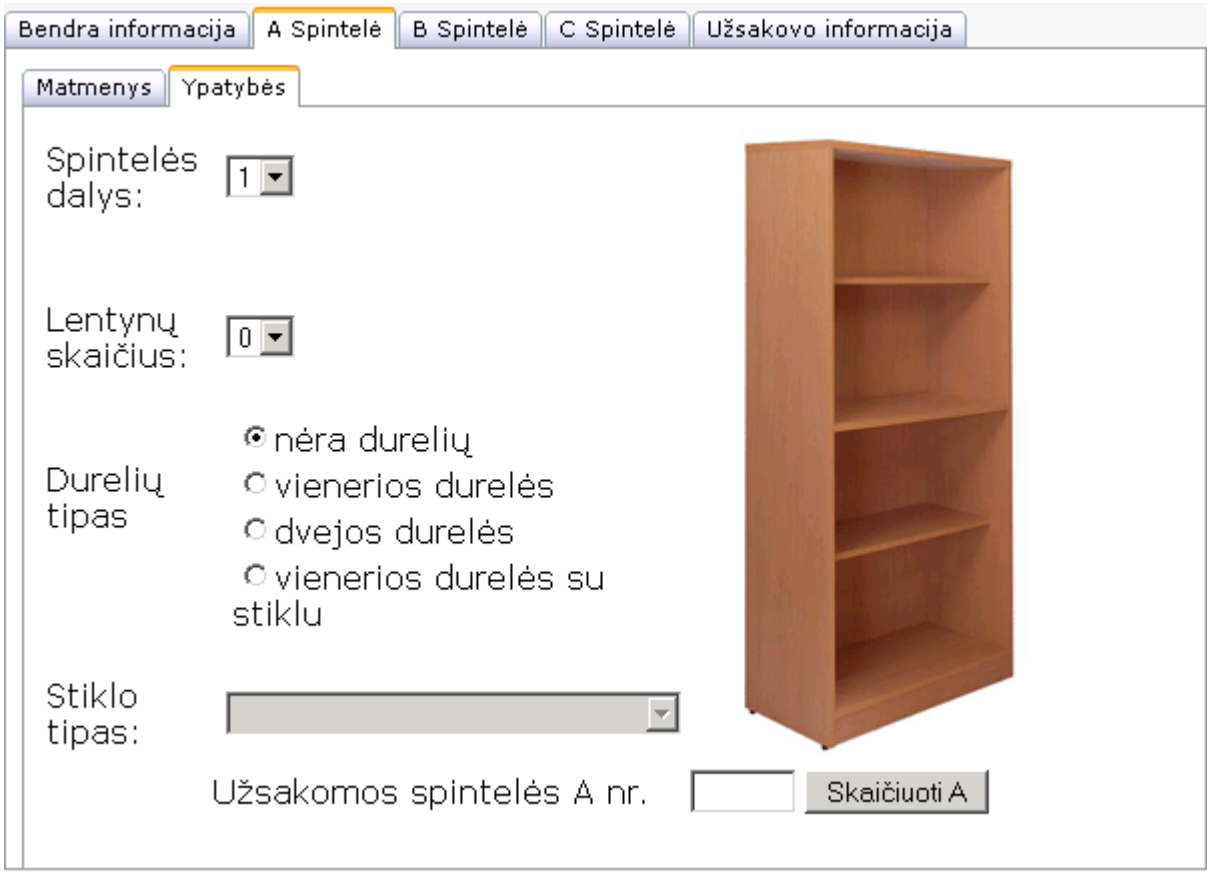
19 pav. Kortelės *Matmenys* grafinė sąsaja

Atvertus kortelės *Ypatybės* langą, pateikiamas parinkties laukelis *Spintelės dalys* (20 paveikslas). Jis skirtas spintelės dalių kiekiui nurodyti. Galimos parinktys: dviejų dalių spintelė arba vienos dalies spintelė.



20 pav. Spintelės dalys

Vienos dalies spintelė. Nurodžius, kad spintelė bus sudaryta iš vienos dalies, sistema leidžia pasirinkti lentynų skaičių, durelių tipą ir stiklo tipą (21 paveikslas).

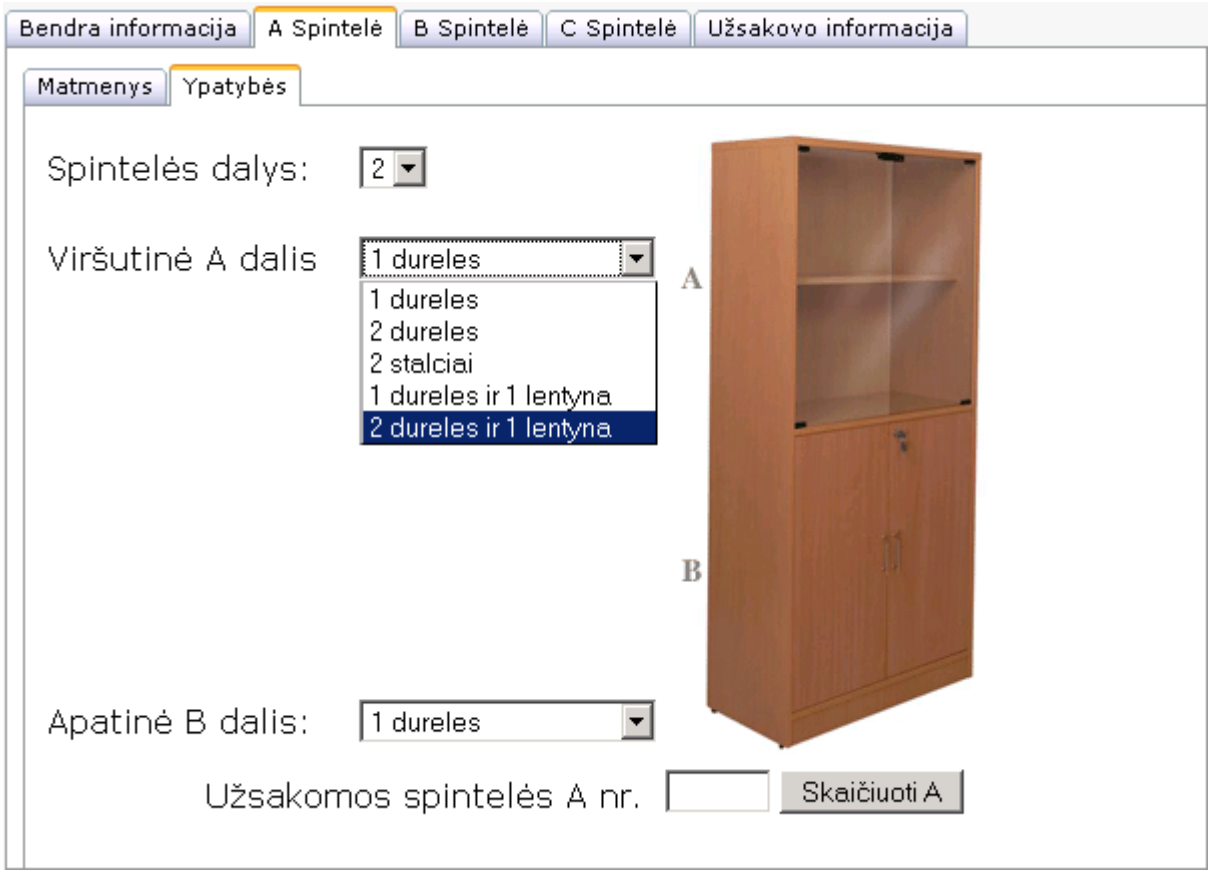


21 pav. Iš vienos dalies sudarytos spintelės konfigūravimas

Galimas lentynų skaičius nuo 0 iki 4. Pasirinkus durelių tipą *vienerios durelės su stiklu*, aktyvuojama parinktis *Stiklo tipas* (galimos parinktys: *skaidrus matinis*, *raštuotas skaidrus - konfeta*, *galaxy*, *flute*, *aero*, *delta* ir t.t.).

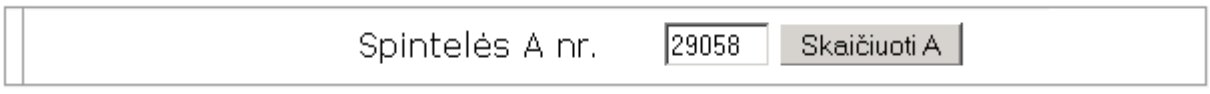
Dviejų dalių spintelė. Jei pasirenkama, kad spintelė bus sudaryta iš dviejų dalių, sistema siūlo pasirinkti viršutinės ir apatinės dalies sudėtį:

- § vienerios durelės;
- § dvejos durelės;
- § du stalčiai;
- § vienerios durelės ir lentyna;
- § dvejos durelės ir lentyna.



22 pav. Iš dviejų dalių sudarytos spintelės konfigūravimas

Baigus spintelių detalizavimą, aktyvuojamas mygtukas *Skaičiuoti A*. Jis skirtas užsakomos spintelės identifikacijos numeriui sugeneruoti.



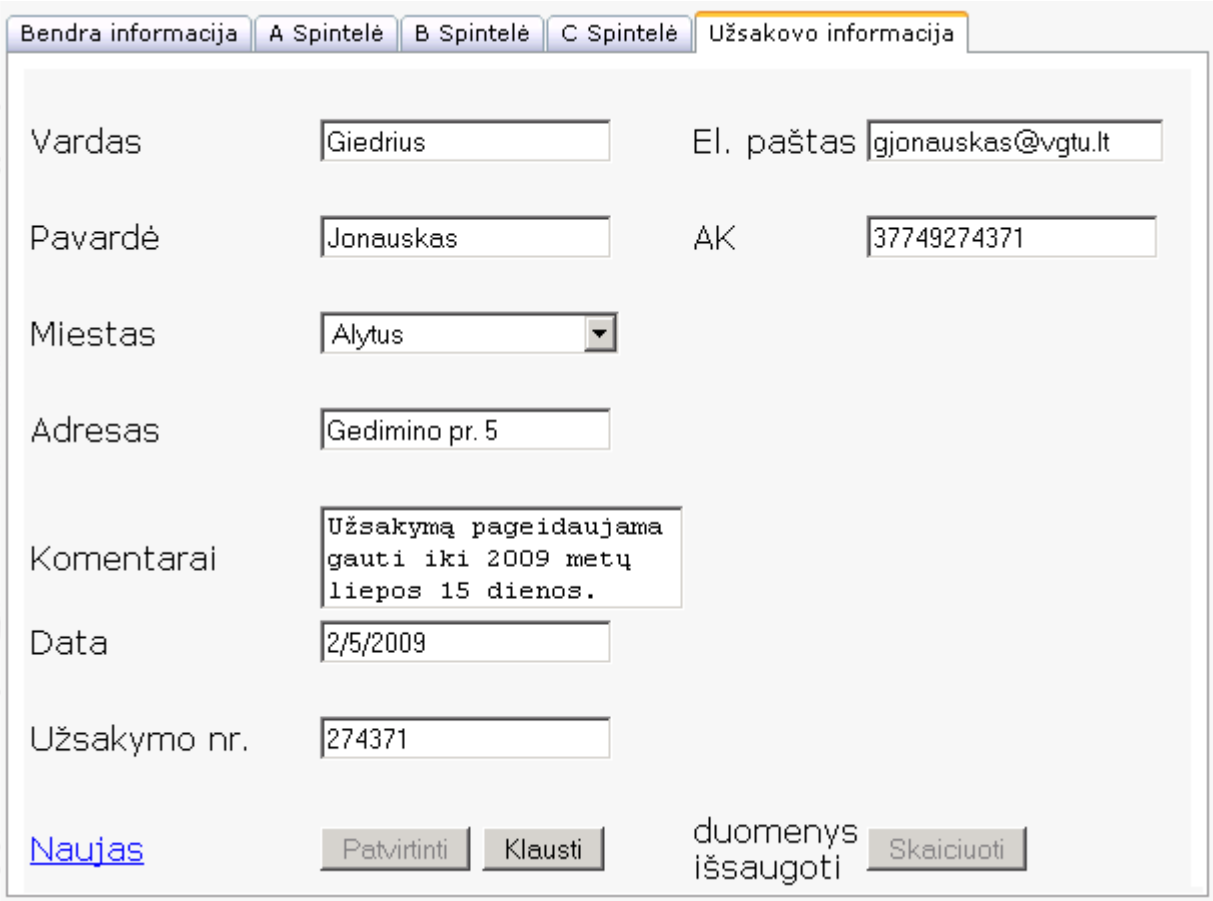
23 pav. *Spintelės A* identifikacijos numeris

