KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

MULTIMEDIJOS INŽINERIJOS KATEDRA

Simona Bartauskaitė

Macromedia Flash ir XML technologijų jungimas naudojant objektinį programavimą

Magistro darbas

Darbo vadovas

doc. A. Lenkevičius

Kaunas, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
MULTIMEDIJOS INŽINERIJOS KATEDRA

Simona Bartauskaitė

Macromedia Flash ir XML technologijų jungimas naudojant objektinį programavimą

Magistro darbas

Kalbos konsultantas
Lietuvių k. katedros lekt.
J. Jonušas
2006-05-24

Vadovas
doc. A. Lenkevičius
2006-05-25

Recenzentas
Doc. B. Paradauskas
2006-05-26

Atliko
IFM 0/3 gr. Stud.
Simona Bartauskaitė
2006-05-23

Kaunas, 2006

TURINYS

1. ĮVADAS

Šiuolaikinis žmogus tikriausiai net neįsivaizduoja savo gyvenimo be kompiuterio. Nuo pat pirmojo kompiuterio sukūrimo dienos programinę įrangą kuriančios kompanijos, „Microsoft“, „Apple“ ir kitos stengiasi sukurti kuo aiškesnę, paprastesnę, vaizdesnę programinę įrangą.

Kuriant internetines aplikacijas Action Script 2.0 programavimo kalba suteikia kur kas daugiau galimybių nei Visual Basic, C ar net Java. Programavimas Action Script 2.0 tapo labai patogus, paprastas bei atliekamas pagal objektiškai orientuoto programavimo principus.

Su kiekviena nauja Action Script 2.0 versija, Flash kūrėjai stengėsi išleisti kuo patogesnę ir funkcionalesnę programavimo kalbą. Siekiant konkuruoti kuriant profesionalias interneto aplikacijas Macromedia patobulino programavimą Action Scriptu išleisdama naują Action Script 2.0 versiją. Programavimo kalba užtikrina funkcionalumą ir programavimo paprastumą. Visa programavimo kalba buvo pertvarkyta, joje pridėta naujų standartų bei galimybių.

Darbo tikslas — naudojantis objektinio programavimo principais sukurti sistemą, skirtą bendradarbiauti su XML dokumentais, iš kurių bus nuskaitoma informacija bei pateikiama į programą. Sistema turi veikti internete. Ši sistema nebus diegiama, jos administravimas ypač paprastas, o valdymas aiškiai suvokiamas net nepatyrusiam kompiuterio vartotojui.

Sistemos reikalavimai:

- Sukurta sistema turi būti ypač paprasta naudojimui.
- Visi meniu punktai ir pasirinkimai turi būti intuityviai suvokiami bei paprastai pasiekiami.
- Sistema turi būti stabili.
- Sistema turi turėti apsaugą nuo teksto bei paveiksliukų kopijavimo.
- Sistema turi būti pasiekiama internetu.

Sistemos naujumas. Lietuvos rinkoje neaptikau informacinės sistemos, skirtos pradedančioms arba esamoms siuvėjoms, kurios rastų informaciją apie įvairius drabužių modelius bei sužinotų įvairių jų siuvimo subtilybių. Taigi sistema bus pritaikyta šiai vartotojų grupei. Sistema bus kuriama su galimybe ateityje susieti ją su duomenų bazėmis, kuriose bus saugomos įvairių modelių iškarpos.

Projekto užsakovas: Kauno technologijos universiteto Informatikos fakulteto Mutimedijos katedra. Atstovas: docentas dr. Antanas Lenkevičius (Antanas.Lenkevicius@ktu.lt).

Darbo vykdytoja IFM-0/3 studentė Simona Bartauskaitė (simonab@kaunas.init.lt).

2. XML IR OBJEKTINIO PROGRAMAVIMO ANALIZĖ

2.1. Pagrindinės objektinio programavimo savybės

Objektininis programavimas (OP) – tai ankstesnių programavimo metodologijų evoliucijos rezultatai. OP yra labiau stabilizuotas, mobilus ir abstraktus, lyginant su tradiciniu. Trys pagrindinės OP charakteristikos:

- Inkapsuliacija.
- Paveldėjimas.
- Polimorfizmas. (2)

Tradicinio (procedūrinio) programavimo pagrindiniai trūkumai:

1. Reikia tiksliai ir iki galo specifikuoti uždavinį, aprašyti duomenis.
2. Sunku atlikti pakeitimus programos projekte.
3. Nenumatytas sukurto projekto tobulinimas. (2)

Tradicinis programavimas — tai dekompozicjos principo ("skaldyk ir valdyk") panaudojimas — t. y. algoritminė dekompozicija, ji orientuojasi į procesus ir veiksmus.

Objektinis programavimas paremtas objektine dekompozicija, t.y. skaidoma į objektus, o ne į algoritmus. Objektas turi savo duomenis - būseną ir galimus veiksmus - elgesį. (2)

2.1.1. Inkapsuliacija

Inkapsuliavimas - savybių (duomenų) ir elgesio (operavimo metodų) jungimo į vieną visumą procesas, vadinamą objektu.(2)

Objektinio programavimo taisyklės reikalauja, kad klasę sudarytų dvi dalys: viešoji ir privačioji. Viešosios dalies metodus gali kviesti bet kuri funkcija arba kitos klasės metodas. Viešoji dalis turi leisti atlikti visus veiksmus, kurie numatyti klasės kūrėjo, bet neatidengti vidinės objekto struktūros. Privačios klasės metodų.(8)

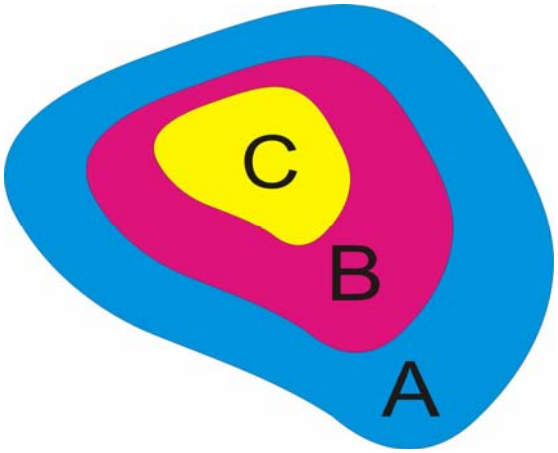
Privačios dalies metodai ir kintamieji nematomi „pašaliniams“, sakoma, kad jie *inkapsuliuoti*. Tai išsaugo objektų universalumą, tačiau objektą galima naudoti nežinant, kaip jis padarytas.

Inkapsuliacijos rezultatas yra naujas duomenų tipas — klasė.(8)

2.1.2. Paveldėjimas

Paveldėjimas - procesas, kuriame palikuonių klasė įgauna visas protėvio (tėvo) klasės charakteristikas ir elgesį. Tai vienos klasės, vadinamos protėviu, metodų ir kintamųjų (savybių) perdavimas kitai klasei, vadinamai palikuoniu.(2, 8)

Paveldėjimo principus galima pavaizduoti ir grafiškai (1. pav.). Čia klasė A yra klasės B palikuonis, o klasė B yra klasės C palikuonis. Tai ryšys, o ne vienkartinis veiksmas, nes, pakeitus protėvio klasę, pasikeitimai automatiškai pereina palikuoniui. Protėvis gali turėti neribotą skaičių palikuonių. Be to, kai kurios OP kalbos laidžia vienam palikuoniui turėti keletą protėvių.(2, 8)



1. pav. Paveldėjimas (6)

Klasės gali modeliuoti ne tik konkrečius dalykus, bet ir idėjas. Tokios klasės vadinamos *abstrakčiomis*. Jose aprašomos tik labai apibendrintos priklausomybės, todėl abstrakčių klasių tipo objektai nekuriami. Tačiau šios klasės yra puikūs protėviai kitoms klasėms, kurios turi pasižymėti reikalingomis vienodomis savybėmis.(2, 8, 9)

2.1.3. Polimorfizmas

Polimorfizmas — galimybė vienodai pavadintą veiksmą realizuoti skirtingais būdais skirtingoms objektų klasėms.(2)

Objektai turi visus savo klasės bei protėvių klasių metodus ir savybes, todėl objektus galima traktuoti kaip ir bet kuriuos jų klasės protėvius. Objektai paveldi visų tiesioginių ir netiesioginių protėvių tipus. Jie tarsi turi keletą tipų, tai vadinama *polimorfizmu*.(2, 8)

Tuo pat metu objektas yra dviejų tipų: tikrasis, kuris buvo nurodytas sukuriant objektą, ir kintamasis. Tikrais tipas nesikeičia. Vykdant programą kintamasis tipas gali keistis į objekto tikrąjį tipą arba bet kuriuo protėvio tipą. Kai kalbama apie objekto tipą, paprastai turimas omenyje tikrasis tipas.(2, 8, 9)

Objektai gali būti naudojami ten, kur reikalingi jų protėvio tipo objektai. Tai labai naudinga, nes galima sukurti universalius metodus, galinčius tinkamai dirbti su daugelio tipų objektais.(8)

Kitaip galime sakyti, kad polimorfizmas — tai galimybė sudaryti bendrą pagal vardą (paprogramės arba funkcijos) veiksmą, galiojantį tuo pat metu visiems hierarchijos objektams. Be to, kiekvienas konkretus objektas hierarchijoje gali turėti to veiksmo specifinę realizaciją, t.y. skirtingi objektai gali turėti taip pat pavadintus parametrus ir metodus, tačiau į kreipimąsi reaguoti skirtingai.(8, 9)

2.1.4. Objektai

Objektai yra pagrindiniai OP elementai.