Panaudos atvejis nr.: UC103

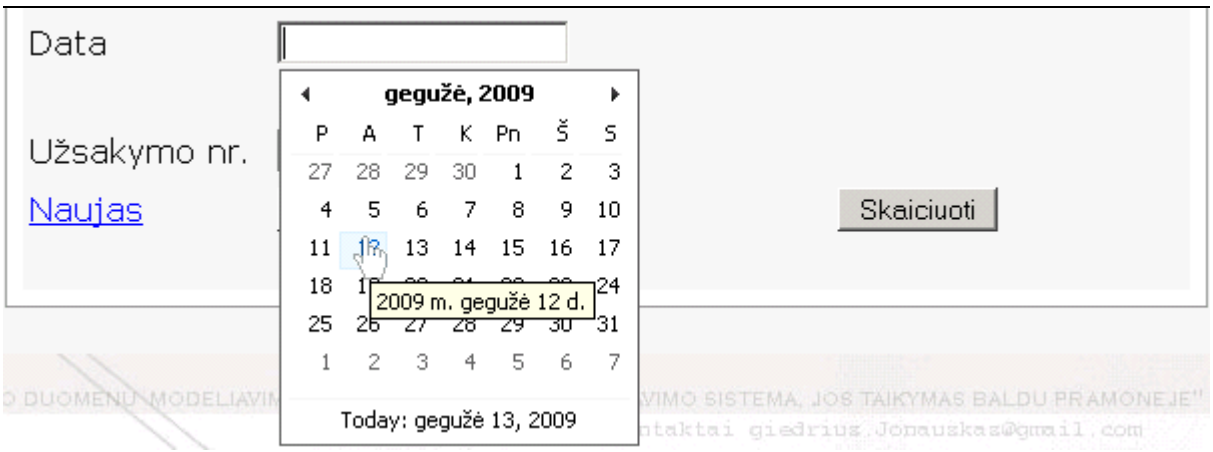
Panaudos atvejis: *Kliento asmeninės informacijos pateikimas*

Aktoriai: Vartotojas/klientas

Aprašymas: Kortelės *Užsakovo informacija* lange (24 paveikslas) klientas pateikia asmeninę informaciją, reikalingą užsakymo įforminimui. Užsakymo pateikimo data gali būti įvedama tiesiai į laukelį arba pasirenkama iš kalendoriaus (25 paveikslas). Į *AK* laukelį sistema leidžia įvesti tik skaitmenis. Užsakymo numeris sugeneruojamas paspaudus mygtuką *Skaičiuoti*. Paspaudus mygtuką *Patvirtinti*, sistema nusiunčia duomenis į duomenų bazę, suformuoja ir nusiunčia elektroninį pranešimą. Apie sėkmingą užsakymo pateikimą klientas informuojamas pranešimu *duomenys išsaugoti*.

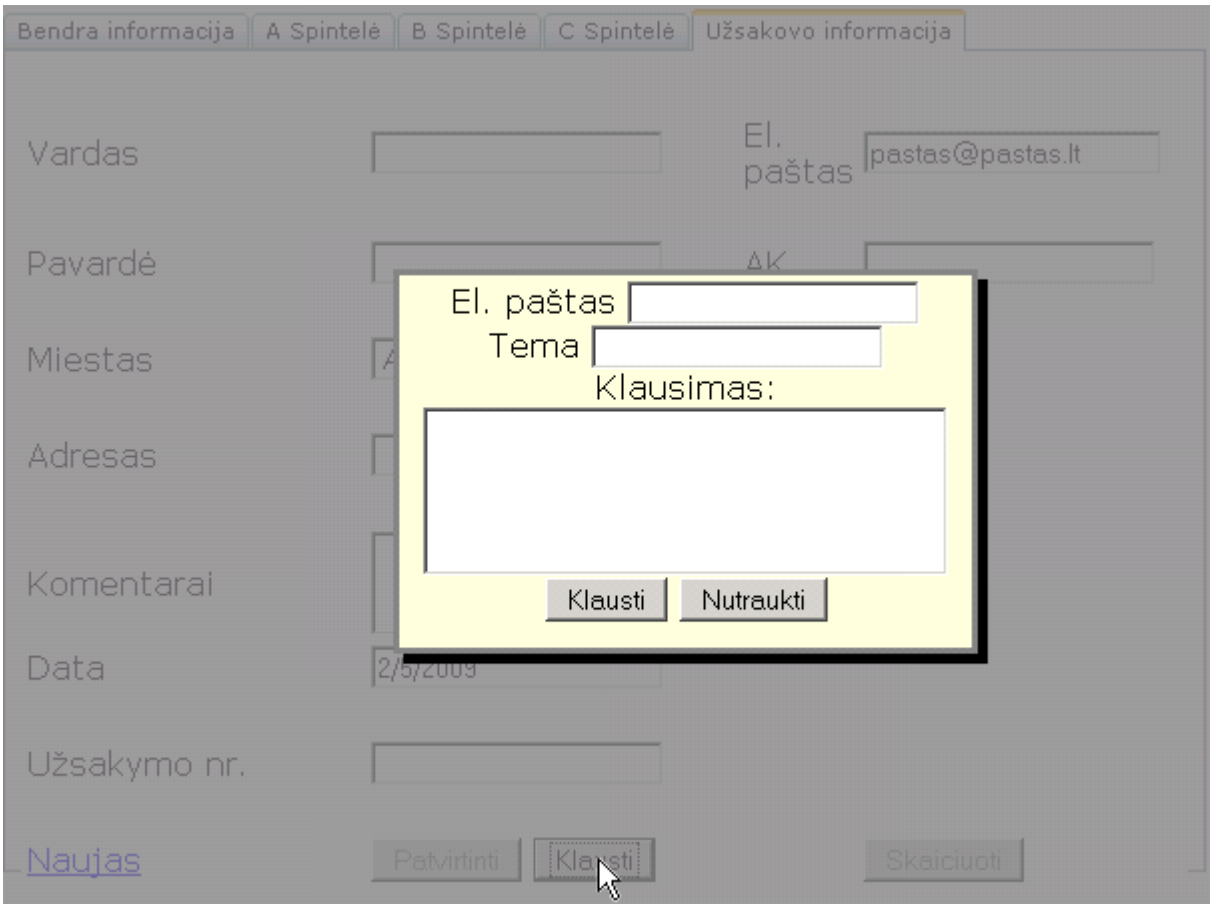


24 pav. Kortelės *Užsakovo informacija* langas



25 pav. Kalendorius

Jei klientui iškyla neaiškumų, jis gali nusiųsti paklausimą tiesiai iš tinklalapio, spustelėjęs mygtuką *Klausti* (26 paveikslas).

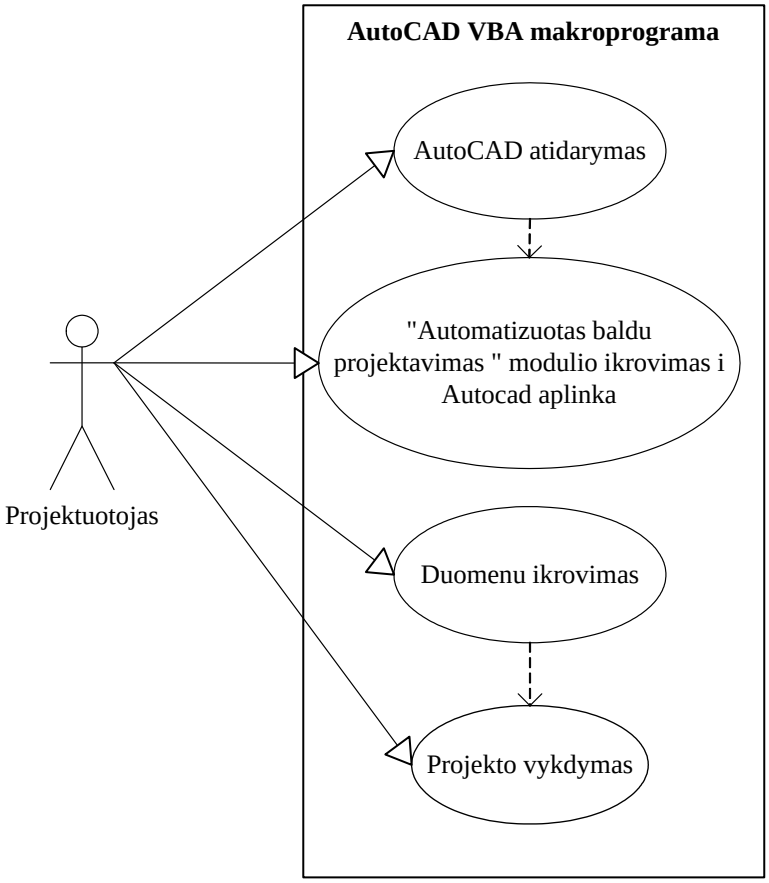


26 pav. Klausimo siuntimo grafinė sąsaja

4.3.2 *AutoCAD VBA* makroprogramos panaudos atvejai ir specifikacijos

Makroprogramą *automatizuotas baldų projektavimas* apibūdina diagrama, kurią sudaro keturi panaudos atvejai:

- *AutoCAD* atidarymas;
- makroprogramos įkrovimas į *AutoCAD* aplinką;
- duomenų įkrovimas;
- projekto vykdymas.



27 pav. Makroprogramos *Automatizuotas baldų projektavimas* panaudos atvejų diagrama

AutoCAD atidarymas ir *Automatizuoto baldų projektavimo makroprogramos įkrovimo į AutoCAD aplinką* panaudos atvejai diagramoje pateikti kaip neatsiejama bendro proceso dalis, tačiau darbe jie nedetalizuojami.