Objektas yra duomenų (kintamųjų) ir veiksmų su jais, vadinamų metodais, rinkinys.

Objektinio programavimo kalboje duomenys ir veiksmai (programos kodas) sujungiami į vieną objektą. Tokia struktūra pasirinkta modeliuojant realų pasaulį. Pavyzdžiui, kiekvienam fiziniam objektui būdingos tam tikros savybės: žmogui — vardas, pavardė, amžius, ūgis, svoris, lytis; automobiliui — rūšis, variklio galingumas, didžiausias greitis, suvartojamų degalų kiekis, spalva ir pan.. Programuojant objektai taip pat turi panašias savybes, kurias atitinka parametrai (kintamieji), taip pat dažniausiai vadinami savybėmis. Jų reikšmės kiekviename objekte gali būti skirtingos.(8, 9)

Objektai yra viena iš struktūrinių duomenų formų, jie gali turėti kitų objektų kaip kintamųjų.

2.1.5. Metodai

Metodai lemia veiksmus, kuriuos objektas gali atlikti. Jie leidžia sužinoti objekto kintamųjų reikšmes arba jas pakeisti, panašiai kaip gyvenime įvairiais veiksmais kaičiame daiktų būsenas, pavyzdžiui, dažome namą, į stiklinę pripilame vandens.(9)

2.1.6. Įvykiai

Įvykis — tai veiksmas arba rezultatas, kurį atpažįsta objektas.

Įvykiui galima parinkti (suprogramuoti) norimą ataskaitą. Įvykis gali įvykti dėl pačios programos, vartotojo arba kitų programų veiksmų. Pavyzdžiui, pelės paspaudimas, lango uždarymas ir pan.. Paprasčiau galime sakyti, kad *savybės* (parametrai) aprašo objektus, *metodai* leidžia šiems objektams atlikti tam tikrus priskirtus veiksmus, o *įvykiai* yra tai, kas atsitinka, kai objektas atlieka kokius nors veiksmus.(9)

2.1.7. Klasės

Klasė — aibė objekto charakteristikų, kuri yra naujas duomenų tipas, naudojamas kaip šablonas objektams kūrti, apimantis reikšmes ir operacijas. Klasė yra tipo koncepcijos išplėtimas.

Realiame gyvenime objektus mes rūšiuojame, vadiname bendrais pavadinimais, pavyzdžiui: dviračiai, automobiliai, žmonės. Visi vienos grupės atstovai turi tas pačias savybes, gali atlikti tuos pačius veiksmus. Objektiškai programuojant sakoma, kad šie objektai (daiktai) priklauso vienai klasei, pavyzdžiui, dviračių, automobilių, žmonių. Klasė tik aprašo metodus ir kintamuosius, bet nesaugo konkrečių jų reikšmių. Reikšmės saugomos objektuose.

Objektai yra tam tikros klasės tipo kintamieji.(8, 9, 10)

2.1.8. Metodų uždengimas

Protėvių metodai yra paveldimi, tačiau jie ne visada tinka palikuonims. Tada šiuos bereikalingus metodus reikia „išjungti“, „uždengti“.

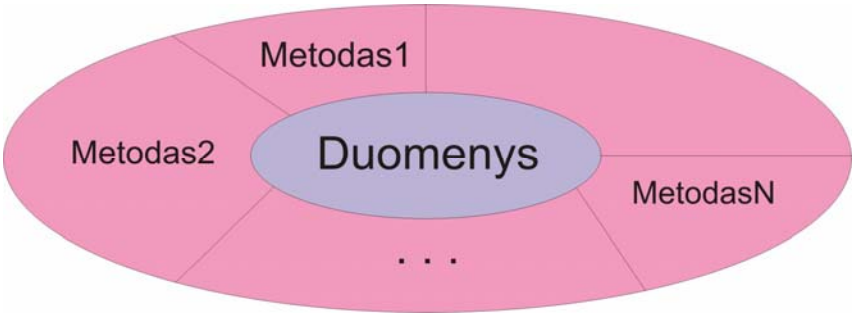
Metodų uždengimas — tai naujo kodo suteikimas protėvių metodams klasėse — palikuoniuose.

Uždengiantysis metodas turi turėti tą patį vardą ir parametrų tipus kaip ir uždengimas. Uždengimo atveju tinkamas metodas kviečiamas atsižvelgiant į objekto tipą. Kaip ir klasės, metodai gali būti abstraktūs: neturi savo kūno, o tik aprašą. Tokius metodus klasės-palikuoniai turi uždengti.(8)

2.2. Objektinio programavimo Java kalba ypatybės

2.2.1. Duomenų ir metodų sujungimas į visumą (encapsulation)

Duomenų ir jų apdorojimo metodų sujungimas į visumą Java kalboje atliekamas naudojant klasę. Taigi klasė yra tipas arba objekto šablonas, kurį sudaro duomenys ir metodai. Objektas jau yra realus klasės egzempliorius kompiuterio atmintyje. Objekto modelį galime pavaizduoti kaip kapsulę:



2. pav. Objekto modelis. (6)

Taigi sudarius A klasę, joks realus objektas atmintyje dar nesukuriamas:

```
class A {
     // klasės turinys
```

```
}
```

Objekto dar nėra ir paskelbus A klasės tipo kintamąjį x:

```
A x;
```

Objektas gali būti sukurtas ir sakiniu new:

```
x = new A(); // Objektas sukurtas!
// arba iš karto:
A x = new A();
```

Taigi objektas sukuriamas dinamiškai programos vykdymo metu sakiniu new:

Klasės Vardas objekto Vardas = **new** *Klasės Vardas();*

Arba dviem etapais:

Klasės Vardas objekto Vardas;

Objekto Vardas =**new** *Klasės Vardas();*

Klasei galima sukurti neribotą skaičių objektų.

Grįžkime prie kapsulės. Ji turi būti uždara. Pirma, tai užtikrina duomenų apsaugą, nes vartotojas gali atlikti su duomenimis tik tas operacijas, kurias jam leidžia tam skirti metodai. Antra, programuotojas-vartotojas gali rašyti programas „aukštesniu“ lygiu, t.y. jis visiškai pasitiki esamais metodais ir jam nebereikia leistis į smulkmenas bei jų tikrinti. Ir trečia, programų (metodų) pakeitimai ir atnaujinimai programuotojui-vartotojui visai neturi reikšmės. (9)

2.2.2. Paveldėjimas (*inheritance*) ir kompozicija

Tai klasės sugebėjimas paveldėti protėvių klasės duomenis ir metodus. Literatūroje naudojama daug skirtingų terminų. Pagrindinė klasė vadinama paveldimąja klase, superklase, protėvių klase, bazine arba tėvo klase. Analogiškai naujoji klasė vadinama paveldinčiąja klase, subklase, palikuonių klase, išvestine arba vaiko klase.(9)

Vaiko klasė paveldi visus matomus (ne *private* tipo) tėvo klasės metodus ir kintamuosius. Visos Java klasės yra kilusios iš *java.lang.Object* klasės ir automatiškai palaiko visus jos metodus. Java neturi daugialypio paveldėjimo (netiesiogiai tai galima išspręsti naudojant sąsajas). (9)

Nereikia painioti sąvokų „kompozicija“ ir „paveldėjimas“. Kompozicija — tai kitos klasės objekto panaudojimas naujai projektuojamoje klasėje. Kompozicija naudojama tada, kai kuriama

nauja klasė tiesiog naudoja kitos klasės metodus. Paveldėjimas naudojamas tada, kai nauja klasė naudoja kitos klasės struktūrą (interfeisą).(9)

2.2.3. Polimorfizmas (polymorphism)

Galima skirti dvi polimorfizmo rūšis. Tai metodų perkrova ir metodų užklotis.