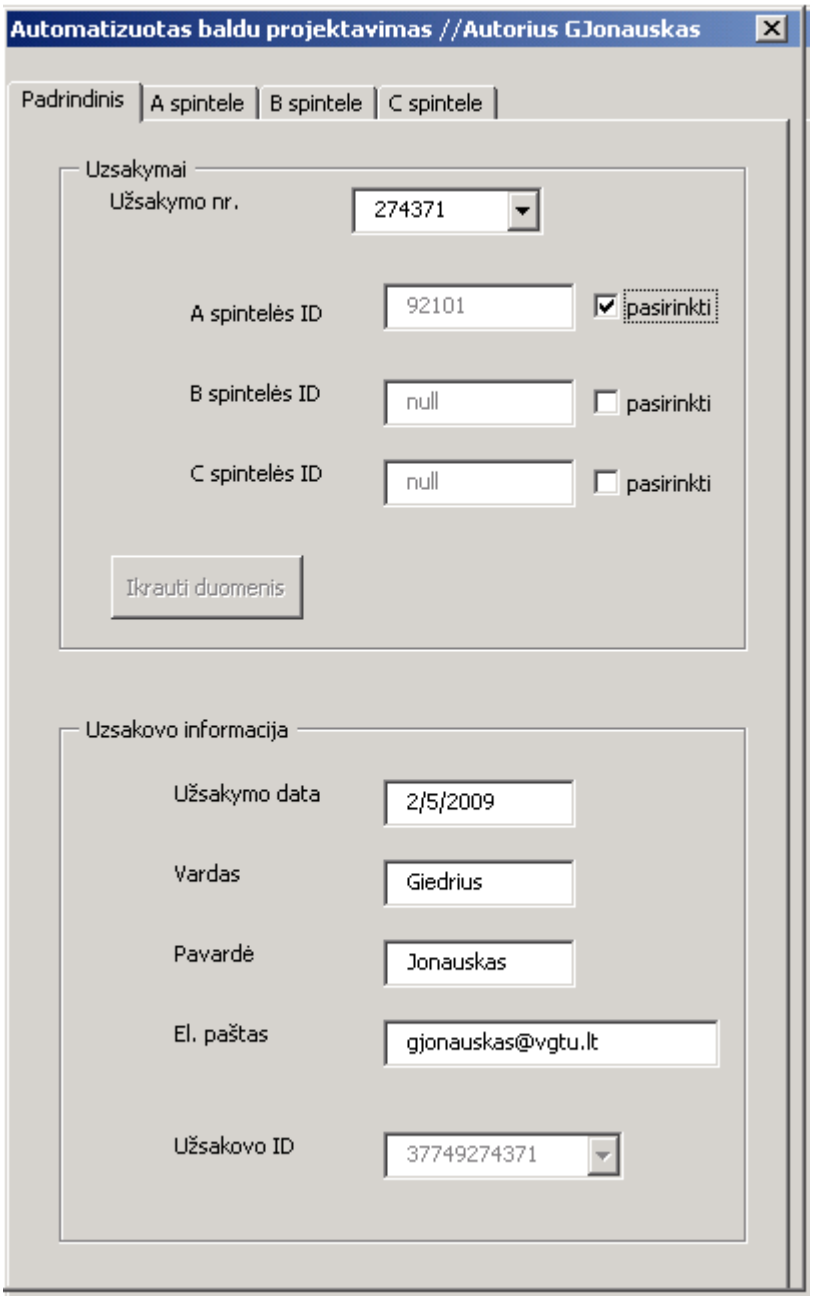
Panaudos atvejis nr.: UC201

Panaudos atvejis: *Duomenų įkrovimas į VBA makroprogramą*

Aktoriai: Projektuotojas

Aprašymas: Projektuotojas įkrautą *VBA* makroprogramą inicijuoja vykdymui. Atidaryta programa pagal nutylėjimą atverčia kortelės *Pagrindinis* langą (28 paveikslas). Duomenų įkrovimas atliekamas paspaudus

mygtuką *Įkrauti duomenis*. Po įkrovimo išsiskleidžiamame meniu *Užsakymo nr.* atsiranda įrašai iš duomenų bazės.



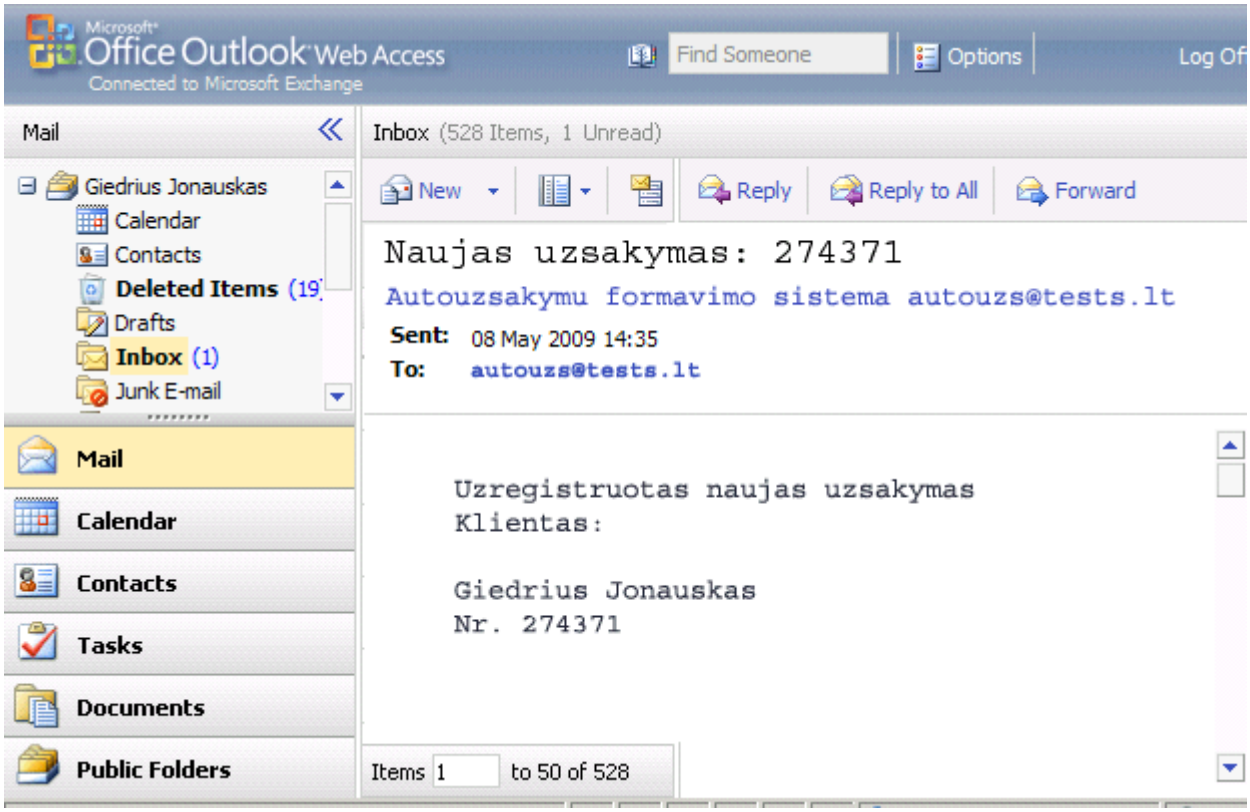
28 pav. Kortelės *Pagrindinis* langas

Panaudos atvejis nr.: UC202

Panaudos atvejis: *Užsakymo įvykdymas*

Aktoriai: Projektuotojas

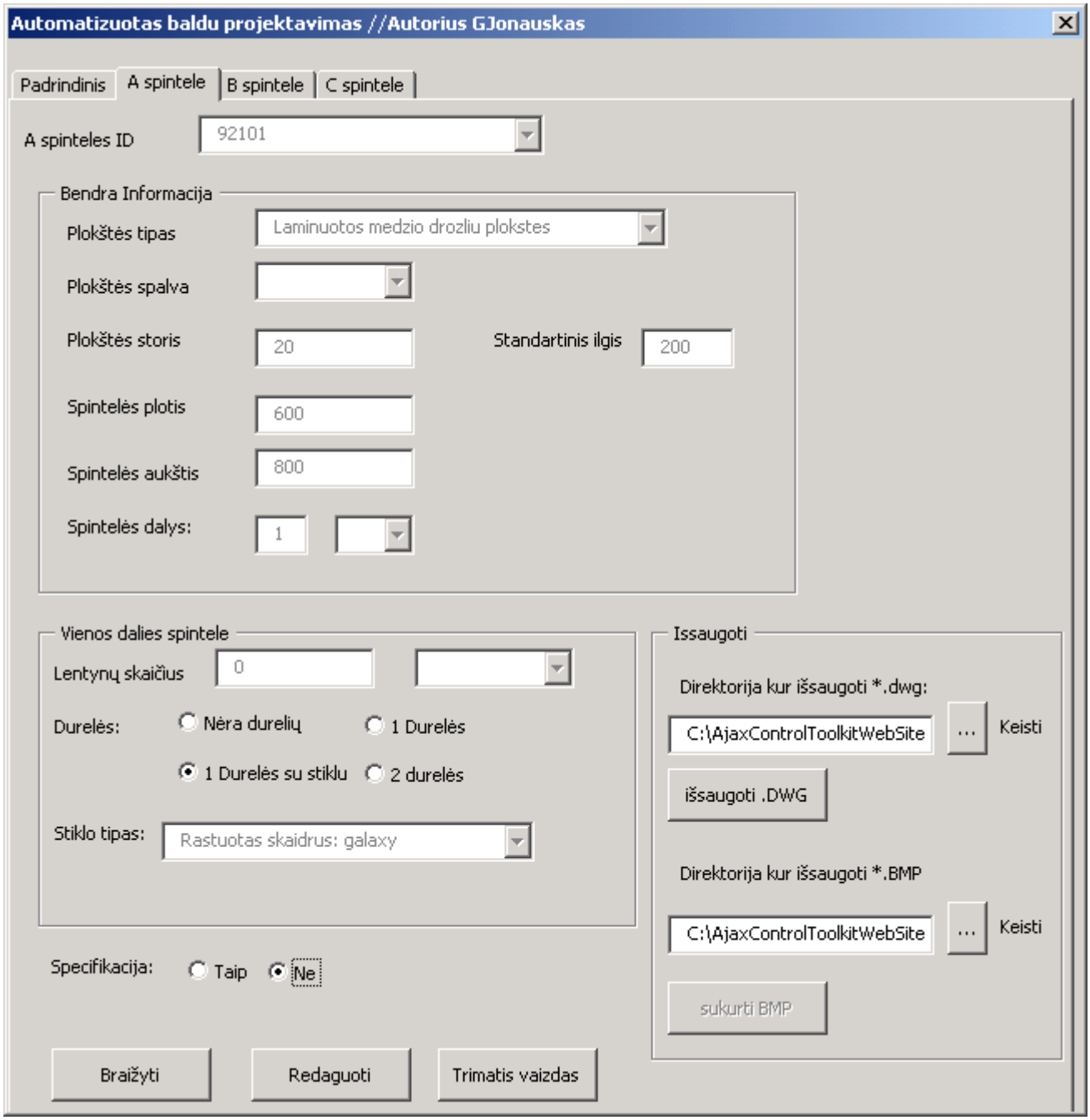
Aprašymas: Apie naują užsakymą projektuotojas informuojamas elektroniniu pranešimu (29 paveikslas). Elektroninį pranešimą suformuoja ir išsiunčia tinklalapyje veikianti pranešimų siuntimo funkcija. Pranešime pateiktas užsakymo numeris, kliento vardas ir pavardė.



29 pav. Elektroninis pranešimas

Iš sąrašo *Užsakymo nr.* pasirenkamas užsakymo numeris. Pagal tai, koks numeris pasirinktas, programa atvaizduoja užsakymo detales: datą, kliento vardą, pavardę, adresą ir kt. (28 paveikslas).