Metodų perkrova (*overloading*). Klasėje naudojama keletas metodų tuo pačiu vardu. Būtina sąlyga — metodai turi skirtis savo antraštėmis (parametrų skaičiumi arba parametrų tipais). Grąžinamos reikšmės tipas čia įtakos neturi. (9)

Metodo pasirinkimą nusako jo iškvietimo formatas. Tai atliekama jau kompiliavimo metu (tai dar vadinama „ankstyvuoju susiejimu“). (6)

Metodų užklotis (overriding). Tėvo ir vaiko klasės turi vienodus metodus. Būtinos sąlygos: turi sutapti ne tik šių metodų vardai, bet ir jų antraštės bei grąžinamų reikšmių tipai. Taigi vaiko klasės metodas gali pakeisti (užkloti) tėvo klasės metodą. (9)

Metodo pasirinkimą lemia objekto, kuriam šis metodas kviečiamas, tipas (jei objektas yra tėvo klasės tipo, tai bus kviečiamas jos metodas, priešingu atveju — vaiko). (9)

Metodas yra parenkamas vykdymo metu (tai dar vadinama „susiejimu vykdant“). (9)

2.2.4. Trys Java programų tipai

Pirmas. Taikomoji programa (application) su komandų eilutės sąsaja.

Tai pats paprasčiausias ir kartu nevaizdžiausias programos tipas. Programa paleidžiama main metodu. Taigi bent viena programos klasė privalo turėti main metodą. Šiaip jau kiekviena klasė gali turėti po vieną main metodą (tai patogu testuojant klases), tačiau vienu metu galima naudoti tiktai vieną main metodą. Pasibaigus visiems main metodo sakiniams, programa baigia darbą. (9)

Pirmojo tipo Java programa, kuri juodame Command Prompt (arba FAR Manager) lange pateikia tekstą „Labas, Java“.

Etapai:

1. Bet kuriuo redaktoriumi (kad ir NotePad) parašome programos tekstą (visi Java kalbos žodžiai pajuodinti):

```
//Pirmoji Java programa
Public class Labas{
    Public static void main(String args[]) {
        System.out.println(„Labas, Java“);
    }
}
```

Ir įrašome į diską vardu „KlasėsVardas.java“ (šiuo atveju vardu „Labas.java“).

1. Kviečiame kompiliatorių **javac.exe**, kuris patikrina programos sintaksę, perkoduoja jos tekstą į bait-kodą ir rašo jį į diską vardu „KlasėsVardas.class“ (šiuo atveju „Labas,class“):

```
>javac Labas.java
```

Jeigu kompiuteryje nėra nustatyto kelio į katalogą, kur yra kompiliatorius **javac.exe**, tai reikia nurodyti visą kelią iki jo:

```
>D:\jdk1.3\bin\javac Labas.java
```

Jeigu nėra klaidų, tai kompiliatorius paprastai jokių pranešimų ir neišveda.

2. Kviečiama JVM **java.exe** vykdyti programą (failo tipas class nenurodomas):

```
>java Labas
```

Naudojant vizualias aplinkas (JBuilder, Visual Sturio J++, VisualCafe ar kitas), vykdymo scenarijus keičiasi.

Antras. Taikomoji programa su vartotojo sąsaja.

Programa čia taip pat paleidžiama **main** metodu. Tačiau paleidimo tikslas dažniausiai yra tik vienas — sukurti vartotojo sąsajos langą ir perduoti jam visą valdymą. Toliau jau sąsajos elementais (meniu, mygtukais, teksto laukais ir t.t.) atliekami norimi veiksmai ir programa veiks tol, kol uždarysime šios sąsajos langą.(6)

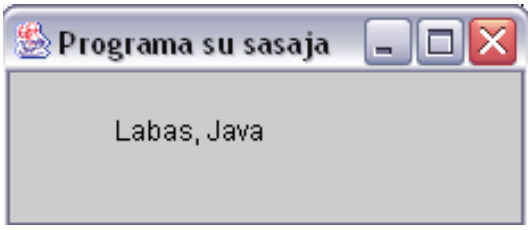
Vėlgi rašome tą pačią programą, išvedančią tekstą ‚Labas, Java“. Programos rašymo ir paleidimo scenarijus visiškai toks pat kaip ir pirmojo tipo programos. Tiesa, programos tekstas atrodys truputėlį kitoks:

```
//Pirmoji Java programa su vartotojo sąsaja.
import javax.swing.*;
import java.awt.*;
public class LabasSuSasaja extends JFrame {
    public void paint(Graphics g){
         g.drawString(„Labas, Java“, 50, 60):
    }
    public static void main(String[] args) {
         LabasSuSasaja langas = new LabasSuSasaja();
```

```
            langas.setDefaultCloseOperation
            (JFrame.EXIT_ON_CLOSE);
            langas.setSize(200, 100);
            langas.setTitle(„Programa su sąsaja“);
            langas.setVisible(true);
        }
    }
```

Vėl kompiliuojame ir vykdome programą:

```
>D:\jdk1.3\javac LabasSuSasaja.java
>java LabasSuSasaja
```



3. pav. Vartotojo sąsaja. (6)

Trečias. Apletas (applet).

Šio tipo taikomąją programą turi tiktai Java. Apletas — tai programa, klaidžiojanti internete ie vykdoma „kliento“ kompiuteryje. Ji nenaudoja **main** metodo. Tačiau kaip ir taikomojoje programoje su vartotojo sąsaja, apleto vykdymas paremtas įvykių apdorojimu. Naršyklė pati rūpinasi apleto veikimu: stabdo jį pereinant į kitą langą, perpiešia keičiant lango dydį ir t.t. (9)

Parašysime tą pačią programą, bet kaip apletą.

Etapai:

1. Naudodamiesi bet kuriuo redaktoriumi prašome programos-apleto tekstą:

```
//Pirmas Java apletas
import java.applet.*;
import java.awt.*;
public class LabasApletas extends Applet {
     public void paint(Graphics g) {
          g.drawString(„Labas, Java“, 50, 60);
     }
}
```

Parašę tekstą įrašome į diską vardu „LabasApletas.java“.

2. Šis etapas analogiškas — kviečiame kompiliatorių **javac**, ir šis gautą bait-kodą įrašo į diską vardu „LabasApletas.class“.

```
>javac LabasApletas.java
```

3. Tačiau jis vykdomas visiškai kitaip negu taikomoji programa. Apletas vykdomas naršykle (galima ir su appletviewer.exe), todėl bet kuriuo redaktoriumi sukuriamas nedidelis HTML failas. Šį failą bet kuriuo vardu įrašome ten pat, kur yra mūsų gautas LabasApletas.class failas:

```
<HTML>
     <applet CODE = „LabasApletas.class“
         Width = 200 height = 100>
     </applet>
</HTML>
```

4. Atidarome šį HTML failą bet kuria naršykle (Explorer, Netscape ar Opera) arba pasinaudojame SDK sudėtyje esančia appletviewer programa (appletviewer xxx.html). Naršyklė savo viduje turi JVM, todėl ji pati ir vykdo šį apletą. (9)

Kai kurios vizualios aplinkos HTML failą generuoja automatiškai.

Naudojant aplete Swing elementus (pirmųjų sąsajų elementai buvo vadinami AWT), anksčiau minėtos struktūros HTML failas kai kurioms naršyklėms nebetinka (appletviewer niekas nepasikeitė). Esmė ta, kad Java 2 versijos nuorodos applet html faile buvo atsisakyta. Apletai dabar paleidžiami naršyklių papildymo programomis Java Plug-in. Jos automatiškai įtraukiamos į naršykles, bet jei jų nėra, tai galima ir atskirai atsisiųsti (nemokamai). (9)

2.3. Objektinio programavimo Visual Basic kalba ypatumai

Objektinio programavimo būdu parengta programa sudaroma iš tam tikrų objektų. Objektas yra iš anksto ar naujai parengtas nepriklausomas programos fragmentas, pavyzdžiui, mygtukas, diagrama, ataskaita. Kuriant objektą, loginė jo prasmė atskiriama nuo realizacijos ypatybių. Pavyzdžiui, objektų klasė „Mygtukas" turi savybes, būdingas visiems mygtukams, tačiau konkretaus mygtuko (objekto) savybės, pavyzdžiui, dydis, spalva, vieta programos pulte, nurodomos jį panaudojant programoje. (8, 10)

Objektai programoje vienaip ar kitaip vaizduoja realaus pasaulio reiškinius. Jų reikia šiems reiškiniams programoje modeliuoti. Pavyzdžiui, objektas CommandButton modeliuoja komandų mygtuką. Į programos pultą jis įterpiamas iš VB lange esančiu objektų klasių (mygtukų objektams

kurti) rinkinio, paspaudus mygtuką ir pelyte -nurodžius jo vietą bei dydį. Mygtuko elgesys ir būsena nustatomi parametrais ir metodais. VB objektai nuo objektų klasių skiriasi tuo, kad objektai egzistuoja tik programai veikiant, o objektų klasės tik aprašo klasės objektų sandarą. Todėl galima naudoti reikiamą kiekvienos klasės objektų kiekį.(8, 10)

Objektai abstrahuodami duomenis, panašiai kaip kalbos sąvokos, supaprastina sudėtingų reiškinių aprašymą. Sunku įsivaizduoti, kaip reikėtų pasakyti, kad „čia yra daug medžių“, jeigu nebūtų apibendrinančios sąvokos „medis“. Tikriausiai reikėtų smulkmeniškai apibūdinti kiekvieną medį. (8, 10)

VB yra turtingas dažnai Windows terpėje naudojamų objektų klasių rinkinys. Rengiant programas, galima sukurti ir savo objektų klases.(8, 10)

Objektai yra apsaugoti nuo išorinio poveikio, jų savybės yra aprašomos tik objektui b8dingais parametrais ir metodais. Tai išsaugo objektų universalumą ir objektą galima naudoti nežinant, kaip jis padarytas. Šią objektų savybę vadina *inkapsuliacija.(8, 10)*

IŠ jau esamų objektų galima sukurti objektus, kurie naudoja jau esamų objektų kodus. Šią objektų savybę vadina *paveldimumu.* Ši savybė leidžia sutrumpinti rengiamos programos kodą.

Skirtingi objektai gali turėti taip pat pavadintus parametrus ir metodus, tačiau į kreipimąsį reaguoti skirtingai. Šią objektų savybę vadina *polimorfizmu.* Tai labai naudinga modifikuojant programas.(8, 10)

2.4. Objektinio programavimo Action Script 2.0 kalba galimybės

Su kiekviena nauja Action Script versija, Flash kūrėjai stengėsi išleisti kuo patogesnę ir funkcionalesnę programavimo kalbą. Siekiant konkuruoti profesionalių interneto aplikacijų kūrime Macromedia patobulino programavimą Action Scriptu išleisdama naują Action Script 2.0 versiją. Programavimo kalba užtikrina funkcionalumą ir programavimo paprastumą. Visa programavimo kalba buvo restruktūrizuota, joje pridėta naujų standartų bei galimybių. (1, 8, 9)

Action Script panaši į tokias programavimo kalbas, kaip JavaScript 2.0 bei ECMAScript 4 proposal.(1)

Action Script redaktorius taip pat labai patobulintas. Paprastas kodo rašymas dėl žemo funkcionalumo buvo panaikintas, žodžių perkėlimas pagaliau įdiegtas redaktoriuje. (9)

Naujasis Action Script 2.0 klasių konstruktorius realizuotas prototipų kūrimu. Žinoma visas prototipavimas yra visiškai paslėptas nuo Action Script 2.0 programuotojų. Pagrindinė senojo prototipinio programavimo kliūtis ta, kad ji sunkiai išmokstama, nes ji neveikė pagal objektiškai orientuoto programavimo standartus. (8, 9)

Action Script 2.0 yra žymiai tikslesnė už ankstesnes versijas. Rašant kintamuosius būtina skirti didžiąsias ir mažąsias raides. Sumaišius didžiąsias ir mažąsias raides – kompiliatorius parodys klaidą. Protingas automatinis klaidų ieškojimas padės nesunkiai surasti netikslumus ir klaidas. (8)

```
var myProduct = "Call Center Server";
trace (myproduct);
```

Kintamųjų aprašymas gali būti konfliktiškas programuojant Action Script 2.0 programavimo kalba flash player 6.0 versijai, nes kintamieji gali konfliktuoti. Pavyzdžiui Flash grotuvo 7.0 versija interpretuos žemiau esantį kodą kaip du skirtingus kintamuosius, tačiau Flash grotuvo 6.0 versija – kaip vienodus. (8)

```
var myProduct = "Call Center Server";
var myproduct = "Call Center Client";
```

Šios klaidos dažniausiai pasirodo ne kompiliavimo, o programos vykdymo metu. Tuo atveju Flash programos kūrėjas pats atsakingas už klaidų taisymą.

Kiekviena Actin Script 2.0 klasė turi savo duomenų tipą , bei kiekvieną kartą deklaravus klasę būtina deklaruoti duomenų tipą. Kuriant objektą iš objekto klasės – naudojamas šis kodas:

```
myObject = new Object ();
```

Tačiau klaidų aptikimui bei kodo išskyrimui geriau būtų naudoti tokią sintaksę: (1, 4, 11)

```
Var myObject :Object = new Object ();
```

2.4.1 Duomenų tipai

Naudojami žemiau pateikti duomenų tipai. Visi jie išskyrus Void yra klasių vardai Action Script programavimo kalboje bei turi būti naudojami atsargiai.

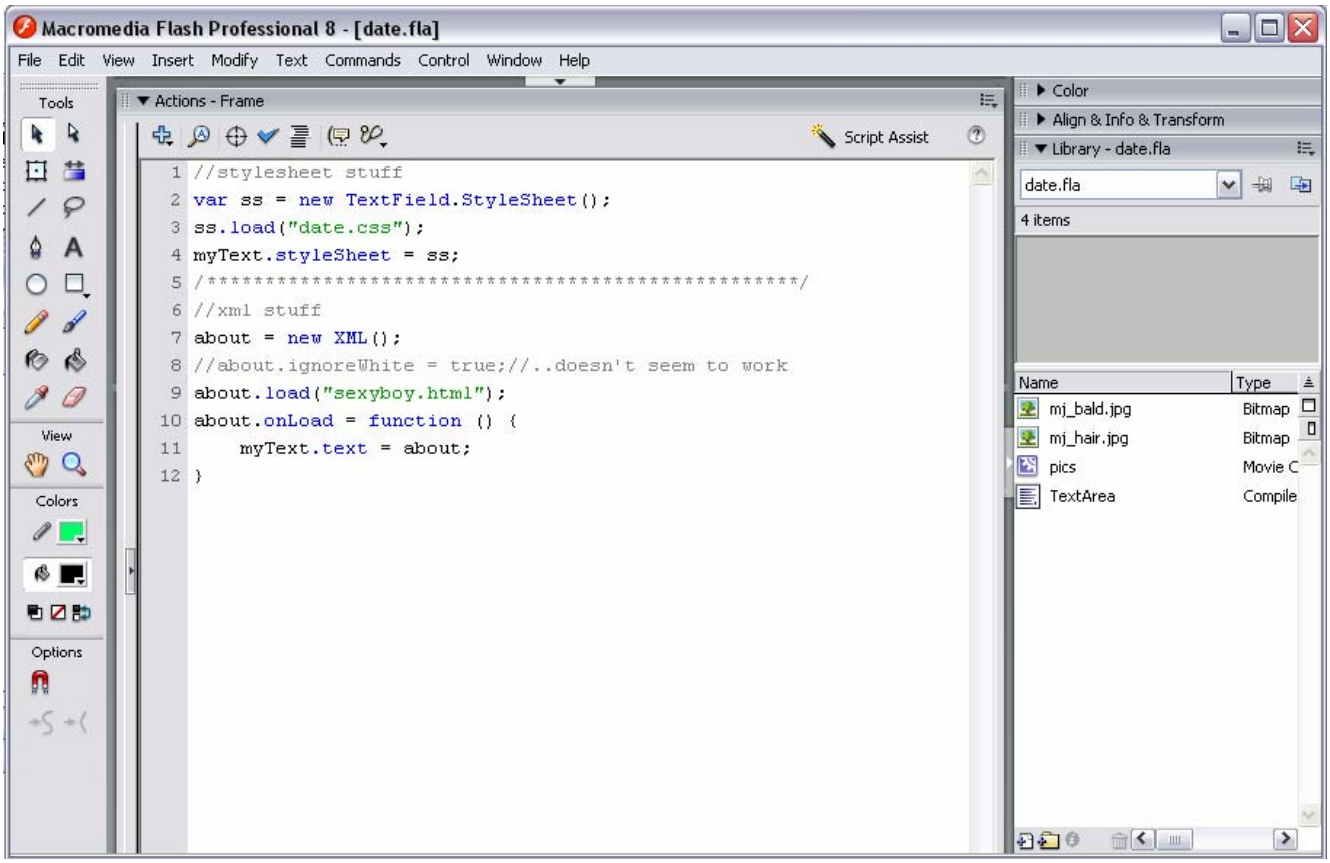
- `Array`
- `Boolean`
- `Button`
- `Color`
- `CustomActions`
- `Date`
- `Function`
- `LoadVars`

- `LocalConnection`
- `Microphone`
- `MovieClip`
- `NetConnection`
- `NetStream`
- `Number`
- `Object`
- `SharedObject`
- `Sound`
- `String`
- `TextField`
- `TextFormat`
- `Video`
- `Void`
- `XML`
- `XMLNode`
- `XMLSocket`

Geriausia yra deklaruoti kintamojo tipą, kurį funkcija gražina. Tai ypač padeda rasti klaidas, nes kompiliatorius gali parodyti tikslius klaidų pranešimus. (1)

2.4.2. ActionScript redaktorius

Flash MX Profesional 2004 sukūrė, o dabar ir Flash Profesional 8 patobulino ActionScript redaktorių. Kai koreguojamas ActionScript dokumentas, visos juostos tampa neaktyviomis, įskaitant laiko, įrankių juostas bei visus komponentus. Taip pat redaktoriuje yra įdiegtos visos kodo klaidų tikrinimo ir automatinio formavimo funkcijos. ActionScript redaktorius pavaizduotas 4 pav. (4, 11)



4. pav. Integruotas Flash Profesional 8 ActionScrit editorius (8)