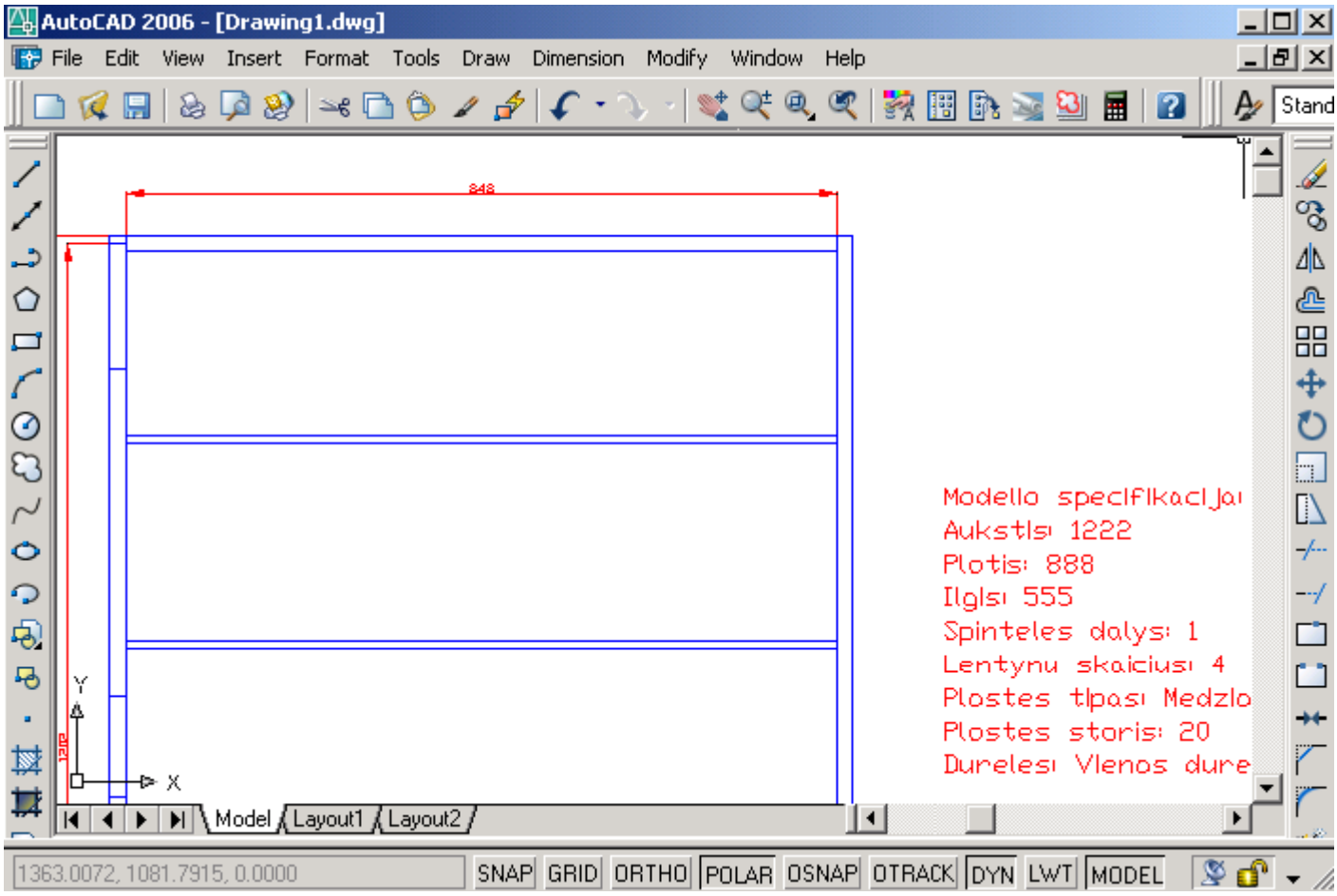
Kortelės *A spintelė* lange (30 paveikslas) atvaizduojama informacija, susijusi su „A“ spintele: plokštės tipas, storis, spintelės plotis ir aukštis, spintelės dalys ir t.t. Įvesties laukai neaktyvūs. Jei projekto vykdytojas mato, kad užsakymas turi neatitikimų (pavyzdžiui, blogai kliento parinkti matmenys) arba sandėlyje nėra medžiagų užsakymui įgyvendinti, jis gali duomenis pakoreguoti, spustelėdamas mygtuką *Redaguoti*. Įvesties laukai tampa aktyvūs.



30 pav. Makroprogramos kortelės *A spintelė* langas

Prieš atliekant braižymą galima parinkti, ar modelis turės specifikaciją. Pasirinkus *Taip* ir spustelėjus mygtuką *Braižyti*, *AutoCAD* lange nubraižomas dvimatis baldo modelis (31 paveikslas). Raudona spalva pateikiami modelio matmenys ir specifikacija, mėlyna – pats modelis.

Sekcijoje *Išsaugoti* galima pasirinkti, kur išsaugoti suprojektuotą modelį: nurodžius išsaugojimo kelią ir spustelėjus *išsaugoti .DWG* arba *sukurti BMP*.

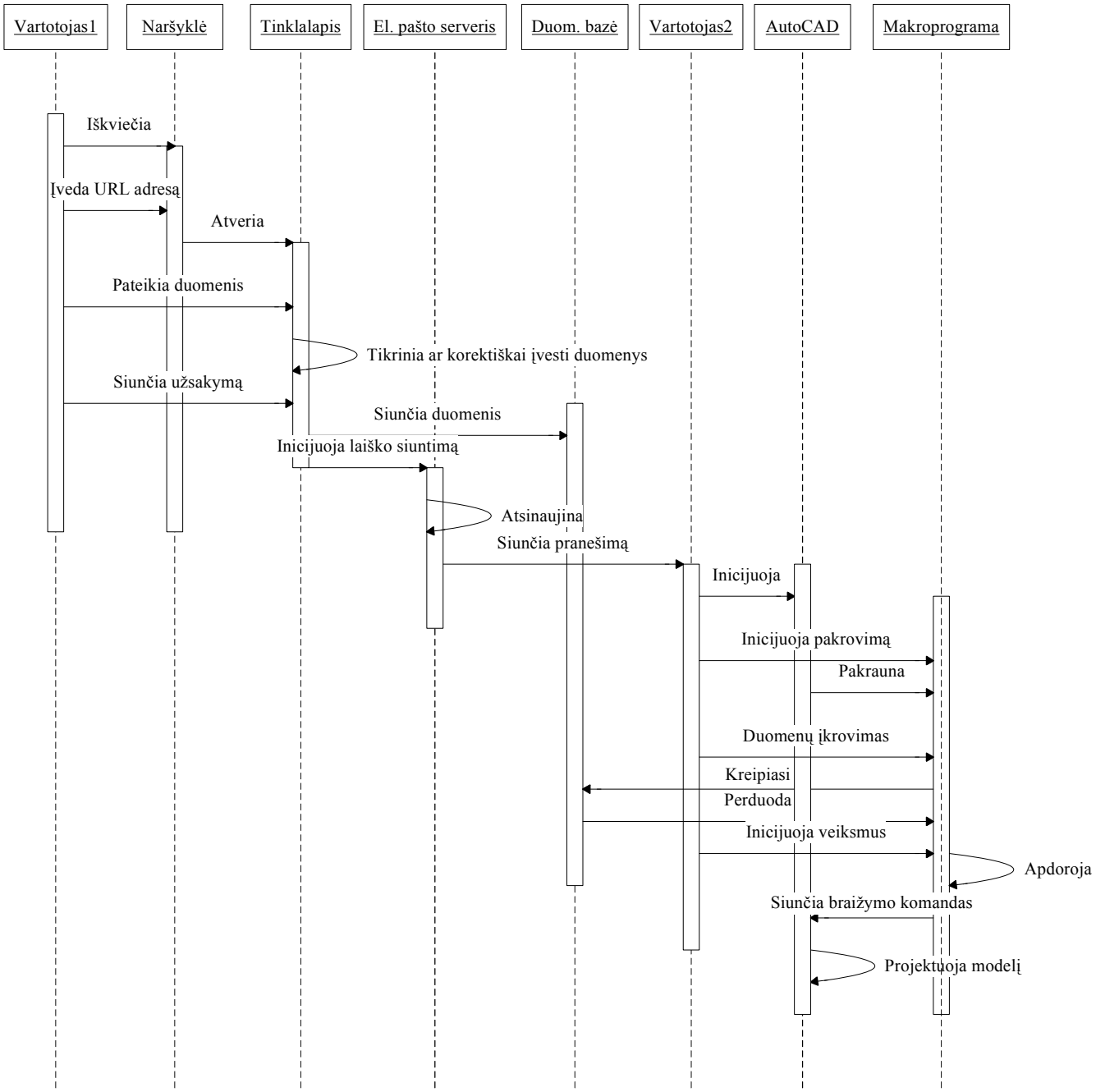


31 pav. Dvimatis modelio vaizdas

32 pav. Trimatis modelio vaizdas

4.3.3 Sistemos naudojimo sekų diagrama

Ši diagrama iliustruoja vartotojų veiksmų, sistemos būsenų ir procesų išsidėstymą laiko atžvilgiu.



33 pav. Sistemos naudojimo sekų diagrama

4.3.4 Duomenų bazės loginė schema

Duomenų saugojimui sukuriama *Microsoft Access* reliacinė duomenų bazė *db.mdb*. Ši duomenų bazė skirta informacijai tinklalapyje atvaizduoti (miestų pavadinimai, fasado plokščių tipai, spalvų kodai) ir iš tinklalapio gautiems užsakymams išsaugoti.