2.4.3. Klasės

Klasių kūrimas Action Script priemonėmis leidžia kurti keičiamo dydžio sprendimus logiškai dėstant programos kodą. Action Script 2.0 turi visiškai naują priėjimą prie klasėmis pagrįsto programavimo, leidžiančio pasinaudoti praktinėmis žiniomis įgytomis programuojant Java, C ir kitomis OO kalbomis. (4)

Klasė yra metodų ir ypatybių rinkinys. Ypatybės apibrėžia duomenis, susijusius su klase, tuo tarpu metodai valdo duomenų srautus. Kartu jie apibrėžia objekto, sugeneruoto klasėje, galimybes. (4)

Action Script 2.0 programavimo kalboje klasė aprašoma naudojant class raktą, klasės vardą, bei kodo lauką, išskirtą skliausteliais. Pavyzdžiui, mums reikia aprašyti klasę, atspindinčią paskolą, studento paskolą ir namo paskolą. Kiekvienu atveju paskola skirsis dydžiu, žmonėmis bei terminu.

```
class Paskola {
            //Ypatybės
            dydis; //paskolos dydis
            rodiklis; //paskolos susidomėjimo rodiklis
            terminas; //paskolos terminas
            //metodai
            skaiciuotiMenesioMokescius (){
            }
}
```

Klasės vardas turi būti toks pat kaip išorinio failo, kuriame yra klasė, pavadinimas. Viskas, kas yra skliaustelių viduje – priklauso klasei. (4)

2.4.4. Ypatybės

Klasė gali turėti ypatybes, kurios yra kintamieji, apibrėžti klasėje. Jie naudojami specifiniams duomenims apie klasę saugoti. Jos dar vadinamos instancijos kintamaisiais.

Raktinis žodis *var* nurodo kintamojo tipą, po kurio seka dvitaškis bei duomenų tipas. Duomenų tipas rašomas atsižvelgiant į didžiąsias ir mažąsias raides.(4, 11)

```
var principal :Number = ”5.0”;
```

Paskolos klasė gali būti patobulinta ypatybėmis:

```
class Paskola {
    //ypatybės
    var dydis :Number; //paskolos dydis
    var rodiklis :Number; //susidomėjimo rodiklis
    var terminas :Number; //paskolos terminas
    var klientoVardas; //paskolos gavėjo vardas
}
```

2.4.5. Metodai

Metodai yra operacijos, kurios gali būti vykdomos su objektu. Jie gali paimti duomenis iš programos bei gražinti juos apdorotus klausytojui. Metodai taip pat yra vadinami funkcijomis. Jie apibrėžia operacijas, kurias gali vykdyti instancija. (4, 11)

```
class Paskola {
    //ypatybės
    var dydis :Number; //paskolos dydis
    var rodiklis :Number; //susidomėjimo rodiklis
    var terminas :Number; //paskolos terminas
    var klientoVardas; //paskolos gavėjo vardas
    //metodai
         function setPrincipal(thePrincipal:Number):Void{
               this.principal = thePrincipal;
         }
}
setPrincipal (5.5);
```

2.5. XML dokumentų struktūra bei galimybės

XML (angl. „eXtensible Markup Language“) yra „žymėjimo“ kalba dokumentams, kuriuose saugoma struktūrizuota tekstinė informacija. W3C patvirtino XML standartą 1998 metais. Iš esmės tai — seniai pramonėje vartojamos kalbos SGML poaibis, specialiai pritaikytas naudojimui internete. Todėl kiekvienas XML dokumentas yra ir SGML dokumentas, bet ne atvirkščiai. (12)

Pagrindinė XML kalbos paskirtis yra užtikrinti lengvesnį duomenų keitimąsi tarp skirtingo tipo sistemų, dažniausiai sujungtų internetu. XML, tai protokolas saugantis ir valdantis informaciją. Tai technologija, kuri gali daryti viską, nuo dokumentų formatavimo iki duomenų filtravimo.(12)

XML yra programavimo kalba, panaši į xHTML. XML tikslas yra kitoks nei xHTML kalbos. XML orientuojasi į informaciją bei jos talpinimą, o xHTML — į informacijos išvedimą.

Siekiant aprašyti informaciją XML dokumentuose naudojama DTD dokumentų tipo aprašymo schema. Turint XML dokumentą bei jo schemą duomenys yra interpretuojami. Schema lyg žemėlapis, aprašantis duomenų struktūrą. XML naudojamas saugoti bei siųsti duomenis, tuo tarpu schema lieka nepakitusi. (12, 13)

Pagrindinis XML kalbos vienetas yra *elementas*. Elementas visada turi vardą ir, be jo, gali turėti:

- Norimą skaičių atributų. Atributas turi savo vardą bei reikšmę.
- Kitus (dukterinius) šio elemento viduje esančius elementus.
- Su elementu susijusį tekstą.

2.5.1. Žymės ir elementai

Norima struktūra išgaunama naudojantis *žymėmis* (angl. „tags“). XML standartas nustato bendras jų sudarymo taisykles, bet neapibrėžia konkretaus žymių rinkinio ar jų reikšmės. Tai apibrėžia kiti XML šeimos standartai (žr. XHTML, XSL ir t. t.). XML galima pavadinti „kalba kitoms kalboms aprašyti“. (12, 13)

Kiekviena žymė prasideda simboliu < ir baigiasi simboliu >. Jų viduje užrašomas žymės vardas ir (galbūt) papildoma informacija, pvz. atributai ar vardų sritys. Žymių vardai yra „case-sensitive“. Yra pradžios bei pabaigos žymės. Pabaigos žymėje prieš jos vardą rašomas simbolis/. Pradžios bei pabaigos žymių apgaubtas turinys vadinamas *elementu*. Elemento viduje taip pat gali būti kiti elementai. Pvz:

```
<žymė>
  Tekstas 1
  <vidinė_žymė>
    Tekstas 2
  </vidinė_žymė>
</žymė>
```

Tuščias elementas gali būti sutrumpintai užrašomas nenaudojant jo pabaigos žymės: `<tuščias_elementas/>`.

Tai ekvivalentu tokiam užrašui: `<tuščias_elementas></tuščias_elementas>`.

Kaip galbūt jau supratote, elementai negali persidengti, t.y. negali būti tokio pavidalo struktūrų: `<1_lementas>...<2_elementas>...</1_elementas>...</2_elementas>`. Tai buvo leidžiama HTML standarte. (13)

Aukščiausio lygio (šakninis) elementas XML dokumente gali būti tik vienas.

2.5.2. Atributai

XML elementas gali turėti *atributus* (angl. „attributes“). Jie užrašomi forma `atributo_vardas="atributo_reikšmė"` arba `atributo_vardas='atributo_reikšmė'`. Atributų vardai yra „case-sensitive“. Atributo reikšmė gali būti tik tekstinė. Atributai talpinami elemento pradžios žymėje, po jos vardo, pvz.: `<studentas pažymėjimo_nr="12345678">…</studentas>`. Elementas negali turėti dviejų atributų vienodais vardais. (12, 13)

XML atributai neturi sutrumpintos formos, kokia buvo leidžiama HTML standarte. Pvz. vietoje `<optikon selected>...</option>` reikia rašyti `<optikon selected="selected"> ... </optikon>`.

Kai kurie atributai (pvz. `xml:lang`, nurodantis elemento kalbą) yra rezervuoti paties XML standarto. T.y., jie turi tą pačią reikšmę visuose XML dokumentuose ir negali būti autorių naudojami saviems tikslams.

Atributuose paprastai saugoma papildoma informacija apie elementą, tačiau kartais atributo reikšmė gali pakeisti paties elemento reikšmę (jei ji tėra tekstinė). Palyginimui: (12, 13)

```
<studentas>
  <vardas>Petras</vardas>
  <pavardė>Petraitis</pavardė>
  <adresas>
    <namo_nr>13</namo_nr>
    ...
  </adresas>
  ...
</studentas>
```

ir

```
<studentas vardas="Petras" pavardė="Petraitis">
  <adresas>
    <namas nr="13"/>
    ...
  </adresas>
```

```
  ...
</studentas>
```

2.5.3. Elementas ar atributas?

Ką pasirinkti saugoti tekstinei reikšmei — elementą ar atributą? Vienareikšmio atsakymo tikriausiai nėra. Atributai tėra papildoma priemonė, tačiau kartais patogi ir net būtina.

Galima taikyti tokį metodą — įsivaizduoti, kad elementas yra dėžė. Tada visi jos viduje esantys daiktai turėtų būti vidiniai elementai, o užrašai ant dėžės — atributai.(12)

Jei turime elementą `<automobilis>`, jo vidiniai elementai galėtų būti `<kėbulas>`, `<variklis>`, o atributai — `pagaminimo_data`, `spalva` ir t. t.

Tačiau jei kuriame elementą `<automobilio_registracija>`, galėtume naudoti vidinius elementus `<pagaminimo_data>` ir `<spalva>`. Tokiu atveju `registracijos_data` turėtų būti atributas, nes jame saugoma informacija apie automobilio registraciją, o ne apie patį automobilį.

2.5.4. Vardų sritys

Vardų sritys („namespaces“) naudojamos pažymėti, kokiam rinkiniui priklauso elementas ar atributas.

XML standartas leidžia savo dokumente (formate) naudoti elementus ir atributus su pavadinimais, kurios turi konkrečią kuriuo nors standartu (pvz., XHTML) apibrėžtą reikšmę. Taigi pvz. žymės `<p>` panaudojimas tokiame dokumente būtų leistinas. Tame kontekste šios žymės reikšmė gali būti bet kokia (sugalvota autoriaus). Tačiau bandant tokį dokumentą (jo fragmentą) įterpti į XHTML dokumentą, kiltų problemų. Naršyklė, „manydama“, kad žymė `<p>` priklauso XHTML, interpretuotų ją kaip „pragrafo pradžią“, kas autoriui nėra priimtina. Tokiu situacijoje naudojamos vardų sritys.(13)

Elementai, kurias norima priskirti kuriai nors vardų sričiai, turi turėti specialų prefiksą. Tokia žymė užrašoma taip: `<prefiksas:žymės_vardas>`.

Prefiksas turi būti prieš tai apibrėžtas dokumente (bet kuriame elemente) tokiu atributu: `prefiksas:xmlns="URI"`. Čia URI — interneto resurso identifikatorius (dažniausiai URL), pagal kurį galima identifikuoti vardų sritį.(12, 13)

Atributas `xmlns="URI"` gali būti panaudotas ir be prefikso. Tokiu atveju vardų sritis, kurią identifikuoja nurodytas URI, bus pagrindinė. T. y., jai priklausys visi po šio atributo dokumente esantys elementai, neturintys prefikso.

Taigi aukščiau minėtas pavyzdys galėtų atrodyti taip:

```
<html xmlns="http://www.w3.org/1999/xhtml"
petro:xmlns="http://www.petras.lt/xml">
  ...
```

```
  <body>
    <h1>Antraštė ... </h1>
    <p>Paragrafas ... </p>
    ...
    <petro:p petro:atributas"reikšmė">Petro elementas</petro:p>
    ...
    <p>Paragrafas ... </p>
  </body>
</html>
```

Tai pačiai vardų sričiai galima naudoti kelis skirtingus prefiksus.

2.5.5. Intarpai

Intarpai (angl. „entities“) naudojami įterpiant į XML dokumentą specialų simbolį, tekstą ar failą. Jie aprašomi XML intarpų apibrėžime. Dokumente užrašomi tokia forma: `&vardas;` arba `&#kodas;`. Čia kodas — tai simbolio kodas, o vardas — tai sinoniminis intarpo vardas, kuris gali būti nurodomas intarpų apibrėžime.(12)

Pvz., intarpai `&bdquo;` ir `&ldquo;` įterpia į dokumentą lietuviškas kabutes „ ir “; `&` — ampersandą & (kuris naudojamas ir pačių intarpų užrašymui); `&copy;` — simbolį ©.

2.5.6. Specialios žymės

XML formatas apibrėžia keletą *speciąlių žymių*, kurios bendros visiems XML dokumentams.

- *XML dokumento antraštė*: `<?xml version="versija" encoding="koduotė"?>`. Čia `versija` — XML versija (kol kas `1.0`), `koduotė` — kodų lentelės pavadinimas (rekomenduoju UTF-8 — tarptautinis standartas).
- *vykdymo instrukcijos* (angl. „processing instructions“), pvz.: `<?xml-stylesheet type="text/xsl" href="stilius.xsl"?>` (nurodo XML dokumento stiliaus failą)
- *komentarai*: `<!-- komentaras -->`
- *CDATA*: `<![CDATA[ turinys ]]>`. Tarp šių žymių turi būti rašoma viskas, ko nereikia interpretuoti kaip XML — skriptų kodas, CSS stiliai, binariniai failai ir t. t. Pvz.:

```
<style type="text/css">
  <![CDATA[
    html { font-family: "Tahoma", "Verdana", sans-serif; }
    body > p { line-height: 1cm; } /* galioja visiems <p>, kurie yra
<body> „child'ai“ */
  ]]>
</style>
```

- *dokumento tipo apibrėžimas* (angl. „document type definition“). Jis nurodo, kokią struktūrą dokumentas turi atitikti, t.y. kokie elementai ir atributai jame leistini, kaip jie turi būti sukomponuoti ir t. t. Gali būti ir išoriniame faile. Pastaruoju metu pereinama prie XML Schema dokumentų tipo apibrėžimo. (12, 13)
- *intarpų apibrėžimas* (angl. „entity declaration“).

2.5.7. XML dokumentas

XML *dokumentas* gali prasidėti antrašte. Po jos gali eiti vykdymo instrukcijos, dokumento tipo ir intarpų apibrėžimai.

Po šio pradžios bloko seka esminis dokumento turinys — elementai.

XML dokumentas gali būti gerai suformuotas („well formed“) ir teisingas („valid“).

Gerai suformuotas dokumentas — tai XML sintaksę atitinkantis dokumentas. T. y., jis turi turėti vieną (ir tik vieną) aukščiausio lygio (šakninį) elementą, jo elementai turi būti teisingai sukomponuoti ir t. t.(13)

Teisingas dokumentas — tai savo tipo apibrėžimą atitinkantis dokumentas (žinoma, jis turi būti ir gerai suformuotas).

Komentarai gali būti bet kur dokumente, o CDATA blokai — ten, kur gali būti elementai.

Visi specialūs XML simboliai, esantys tekste, o *ne* žymėjime (*ne* žymėse, atributuose, CDATA blokuose ir t. t.), turi būti pakeisti intarpais.(12)

Radusi klaidą, XML dokumentą apdorojanti programa turi pranešti apie klaidą ir nutraukti apdorojimą — skirtingai nuo HTML, kur klaidas buvo leidžiama ignoruoti.

XML dokumento struktūra kartais skirstoma į „orientuotą į duomenis“ ir „orientuotą į dokumentą“.(12)

2.5.8. XML žodynai

Vartotojai kuria XML žodynus. Žodynai gali būti formalizuoti ir publikuojami ir naudotojams, ir XML procesoriams, arba gali būti užslėptas dokumente.

Akivaizdu, jog be išorinio žodyno XML netikrina sintaksės. Kai žodynas formalizuotas DTD schemoje, galima naudoti šį dokumentą XML ir jo techninių žodyno taisyklių vientisumo patvirtinimui. Taip pat ir su formaliu DTD žodymu negalima patikrinti sintaksės. Visi duomenys tarp pradžios ir pabaigos yra tik teksto grandinė — XML žymės. (5)

2.5.9. XML tikrinimo schema

XML gali apibūdinti beveik bet kurią informaciją. Tai suteikia didžiulę lankstumo galimybę, nes niekas nėra apibrėžta. Elementų vardai, kuriuos naudojame, neturi esminių reikšmių XML standarte. Mes juos grupuojame taip, kad apibūdintume savo informaciją.

Čia nėra taisyklių kaip kiekvienas dokumento elementas sistemina savo komponentų elementus arba kaip tie elementai yra tarpusavyje sukomponuoti.

Gamintojas ir vartotojas turi sutikti su XML dokumento struktūra ir kiekvienu jos elementu. Buvo siūlomi keli duomenų formalizavimo variantai. Originalus ir paprasčiausias yra Document Type Definition (DTD) — Dokumento Tipo Paskirtis. Flash XML skaitytojai (parser) neskaito DTD tiesiogiai.(5)

Reikia pabrėžti, kad Flash skaitytojas (parser) nėra valiuojantis skaitytojas (parser). Flash skaitytojas neatidaro DTD dokumento ir nenaudoja jo turinio. Taigi DTD turime naudoti tam, kad patikrintume XML — ne atsižvelgus į XML standartą, bet dėl XML sąsajos su aukštesnio standarto aplikacijos patvirtinimu. Norminis DTD sudaro puikią galimybę skelbti formatą, kuriame priimame duomenis. Leidėjai ir vartotojai ne tik gali išreikšti šiuos formatus, bet lengvai juos testuoti. (5)

2.5.10. XML Schemos problemos

Pagrindinės XML Schemos problemos:

- XML Schema yra labai komplikuota (specifikacija užima kelis šimtus lapų kiekvienoje techninėje kalboje), taigi sudėtinga naudoti be ekspertų, bet daugeliui nepatyrusių specialistų schema reikalinga duomenų formatų apibūdinimui. Taip pat painus dizainas reikalauja nesuvokiamo stiliaus specifikacijos.
- Praktiniai išreiškimo trūkumai: (13)
    1. Negalima reikalauti specifinio **šakninio elemento** (root element – tai ekstra informacija reikalaujanti patvirtinti netgi paprasčiausius dokumentus).
    2. Apibūdinant bendrą turinį, charakteringi duomenys negali būti apibrėžti.
        1. Turinys ir atributų deklaravimai negali priklausyti nuo atributų ar elementų konteksto (tai buvo traktuojama kaip ir DTD problema).
- Numatyti elementai negali specifikuotai atskirti nuo deklaruotų (tai apsunkina gimininingų schemų sudarymą).
- Nustatyti elementai gali būti charakteringi duomenys.
- Techninės problemos:
- Sąvoka „type“ prideda papildomą painumą:
    3. dokumentuose yra elementai, kurie turi „elemento vardą“.
    4. Schemose elementai yra apibūdinami kaip „elemento apibrėžimai“, kurie asocijuoja „Elemento vardą“ su „Elemento tipu“.
    5. Tipo apibrėžimas sujungia „tipo vardą“ su „elemento aprašymu“, kuris apibūdina dokumento elementus.
- *Xsi:type* atributai yra būtini dokumentuose kai išvesti tipai yra naudojami bendrame tipe.
- Pakeičiamos grupės ir lokalūs paskelbimai (su nepriklausomais vardais) apsunkina duotojo elemento apibrėžimo paiešką.