Duomenų bazė sudaryta iš 8 lentelių:

§ *Uzsakymai* – pagrindinė lentelė, kurioje saugomi įrašai, gauti klientui suformavus užsakymą;

§ *UzsAspint* – informacija apie „A“ spintelę;

§ *UzsBspint* – informacija apie „B“ spintelę;

§ *UzsCspint* – informacija apie „C“ spintelę;

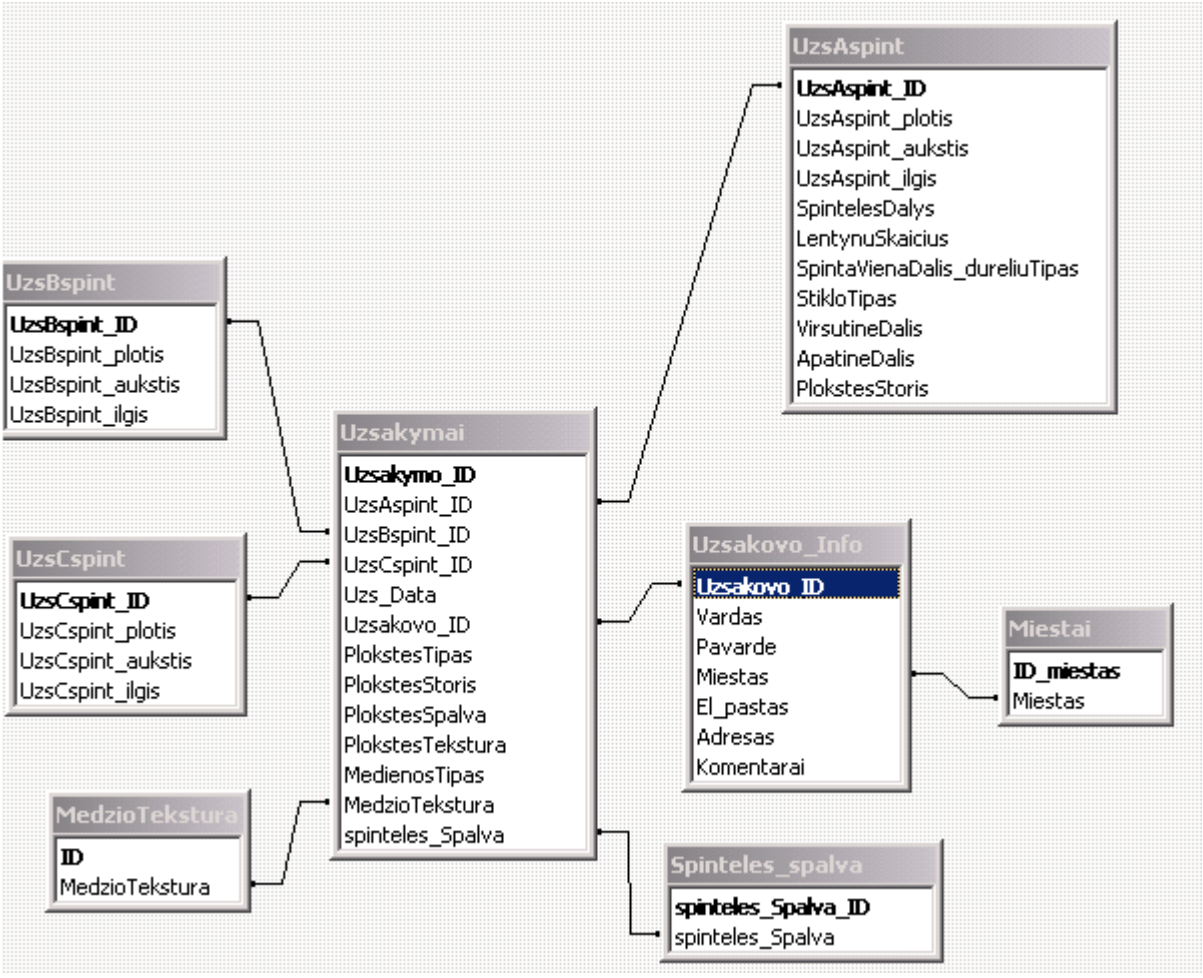
§ *Uzsakovo_Info* – šios lentelės įrašai yra kliento duomenys, gauti užsakymo formavimo metu;

§ *Miestai* – Lietuvos miestai;

§ *Spinteles_spalva* – spalvų kodai;

§ *MedzioTekstura* – medžio tekstūrų pavadinimai.

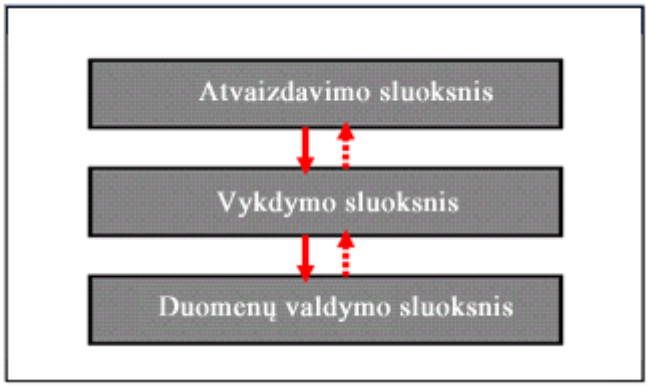
Lentelės, jų sąsajos ir pirminiai raktai pateikiami 34 paveiksle.



34 pav. Duomenų bazės loginė schema

4.4. Žiniatinklio taikomoji programa *baldų e.užsakymas*

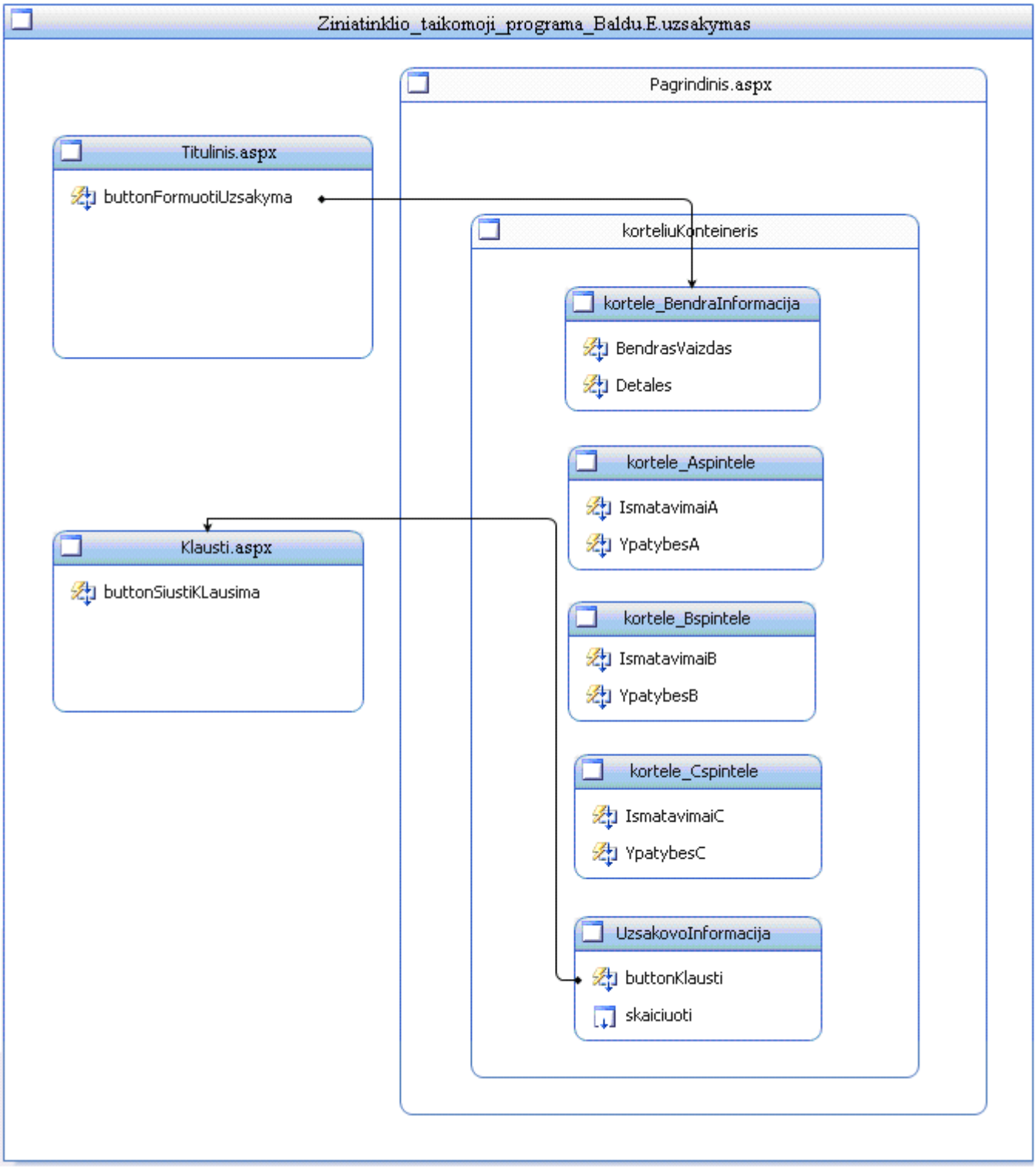
Žiniatinklio taikomoji programa *baldų e.užsakymas* kuriama remiantis trijų sluoksnių kliento-serverio architektūra (35 paveikslas).



35 pav. Trijų sluoksnių kliento-serverio architektūra [3]

- Atvaizdavimo sluoksnyje realizuojama grafinė vartotojo sąsaja, skirta baldų konfigūravimo parinkčių atvaizdavimui ir vartotojo duomenų įvedimui. Vartotojo sąsaja realizuojama puslapiuose: *Titulinis.aspx*, *Pagrindinis.aspx*, *Klausti.aspx*. Ji sudaryta iš statinių ir dinaminių elementų, išdėstytų tam tikrose srityse. Statiniai elementai yra kortelės *Užsakovo informacija* lange esantys kliento asmeninės informacijos pateikimo laukeliai. Dinaminių elementų atsiradimą įtakoja vartotojas, jo atliekami veiksmai (pavyzdžiui, pasirinkus vienos dalies spintelę, sistema pateikia šiai spintelei būdingus konfigūravimo atributus).
- Vykdymo sluoksnis yra skirtas užsakymo formavimo ir pateikimo funkcionalumui. Šiame sluoksnyje realizuojama baldų konfigūravimo ir informacijos apdorojimo logika, jame integruojami duomenys iš duomenų bazės, vykdomi serverio srities *servletai* ir verslo taisyklės.
- Duomenų valdymo sluoksnis yra skirtas sistemos duomenų bazės valdymui [3].

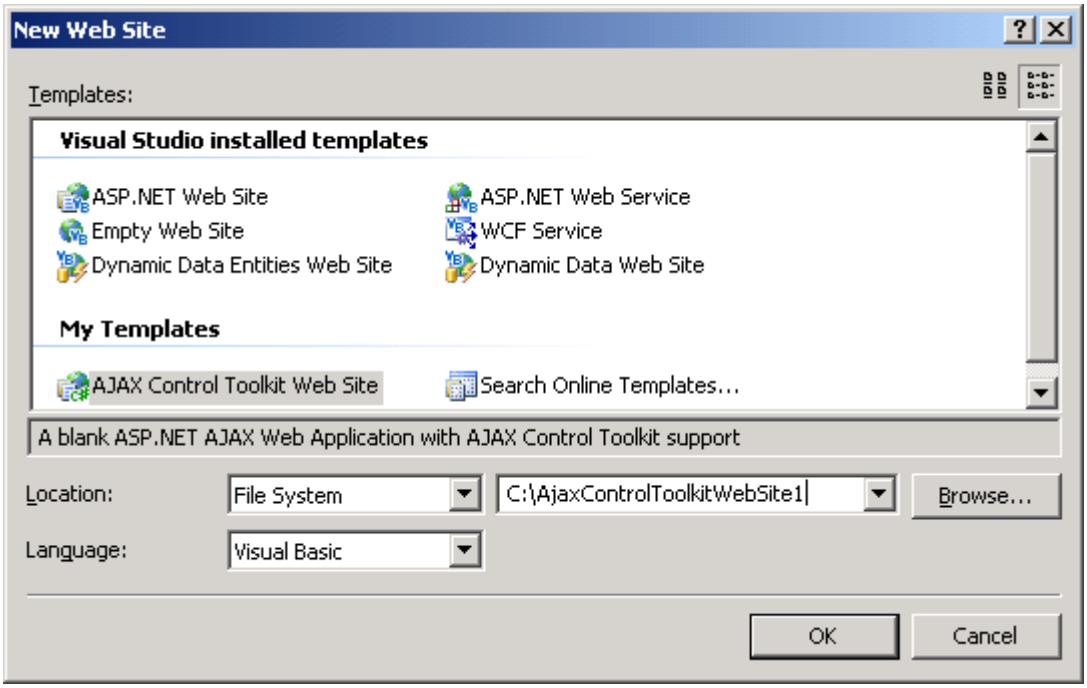
Atvaizdavimo ir vykdymo sluoksnių kūrimui naudojama *Microsoft Ajax ASP.NET* platforma. *Ajax ASP.NET* technologiniu požiūriu yra ta pati *ASP.NET* platforma, papildyta nemokamomis *Ajax* bibliotekomis [29].