Ne minimalistinis dizainas:

- Tipo kilmės ir pakaitalas grupių mechanizmas atrodo kitaip mėgina spręsti tas pačias problemas.
- *XPath* susivienijimas išreiškia unikalumą ir raktus (taip pat unikalumas ir raktai yra pagrindė koncepcija schemoms).
- Įdiegtų duomenų tipų rinkinys ne minimalistinis.
- *Perl* tipo reguliarių ekspresijų naudojimas pažeidžia XML sintaksės apibūdinti XML sintaksę principus. (13)

2.5.11. XML panaudojimas

XML sparčiai plintanti ir besivystanti duomenų atvaizdavimo ir aprašymo kalba.

Kaip jau minėta, leidžia patogiai saugoti struktūrizuotą tekstinę informaciją ir pateikti ją internete. (12)

XML privalumai: tai paprastas, išplečiamas, nuo konkrečios OS ar programavimo kalbos nepriklausomas, tarptautiniam naudojimui pritaikytas, atviras ir nemokamas standartas.

XML (ir tam tikru dokumento tipo apibrėžimu) daugeliu atveju galima patogiai pakeisti atgyvenusius tekstinius ar binarinius formatus (protokolus). XML duomenimis galima lengvai manipuliuoti programose, naudoti juos duomenų bazėje ar net vietoj jos.(12,13)

Vienas didžiausių XML privalumų — jį galima transformuoti. Tiems patiems duomenims pritaikius skirtingas transformacijas, galima gauti keleto (galbūt visiškai skirtingų) pavidalų rezultatus. Pvz., tam tikros apibrėžtos struktūros XML failuose saugomi WWW serverio „logai“ gali būti viena XSLT transformacija pervesti į visai kitokios struktūros XHTML dokumentą internetui, kita — į SVG grafiką, trečia — į WML dokumentą, skirtą mobiliesiems telefonams. Beje, XSLT transformacijos taip pat yra XML dokumentai.

Privalumai:

- Lengvai transformuojama į kitus duomenų formatus.
- Lankstus apsikeitimas duomenimis naudojant metaduomenis, integracija su kitais šaltiniais, išplėstos paieškos galimybės, filtravimo galimybės, automatinė validacija, mažu ir pakartotinai panaudojamų fragmentų valdymas bei turtingos atvaizdavimo galimybės.(7)
- Priėmus XML kaip duomenų apsikeitimo standartą tarp vartotojo ir portalo, galima būtų priartėti prie vieningos duomenų struktūros tarp išorinių verslo sistemų teikiamų paslaugų sistemų. Kuriama platforma turėtų palaikyti ir valdyti XML turinio duomenis verslo procesų kontekste, turėti integruotą XSLT (transformacijų variklį) bei galėti saugoti neribotą skaičių XSL (Extensible Stylesheet language).(12)

2.5.12. XML ir Flash

XML gali būti naudojamas duomenų skaitymui iš serverio ir rašymui į serverį,

Flash XML teikia pirmenybę XML objektui. Šis objektas užkrauna XML, transformuoja XML į ActionSdript suprantamą duomenų tipą ir suteikia metodams galimybę valdyti duomenis. XML objektas taip pat suteikia galimybę siųsti XML į serverį. Tai ypač naudinga, kai siunčiami duomenys į serverį, kuriame naudojamas saugus susijungimas, kaip pvz: slaptažodžiai, ar kai naudojami serverio pusės skriptai kartu su xml. XML leidžia mums skaityti ir rašyti į serverį be jokių papildomų duomenų pasiekimo komponentų.(5, 6)

Vykdymas ir duomenų krovimas. Kad panaudoti XML dokumentą Flash aplinkoje, pirmiausia dokumentas turi būti parsiųstas ir išanalizuotas, tik tada gali būti panaudotas. Dideli XML dokumentai gali būti vėlinimo priežastis aplikacijoje. Kraunant tą pačią informaciją iš tekstinio dokumento bus greitesni, nes reikalaujama mažai arba nereikalauja išvis jokios analizės. Po to, kai tekstiniai dokumentai užkrauti, duomenys tampa pasiekiami. Kitas pasirinkimas yra suskirstyti XML i mažesnius dokumentus ir užkrauti juos palaipsniui.(5, 6)

Krovimas XML duomenų sukurtų kitai aplikacijai. Jeigu XML duomenys yra sukurti kitai aplikacijai, bandant sujungti juos su Flash aplinka, gali kainuoti pernelyg sudėtingos analizės.

Serverių ir duomenų bazių krovimas. Jeigu duomenis reikalinga apdoroti nedelsiant, XML reikėtų generuoti kiekvienai užklausai. Jeigu generuojant tiesioginius duomenis kiekvienai užklausai kainuoja serverio darbo resursus, tada reikia generuoti XML periodiškai (kas valandą ar du kart per dieną).(5, 6)

2.5.13. XML ir Java

Informacijos iš XML dokumentų skaitymas yra gana sudėtingas uždavinys. Laimei nebūtina visko atlikti pačiam. Galima pasinaudoti XML „parseriu“, kuris nuskaito dokumentą. XML „parseris“ tyrinėja XML dokumento turinį per API. Klieno programa nuskaito XML dokumentą per šią API. Programėlė taip pat patikrina XML dokumento teisingumą ir suradusi klaidą praneša kliento aplikacijai. (7)

XML skaitytojai (parsers)

Siekiant išvengti sunkumų nuskaitant informaciją iš XML, beveik visos programos, dirbančios su xml dokumentais naudoja XML skaitytoją (parser) dokumento nuskaitymui. Tai programinė biblioteka (Java klasėje), kuri nuskaito XML dokumentą ir patikrina klaidas. Kliento programa naudoja metodų iškvietimą, numatytą tyrinėtojo API užklaustos informacijos gavimui. (14)

XML API pasirinkimas

Svarbiausias sprendimas kuriant XML projektą – tai programavimo interfeiso (API) pasirinkimas. Jei naudojamas skaitytojas negali atlikti užduoto darbo – galima pakeisti ji dažnai net be programos perkompiliavimo. Kita vertus priklausomai nuo pasirinkto API gali tekti netgi perrašyti visą kodą naujai. (7)

Yra du pagrindiniai API standartai XML dokumentų nuskaitymui Java programavimo kalboje. Paprastas API (SAX) bei dokumento objektinis modelis (DOM).

SAX – paprastas, dar vadinamas auksiniu XML API. Jis yra labiausiai išbaigtas ir korektiškiausiai veikiantis. Su šiuo interfeisu galima nuveikti beveik visus XML nuskaitymo darbus. SAX veikia labai greitai bei labai taupiai naudoja atmintį (nesaugo viso dokumento atmintyje). Tačiau SAX programos gali būti sunkiau projektuojamos bei rašomos, nes programuotojui tenka pačiam sukurti duomenų struktūras, saugančias XML dokumento duomenis. (7,14)

DOM (Document Object Model) yra gana sudėtingas API, modeliuojantis XML dokumentą kaip medį. Skirtingai nuo SAX, DOM yra skaitymo-rašymo API. Jis gali ir tyrinėti (nuskaityti) ir sukurti XML dokumentą. Kiekvienas XML dokumentas yra reprezentuojamas kaip objektas. Dokumentai yra ieškomi, eiliuojami bei atnaujinami iškviečiant metodus konkrečiam dokumento objektui bei objektams, kuriuos ji turi. Dėl to DOM yra daug naudingesnis kai reikalingos labai išsimėčiusios XML dokumento dalys. Tačiau lyginant su SAX šis API yra reiklesnis atminčiai, bei nėra efektyvus nuosekliam duomenų skaitymui. (7)

JDOM yra paprastas JAVA medžio struktūros interfeisas, skirtas pašalinti nemažai su DOM susijusių sunkumų. Kaip ir DOM, JDOM nuskaito visą dokumentą į atmintį prieš pradėdamas

su juo dirbti. Programos kodas darbui su JDOM yra labai panašus kaip ir su DOM. Tačiau žemo lygmens kodas yra žymiai paprastesnis. JDOM naudoja realias klases bei konstruktorius, tuo tarpu DOM naudoja interfeisus bei metodus. (7)

XML skaitymo (*parser*) pasirinkimas

Skaitytojai gali būti padalinti į tris kategorijas:

- Pilnai tikrinantys skaitytojai.
- Skaitytojai netikrinantys dokumento, bet nuskaitantys DTD (dokumento tipo paskirtis).
- Skaitytojai, skaitantys tik DTD poaibį, bet nevaliduojantys.(7)

Praktikoje yra ir ketvirta kategorija, nuskaitanti dokumentą tačiau neatliekanti jokių formatavimo patikrų. Techniškai šie skaitytojai neleidžiami pačios XML specifikacijos, tačiau tokių yra daug.(7)

Jei naudojamas dokumentas turi DTD, tuomet reikia naudoti pilnai tikrinantį skaitytoją. Tačiau nebūtina įjungti tikrinimo jei nėra reikalo. Anaiptol XML konstruotas taip, kad neįmanoma būti tikram, jog gauname visą XML turinį neperskaitę jo DTD.