36 pav. Tinklalapio srautų diagrama

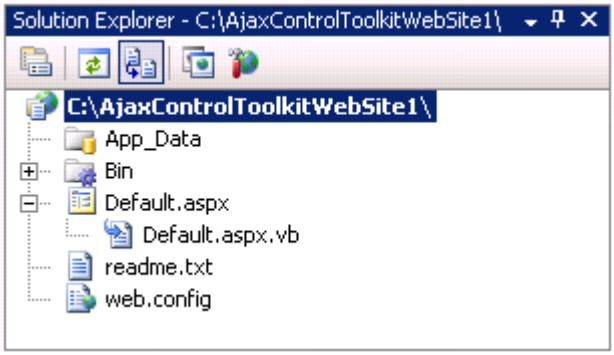
4.4.1 Tinklalapio konstravimas

Visual WEB Developer 2008 programavimo aplinkoje pasirenkama *File* → *New Web Site*. Programavimo kalba – *Visual Basic*, šablonas – *AJAX Control Toolkit Web Site*. Nurodomas projekto pavadinimas *AjaxControlToolkitWebSite1* (37 paveikslas).



37 pav. *Ajax* tinklalapio kūrimas

Ajax projekte automatiškai sugeneruojami konfigūracijos, kontrolių, kodo failai ir *App_Data* duomenų šaltinio aplankas (38 paveikslas).



38 pav. Naujas *Ajax* projektas

Pagal nutylėjimą *Default.aspx* faile pateikiamos standartinės *html* puslapio žymos. Faile *Default.aspx.vb* sugeneruojama tuščia *Default* klasė. Pervadinus *Default.aspx* ir *Default.aspx.vb* failus į *Titulinis.aspx* ir *Titulinis.aspx.vb*, papildžius programiniu kodu ir tituliniu fonu, gaunamas pirmas reprezentacinis-titulinis puslapis. Iš šio puslapio vartotojas patenka į *Pagrindinis.aspx* puslapį.

Sukuriamas *pagrindinis.aspx* tinklalapio failas. Jame įdiegtas *Ajax* kortelių konteineris suteikia puslapiui dinamikos ir interaktyvumo. Pasinaudojus *Visual Web Developer 2008* programavimo aplinkos teikiamomis galimybėmis, faile *Pagrindinis.aspx* „paimk ir padėk“ principu suformuojama pradinė baldų komplektavimo, konfigūravimo ir e.užsakymo formavimo grafinė vartotojo sąsaja. Įdedamos lentelės, išdėstomi mygtukai, laukeliai, sąrašo meniu, *radio* mygtukai, *checkbox* parinktys, nuorodos ir k.t.

Sistemos logika realizuojama *Pagrindinis.aspx.vb* faile. Klasėje *Pagrindinis* aprašomos puslapio funkcijos ir įvykių metodai. Iš viso buvo realizuota 19 metodų ir 4 funkcijos. Pateikiami keli jų pavyzdžiai:

§ *Protected Sub insert()* – metodas, skirtas duomenų įterpimui į duomenų bazę.

Darbui su duomenimis naudojamas *OleDbConnection* objektas.

Tiekėjas: *Microsoft.Jet.OLEDB.4.0*

Duomenų šaltinis: *db.mdb*

Naudojamos papildomos bibliotekos:

- *System.Data*
- *System.Data.OleDb*

§ *Public Function SendEmail()* – funkcija, skirta elektroninių pranešimų siuntimui.

Naudojamas pašto siuntimo serveris*: smtp.gmail.com*

Papildomos bibliotekos:

- *System.Net.Mail*
- *System.Net*

§ *Protected Sub TipasDropDownList1_SelectedIndexChanged()* metodas reaguoja į vartotojo veiksmus, kai keičiamas fasado plokštės tipas (priklausomai nuo pasirinkto tipo aktyvuojami arba deaktyvuojami tam tikri puslapio elementai).

§ *Protected Sub SpalvaDropDownList_SelectedIndexChanged()* metodas priklausomai nuo pasirinkto spalvos kodo pateikia spalvos paveikslėlį *.gif* formatu.

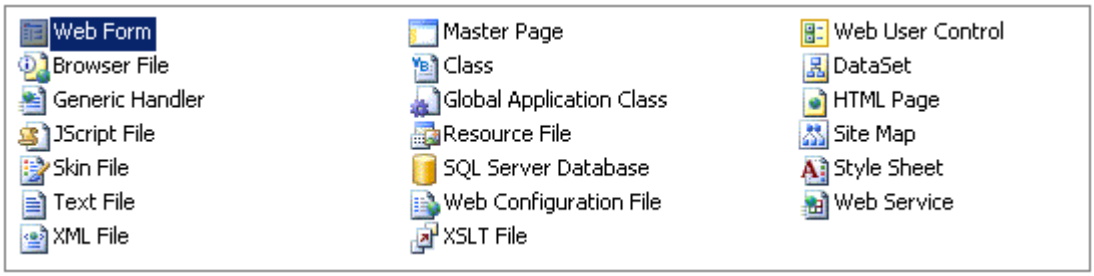
§ *Protected Sub SkaiciuotiButton_Click ()* metodas yra skirtas užsakymo numerio generavimui.

Į projektą įtraukiami papildomi komponentai iš komponentų sąrašo (39 paveikslas).

§ *db.mdb* failas yra įkeliamas į aplanką *App_Data*;

§ *funkcijos.js* – realizuojamos kliento srityje veikiančios *JavaScript* funkcijos;

§ *StyleSheet.css* failas yra skirtas puslapio stiliaus išlaikymui (*StyleSheet.css* apibrėžia skirtingus šriftus, paraštes, tarpus tarp eilučių ir pan.) (40 paveikslas).

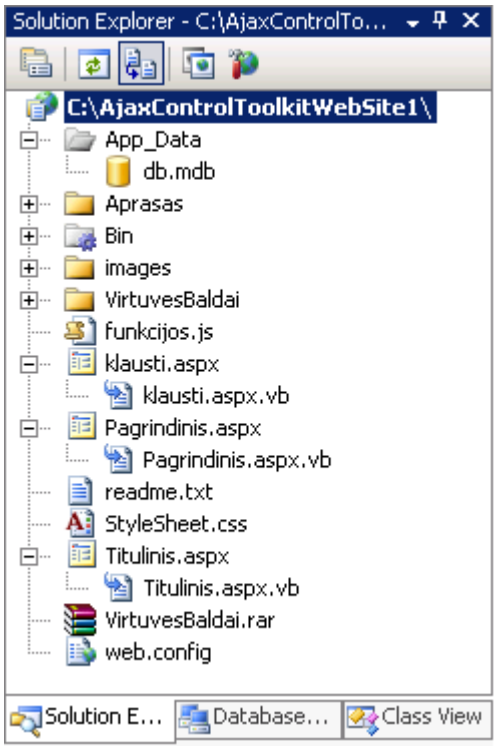


39 pav. Komponentų sąrašas

StyleSheet.css

```
1           .popupControl {
2              background-color:#AAD4FF;
3              position:absolute;
4              visibility:hidden;
5              border-style:solid;
6              border-color:Black;
7              border-width: 2px;
8           }
```

40 pav. *StyleSheet.css* kodo dalis

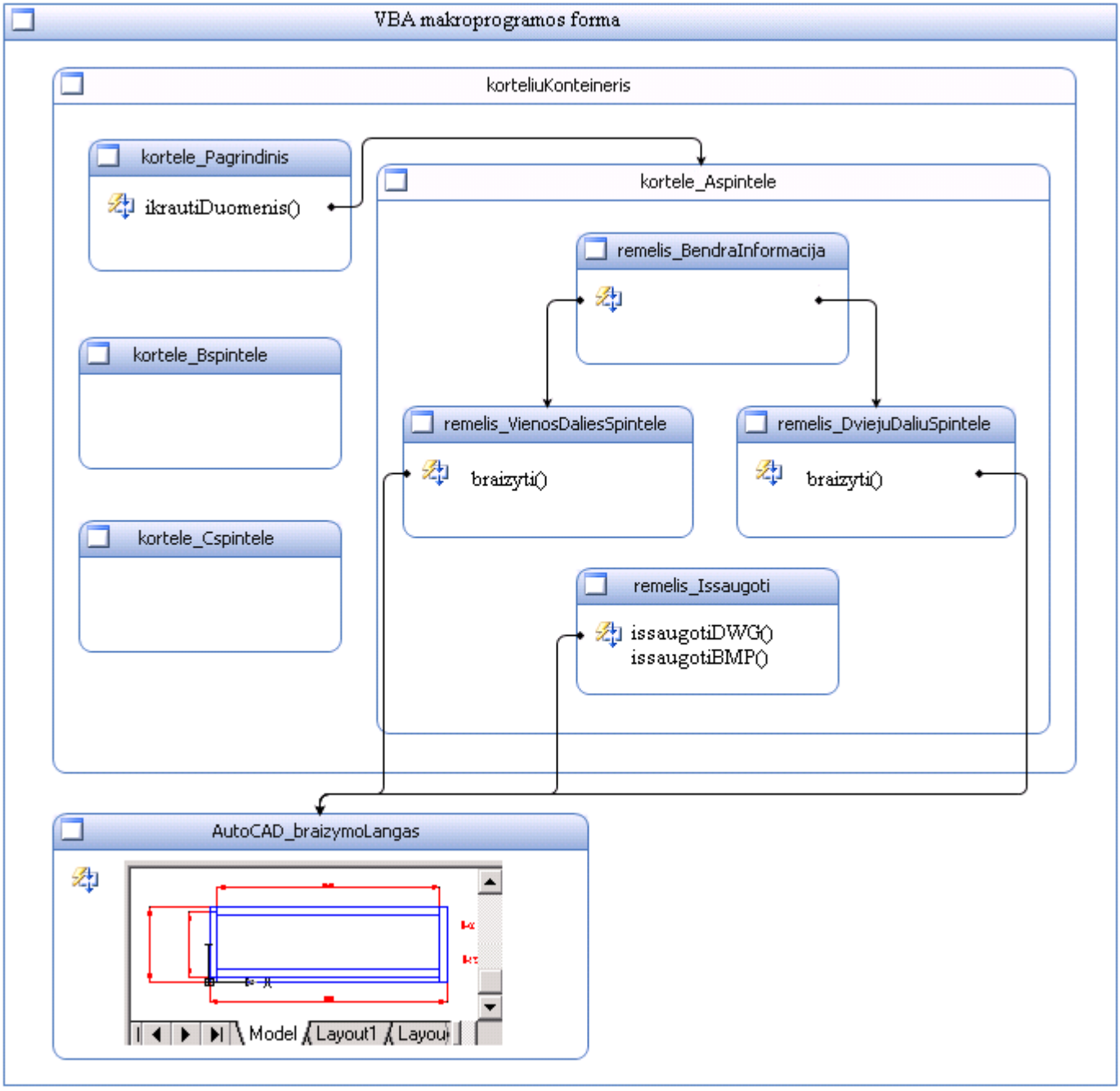


41 pav. Žiniatinklio taikomosios programos projekto sudėtis ir komponentų hierarchija

Tam, kad sukurtas tinklalapis būtų pasiekiamas visiems interneto vartotojams, jis turi būti patalpintas į žiniatinklio serverį. Yra du galimi tinklalapių talpinimo į serverius būdai: galima diegti į nuosavą žiniatinklio serverį arba į internetinių puslapių talpinimo paslaugas teikiančių įmonių serverius. Sukurtas tinklalapis patalpinamas į nuosavą žiniatinlio serverį. Šiam tikslui naudojama *Microsoft Windows XP Professional* operacinė sistema ir sukonfigūruotas žiniatinklio programų serveris *IIS 6.0*.

4.5. Automatizuoto baldų projektavimo *VBA* makroprograma

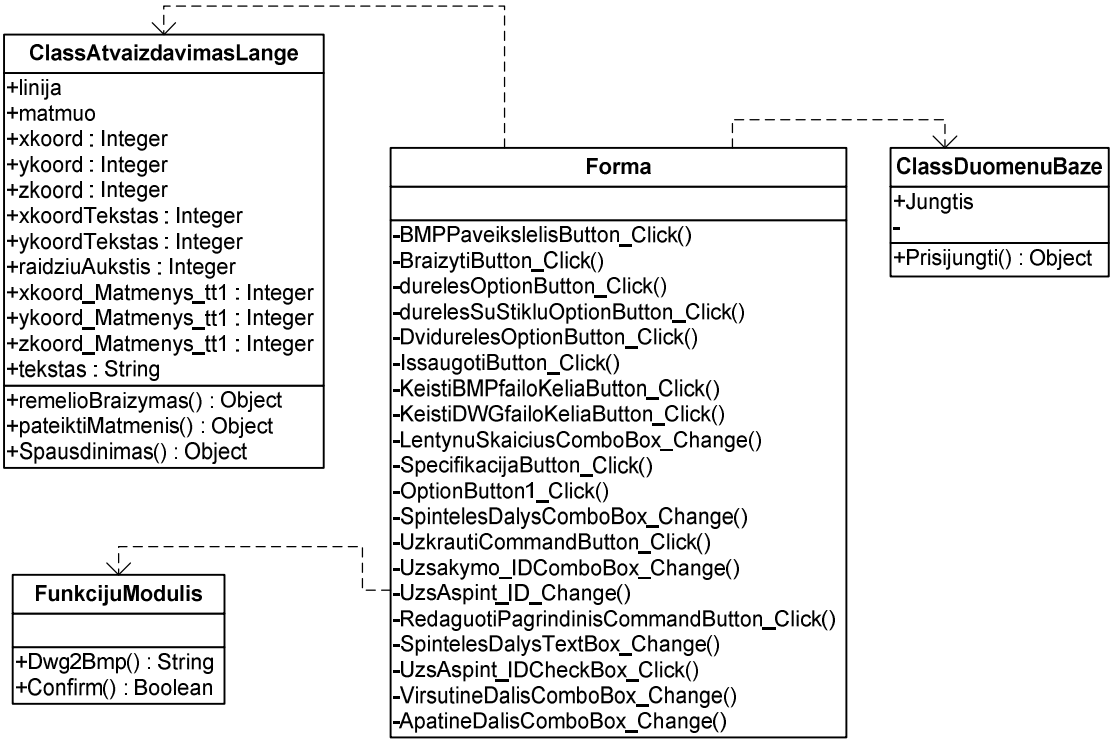
Automatizuoto baldų projektavimo VBA makroprogramos srautų diagrama pateikiama 42 paveiksle.



42 pav. *VBA* makroprogramos srautų diagrama

4.5.1 Klasių diagrama

VBA makroprogramos klasių diagrama pateikiama 43 paveiksle.



43 pav. *VBA* makroprogramos klasių diagrama

4.5.2 *VBA* makroprogramos konstravimas

AutoCAD aplinkoje sukuriamas *VBA* projektas *dubaze.dvb.* Projektą sudaro forma, kurioje realizuota makroprogramos grafinė vartotojo sąsaja (kortelių konteineris su kortelėmis: *Pagrindinis, Aspintelė, Bspintelė, Cspintelė,* mygtukai, įvedimo, parinkties laukeliai ir kt.). Braižymo, matmenų pateikimo, sąveikos su duomenų baze ir kiti metodai deklaruojami klasių moduliuose: *ClassAtvaizdavimasLange* ir *ClassDuomenuBaze.* Funkcijos *Dwg2Bmp* ir *Confirm* deklaruojamos modulyje *FunkcijuModulis*. Iš viso buvo realizuota 27 metodai ir 2 funkcijos. Pateikiami keli jų pavyzdžiai:

§ *Public Sub Prisijungti()* metodas yra skirtas *AutoCAD* prisijungimui prie DB. Jis realizuotas modulyje *ClassDuomenuBaze.*

Darbui su duomenimis naudojamas *OleDbConnection* objektas.

Tiekėjas: *Microsoft.Jet.OLEDB.4.0*

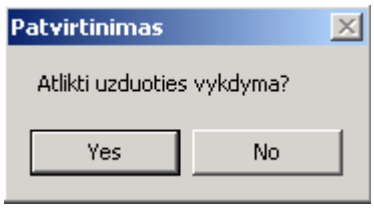
Duomenų šaltinis: *db.mdb*

§ *Private Sub KeistiDWGfailoKeliaButton_Click()* metodas yra skirtas rezultatų failo išsaugojimo kelio pakeitimui. Jis realizuotas modulyje *Forma*. Naudojami objektai: *Shell32.Shell* ir *Shell32.Folder2*.

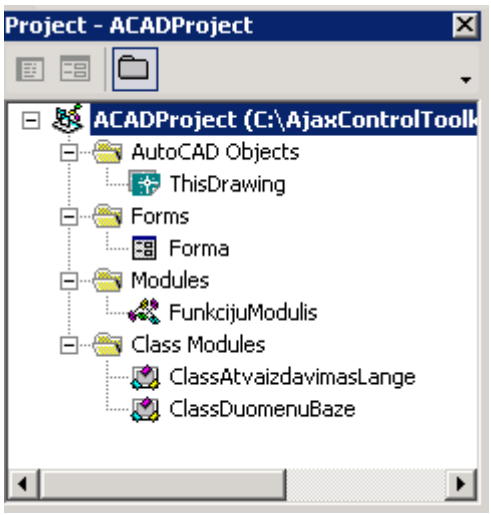
Papildomos bibliotekos:

Microsoft Shell Controls And Automation - Shell32.dll

§ *Function Confirm(MsgText As String) As Boolean* funkcija yra skirta patvirtinimo lentelės pranešimui išvesti (44 paveikslas).



44 pav. Patvirtinimo pranešimas



45 pav. Makroprogramos projekto failų hierarchinis išsidėstymas

4.6. Programinės įrangos kokybės užtikrinimas

Funkcinis testavimas

Funkcinio testavimo tikslas – patikrinti atskirų funkcinių sričių veikimą. Buvo patikrintos šios funkcinės sritys:

§ žiniatinklio taikomoji programa – duomenų įvedimas, redagavimas, e.užsakymo formavimas ir siuntimas, duomenų išsaugojimas į duomenų bazę, vartotojo sąsaja;

§ *VBA AutoCAD* makroprograma – makroprogramos įkrovimas į *AutoCAD* aplinką, duomenų įkrovimas, informacijos redagavimas, modelio projektavimas, vartotojo sąsaja.

Šios sritys testuojamos remiantis testavimo scenarijais (angl. *Test case* – TC):

Duomenų įvedimas. TC tikslas yra patikrinti, ar žiniatinklio taikomoji programa leidžia įvesti reikiamus duomenis, ar šie duomenys yra teisingai išsaugomi ir teisingai rodomi vartotojui. Taip pat tikrinama, kaip sistema elgiasi įvedus nekorektiškus duomenis.

E.užsakymo formavimas ir siuntimas. Šio TC tikslas yra patikrinti ar žiniatinklio taikomoji programa korektiškai suformuoja e.užsakymą ir išsiunčia jį gavėjui. Tikrinamas pranešimo siuntimo procesas.

VBA makroprogramos įkrovimas. Testuojama, ar *AutoCAD* leidžia įkrauti *VBA* makroprogramą, ar nėra nenumatytų trikdžių. Tikrinamas makroprogramos veikimas *AutoCAD* aplinkoje (atidaroma/ uždaroma forma, nutraukiamas vykdymas proceso metu ir t.t.).

Duomenų įkrovimas. Šio TC tikslas yra patikrinti, ar išsaugoti duomenys yra teisingai įkraunami į makroprogramą, ar teisingai rodomi vartotojui.

Duomenų redagavimas. Pagal reikalavimus įvesti duomenys gali būti redaguojami. Tikrinamos įkrautų duomenų redagavimo galimybės, tikrinama, ar po duomenų redagavimo teisingai projektuojamas modelis.

Vartotojo sąsajos tikrinimas. Šios funkcinės srities testavimo tikslas yra patikrinti, ar visi grafinės vartotojo sąsajos elementai yra įgyvendinti sistemoje, kaip jie veikia tarpusavyje. Taip pat šiai funkcinei sričiai priskiriamas sistemos stiliaus testavimas.

Modelio projektavimas. Tai pati svarbiausia funkcinė sritis, jos testavimui skiriama daugiausiai dėmesio ir laiko. Tikrinama, ar baldų modelis teisingai atvaizduojamas *AutoCAD* projektavimo lange, ar braižymui naudojami teisingi matmenys, ar brėžinio matmenys atitinka modelyje pateiktiems matmenims, ar pateikta modelio specifikacija rodo teisingą informaciją.

Modelio išsaugojimas į failą. Šio TC tikslas yra patikrinti, ar suprojektuotas modelis yra teisingai išsaugomas į failą, ar jis išsaugomas teisingu formatu ir ar teisingas yra išsaugoto failo turinys.

Testavimo rezultatai ir jų reikšmė

Sistema ištestuota remiantis regresinio testavimo principu, t.y. po kiekvieno klaidos pataisymo buvo patikrinta ne tik ištaisyta klaida, bet ir su šia klaida susijusios funkcinės sritys (tam, kad būtų įsitikinta, jog taisant klaidą nebuvo pažeistos šios sritys). Testavimo metu rastos klaidos ir problemos ištaisytos. Pilnas sistemos testavimas baigtas, įvykdžius visus TC ir neradus daugiau sistemos klaidų.

4.7. Apibendrinimas

Šiame skyriuje buvo aprašyta sistema, jos kūrimo etapai nuo reikalavimų analizes, kodo parašymo, testavimo iki atitinkamų failų importavimo į serverį. Sistema buvo kuriama remiantis *krioklio* tipo sistemos kūrimo gyvavimo ciklu ir *ASP.NET* bei *VBA AutoCAD* technologijomis. Atskleisti sistemos

projektavimo ir konstravimo aspektai ir įrodyta, kad šios technologijos puikiai tinka sistemos, skirtos baldų pardavimo ir projektavimo procesams optimizuoti, kūrimui.

4.8. Tolesnio sistemos tobulinimo ir plėtojimo galimybės

Sistemą galima toliau plėtoti ir tobulinti, įdiegiant „B“ ir „C“ spintelių konfigūravimo ir projektavimo funkcionalumus, papildant tinklalapio sistemą užsakymo kainos skaičiavimu. Ateityje galima įdiegti realaus konfigūruojamo baldų modelio vaizdo pateikimo sąsają, galima patobulinti automatizuoto projektavimo algoritmą.

IŠVADOS IR PASIŪLYMAI

1. Išnagrinėjus baldų verslo procesus, nustatyta, kad pardavimų ir gamybos procesai organizuojami neefektyviai. Kliento poreikius nusakančios informacijos surinkimo, analizavimo, dokumentavimo ir užsakymo formavimo metu sukaupta informacija keliaudama iki gamybos yra apdorojama rankiniu būdu dalyvaujant įvairių grandžių darbuotojams. Taip apdorojama informacija ilgina procesą ir lemia klaidų atsiradimą.

2. Buvo pasiūlytas šių procesų optimizavimo sprendimas, pagal kurį klientas tiesiogiai internetiniame tinklalapyje pateikia užsakymą, o pagal gautą informaciją makroprograma automatizuotai formuoja gamybai reikalingą techninę dokumentaciją *AutoCAD* aplinkoje.

3. Siūlomai internetinei sistemai realizuoti buvo panaudota *ASP.NET* technologija. Lyginant su *Java* ir *PHP* technologijomis, *ASP.NET* yra dinamiška ir daug kalbų palaikanti technologija, užtikrinanti nuoseklią objektinio programavimo ir kodų vykdymo aplinką, kuri sumažina programinės įrangos kūrimo ir diegimo laiko sąnaudas. Naudojant *ASP.NET* technologiją sukurtos taikomosios programos veikia našiau nei sukurtos naudojant *Java* technologiją.

4. Automatizuoto baldų projektavimo makroprogramos kūrimui buvo naudojama *AutoCAD VBA* programavimo technologija. *VBA* technologija yra pranašesnė už *AutoLISP* technologiją, nes naudojant *VBA* galima realizuoti objektinį programavimą, veiklos logiką skaidyti į specialias klases ar modulius, daug paprastesnis yra formų kūrimas ir palaikymas. *VBA* makroprogramas galima susieti su kitomis programomis, jose galima naudoti *COM* bibliotekas. *VBA* programos veikia sparčiau nei *AutoLISP*.

5. Nustatyta, kad įmonė, įdiegusi sukurtą baldų e.užsakymo ir automatizuoto projektavimo sistemą, gautų šios naudos:
    - § sumažėtų užsakymo formavimo ir projektavimo laiko sąnaudos;
    - § gamintojui būtų pateikta tiksli kliento poreikių informacija;
    - § projektuojant automatizuotai būtų išvengiama žmogiškojo faktoriaus klaidų;
    - § atsiradus e.užsakymo pateikimo galimybei, padaugėtų klientų;
    - § sumažėtų užsakymo įvykdymo laikas;
    - § padidėtų baldų įmonės produktyvumas.

LITERATŪROS SĄRAŠAS

1. Ahmad J., Sabah A. Computer-Integrated System for Estimating the Costs of Building Projects. *Journal of Architectural Engineering*. 2007, Vol. 13, Issue 4, p. 205-223.

2. AutoCAD 2010 User Documentation Overview of ObjectARX. [žiūrėta 2009-04-17]. Prieiga per internetą: <http://docs.autodesk.com/ACD/2010/ENU/AutoCAD%202010%20User%20Documentation/index.html>

3. Ballenger R. M. An eCommerce Development Case: Your Company's eCommerce Web Site. *Journal of Information Systems Education*. 2007, Vol. 18, Issue 4, p. 409-414. ISSN:1055-3096.

4. Basa M. M. Customization Methods in AutoCAD. [žiūrėta 2009-04-17]. Prieiga per internetą: <http://www.dailyautocad.com/autolisp/customization-methods-in-autocad>

5. Birnbaum M. H. Human Research And Data Collection Via The Internet. *Country of Publication*. 2004, Vol. 55, p 803-832. ISSN: 0066-4308.

6. Belevičius R. Java technologijos. Vilnius: Technika, 2005, 133 p.

7. Berners-Lee T. Uniform Resource Locators (URL) [žiūrėta 2009-04-17]. Prieiga per internetą: <http://www.ietf.org/rfc/rfc1738.txt>

8. Bollaert J. Advantages and disadvantages of client-side scripts [žiūrėta 2009-04-17]. Prieiga per internetą: <http://www.opensourcetutorials.com/tutorials/Design-And-Layout/Usability/web-widgets-part-2/page2.html>

9. Booch G. Unified Modeling Language User Guide. Addison Wesley, 1998, p. 469. ISBN: 0-201-57168-4

10. Boudreaux T. J. PHP 5 vaizdžiai. Kaunas: Smaltijos leidykla. 2007, p. 320.

11. Darryl K. ASP.Net AJAX comes to Linux. Move will let developers use Microsoft's platform to build apps for Linux. *eWeek*. 2008, Vol. 25, Issue 13, p. 30. ISSN: 1530-6283.

12. Delisle P., Luehe J., Roth M. JavaServerPages™ Specification: version 2.1. 2006.

13. Differences between ASP and ASP.NET [žiūrėta 2009-04-17]. Prieiga per internetą: <http://www.w3schools.com/aspnet/aspnet_vsasp.asp>

14. Domeika P. Įmonės apskaitos informacinės sistemos kūrimo metodologiniai aspektai. *Management Theory & Studies for Rural Business & Infrastructure Development*. 2008, Vol. 15, Issue 4, p. 41-49.

15. Edwards M. D. Handling and Avoiding Web Page Errors Part 1: The Basics. Scripting (General) Technical Articles. [žiūrėta 2009-04-17]. Prieiga per internetą: <http://msdn.microsoft.com/en-us/library/ms976140.aspx>

16. Erbschloe M. Business Applications of Java. *Research Starters Business*. 2008, p.1-14.

17. Garrnett J. J. Ajax: A New Approach to Web Applications. [žiūrėta 2009-01-10]. Prieiga per internetą: <http://www.adaptivepath.com/ideas/essays/archives/000385.php>

18. Goodwill J. Java Web Applications. [žiūrėta 2009-04-17]. Prieiga per internetą: <http://www.onjava.com/pub/a/onjava/2001/03/15/tomcat.html>

19. Habersack M. Guide: Porting ASP.NET Applications. [žiūrėta 2009-02-19]. Prieiga per internetą: <http://www.mono-project.com/Guide:_Porting_ASP.NET_Applications>

20. Hay D. C. Requirements Analysis: From Business Views to Architecture. New Jersey: Prentice Hall, 2003, 460 p. ISBN 0-13-028228-6.

21. Huang M.Y., Lin Y.J. A framework for web-based product data management using J2EE. *International Journal of Advanced Manufacturing Technology*. 2004, Vol. 24, Issue 11/12, p. 847-852.

22. Information technology - Software Product Evaluation - Quality characteristics and guidelines for their use. ISO/IEC 9126 – 1991. [žiūrėta 2009-03-02]. Prieiga per internetą: <http://www.cse.dcu.ie/essiscope/sm2/9126ref.html>

23. Introduction to ASP.NET. Developer's Guide. [žiūrėta 2009-04-17]. Prieiga per internetą: <http://msdn.microsoft.com/en-us/library/4w3ex9c2(VS.71).aspx>

24. JavaTM Servlet and JavaServer PagesTM Technology Comparing Methods for Server-Side Dynamic Content. White paper. [žiūrėta 2009-02-19]. Prieiga per internetą: <http://java.sun.com/products/jsp/jspservlet.html>

25. Jančoras Ž., Okulič-Kazarinas M. Laisvieji informaciniai sprendimai: mokomoji knyga. Vilnius: Technika, 2008, 276 p.

26. Jonauskas G. 12-osios Lietuvos Jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis“ 2009 metų teminės konferencijos KOMPIUTERINĖ GRAFIKA IR PROJEKTAVIMAS (2009 m. balandžio mėn. 8 d.) medžiaga.

27. KitchenDraw. [žiūrėta 2009-04-17]. Prieiga per internetą: <http://www.aga-cad.lt/produktai/gamybai/kitchendraw/227#Apzvalga>

28. Kriučkova M. Ajax – naujas požiūris į žiniatinklio aplikacijų kūrimą. 10-osios Lietuvos Jaunųjų mokslininkų konferencijos „Mokslas – Lietuvos ateitis“ 2007 metų teminės konferencijos KOMPIUTERINĖ GRAFIKA IR PROJEKTAVIMAS (2007 m. balandžio mėn. 11 d.) straipsnių rinkinys. Vilnius: Technika, 2007, p. 93-99.

29. Langley N. Avoid „gibberish“ of Ajax with free MS framework. *Computer Weekly*. 2008, p. 42-48. ISSN: 0010-4787

30. Langley N. ASP.net eases move to web for Windows developers. *Computer Weekly*. 2007, p. 28-34 ISSN:0010-4787

31. Martin R. C. UML Tutorial: Sequence Diagrams. [žiūrėta 2009-01-10]. Prieiga per internetą: <http://www.objectmentor.com/resources/articles/UMLSequenceDiagrams.pdf>

32. Omura G. Mastering AutoCAD and AutoCAD LT 2008. Indianapolis: Wiley Publishing, Inc., 2007, 957 p. ISBN: 978-0-470-13738-3

33. Popovas V., Jarmolajevas A., Grigorjeva T. Šiuolaikinės automatizuotos projektavimo sistemos. *Statyba*. 2003, Nr. 6, 7.

34. Popovas V., Jarmolajevas A., Grigorjeva T. Projektavimo sistemos, užtikrinančios architektų darbo kokybę. *APS*, 2007.

35. Popovas V., Sniečkus V., Jarmolajevas A. Kompleksinis inžinerinių uždavinių sprendimas, taikant integruotas kompiuterinio projektavimo sistemas. *Aviacija*, 1999.

36. Richards K., Castillo C. Why Model With UML? [žiūrėta 2009-04-17]. Prieiga per internetą: <http://www.netbeans.org/kb/55/uml-why-model.html>

37. Ruvalcaba Z. Build Your Own ASP.NET 2.0 Web Site Using C# & VB. Collingwood:SitePoint Pty, 2006, 721 p. ISBN 0-9752402-8-5.

38. Sobocinski P. Moving your web app to the client-side. [žiūrėta 2009-04-17]. Prieiga per internetą: <http://dev.aol.com/node/333>

39. Sokas A. Grafinių objektų ir informacijos valdymas VBA kalba. Mokomoji knyga. Vilnius: Technika, 2007. 59 p.

40. Sommerville I. Software Engineering (6th edition). Harlow: Addison Wesley Longman, 2000, 720 p. ISBN: 978-0-201-39815-1

41. Taft, D. K. Dynamic languages gain steam. *eWeek*. 2008, Vol. 25, Issue 8, p. 28-29.

42. Tomcat Apache. [žiūrėta 2009-04-17]. Prieiga per internetą: <http://tomcat.apache.org/>

43. VisualStudio2010 Product Overview. [žiūrėta 2009-04-17]. Prieiga per internetą: <http://www.microsoft.com/visualstudio/en-us/products/2010/default.mspx>

44. What is CAD Customization? [žiūrėta 2009-04-17]. Prieiga per internetą: <http://www.caddsoftsolutions.com/What-is-CAD-Customization.htm>

45. .NET Framework Overview. [žiūrėta 2009-01-10]. Prieiga per internetą: <http://www.microsoft.com/net/Overview.aspx>

PRIEDAI