Kai kurie skaitytojai pateikia papildomos informacijos, nereikalingos normaliam XML nuskaitymui.(13)

API palaikymas

Dažniausiai skaitytojai (*parser*) palaiko SAX ir DOM programų programavimo interfeisus, tačiau yra keli palaikantys tik SAX. Todėl pasirinkdami skaitytoją turime atsižvelgti į tai, kokį interfeisą naudosime.(14)

SAX dažniausiai turi daug papildomų funkcijų, kurios skaitytojoms yra nebūtinos. Šios funkcijos gali būti tikrinimas, komentarų ataskaitos, išsami informacija apie DTD bei kitos. (14)

Efektyvumas

Vienas iš skaitytojo pasirinkimo kriterijų yra jo greitaveika bei atminties naudojimas. Skaitant XML dokumentą iš tinklo greičiausiai sistemą stabdys nedidelis tinklo pralaidumas, tačiau retais atvejais sistemos greitaveika gali padidėti pakeitus XML skaitytoją.

Skaitant duomenis iš disko ar tinklo būtina greitaveikos sąlyga duomenų buferiavimas. Galima buferiuoti baitus naudojant *BufferedInputStream* arba simbolius su *BufferedReader*. Tačiau dauguma skaitytojų veikia geriau siunčiant jiems baitus konvertavimui į simbolius (skaitytojai dažniausiai geriau atlieka šį darbą nei kliento kodas). Todėl rekomenduojama naudoti *BufferedInputStream*, taip gaunant užtikrintą simbolių konvertavimą.(14)

Populiariausi skaitytojai

Xerces-J – tai skaitytojas iš Apache XML projekto. Tai yra labai išbaigtas valiuojantis skaitytojas, atitinkantis XML 1.0 standartą. Jis pilnai palaiko SAX2 bei DOM antrojo lygio API. Taip pat palaikomas JAXP. (7)

Šis skaitytojas yra labai lanksčiai konfigūruojamas bei tinkamas beveik visiems skaitymo darbams. Šis skaitytojas vienintelis, palaikantis W3C XML kalbą, tačiau palaikymas nėra 100% išbaigtas.(7)

Crimson anksčiau buvo žinomas Java Project X vardu. Jis yra kartu su JDK 1.4 versija. Crimson palaiko beveik tokias pačias programavimo aplinkas, kaip ir Xerdes (SAX2, DOM2, JAXP, XML 1.0). (7)

Piccolo – vėliausiai sukurtas skaitytojas (*parsek*)**.** Pasižymi mažu dydžiu, labai greitu veikimu, bei yra atviro kodo (open source) programinė įranga, netikrinanti XML dokumentų bet skaitanti DTD posistemę. Skaitytojas palaiko tik SAX API.(7)

2.6. Išvados

1. Internetinių aplikacijų kūrime Action Script 2.0 programavimo kalba suteikia daugiau galimybių nei Visual Basic, C ar net Java. Programavimas Action Script 2.0 tapo labai patogus, paprastas bei atliekamas pagal objektiškai orientuoto programavimo principus.
2. Action Script tapo panaši į tokias programavimo kalbas, kaip JavaScript 2.0 bei ECMAScript 4 proposal.
3. Naujasis Action Script 2.0 klasių konstruktorius realizuotas prototipų kūrimu, tačiau prototipų kūrimas yra paslėptas nuo programuotojo, o programuojama pagal objektinio programavimo taisykles.
4. Flash MX 2004 teikia pirmenybę XML objektui. Šis objektas užkrauna XML, transformuoja XML į ActionSdript suprantamą duomenų tipą ir suteikia metodams galimybę valdyti duomenis.
5. XML - Sparčiai plintanti ir besivystanti duomenų atvaizdavimo ir aprašymo kalba. Lengvai transformuojama į kitus duomenų formatus. Lankstus apsikeitimas duomenimis naudojant metaduomenis, integracija su kitais šaltiniais, išplėstos paieškos galimybės, filtravimo galimybės, automatinė validacija, mažu ir pakartotinai panaudojamų fragmentų valdymas bei turtingos atvaizdavimo galimybės.

3. Sistemos projektas

3.1. Funkciniai sistemos reikalavimai ir reikalavimai duomenims

3.1.1. Sistemos paskirtis

Sukurta sistema yra skirta visiems vartotojams, kurie domisi siuvimu bei ieško įvairios informacijos internete. Tai informacinė sistema veikianti internete. Sukurtos sistemos paskirtis — pateikti vartotojui siuvimo modelių ir jų iškarpų pasirinkimo galimybę su aprašymais bei išsamiais paaiškinimais (mokomąja medžiaga).

Projekto kūrimo pagrindas. Atlikus išsamią tinklalapių analizę panašių informacinių tinklalapių neradau. Apie tokios informacijos populiarumą byloja gana nemažas „Burda“ ir panašių žurnalų populiarumas tarp siuvėjų.

Sistemos tikslas. Sistema turi pasižymėti labai paprastu valdymu, intuityviai suvokiamu meniu bei paprastu informacijos pasiekiamumu. Taip pat turi būti užtikrinta apsauga nuo teksto bei paveiksliukų kopijavimo užtikrinant pateiktos informacijos apsaugą nuo kopijavimo. Sistema kuriama naudojantis naujausiomis Flash MX ActionScript bei XML technologijomis, užtikrinančiomis puikų sistemos veikimą.

Suinteresuoti asmenys.

Sistemos užsakovas — Kauno technologijos universiteto informatikos fakultetas.

Vartotojai — pradedančios siuvėjos bei visi, besidomintys šiuo darbu.

Sprendimus priimantys asmenys:

Simona Barauskaitė IFM-0/3 studentė atsakinga už sistemos projektavimą, kūrimą bei testavimą.

Antanas Lankevičius – projekto vadovas, padedantis projektuotojai sukurti atitinkančią visus reikalavimus sistemą.

Žmonės, atsakingi už sistemos testavimą: Simona Bartauskaitė, Renaldas Zajančkauskas.

Sistemos vartotojai. Sistema naudosis įvairų išsilavinimą turintys vartotojai, todėl ji turi būti ypač paprasta vartoti bei lengvai suprantama.

3.1.2. Projekto apribojimai

Apribojimai sprendimui.

- Sukurta sistema turi būti ypač paprasta naudojimui.
- Visi meniu punktai ir pasirinkimai turi būti intuityviai suvokiami bei paprastai pasiekiami.
- Sistema turi būti stabili.
- Sistema turi turėti apsaugą nuo teksto bei paveiksliukų kopijavimo.
- Sistema turi būti pasiekiama internetu.

Sistemos kūrimo biudžetas.

Projektas bus kuriamas savo lėšomis. Sistemos kūrimo periodu naudosiuosi Macromedia Flash MX kompiliatorius bei NotePad, arba WordPad programa programos tekstui rašyti. Programa bus kompiliuojama bandomąja Flash MX Pro versija.

Kuriama programinė įranga bus pateikta kaip atviro kodo programinė įranga visiems norintiems ja naudotis.

Veiklos padalinimas.

Lentelė 1. Veiklos įvykių sąrašas

Eil. Nr.	**Įvykio pavadinimas**	**Įeinantys/Išeinantys informacijos srautai**
1	Administratorius talpina tekstinę informaciją XML formatu bei duomenys į internetą pateikiami kartu su nuotraukomis.	Informacijos talpinimas į sistemą
2	Vartotojas interneto naršyklėje surenka tinklapio adresą	Sistemos pasiekimas
3	Vartotojas meniu punkte pasirenka norimą meniu	Meniu pasirinkimas
4	Vartotojas pasirenka straipsnį ar norimą iškarpą	Informacijos pasiekimas

3.1.3. Sistemos eksploatavimo aplinka

Diegimo aplinka.

Sistema neturi priklausyti nuo konkrečios platformos. Ji privalo puikiai veikti tiek su Apple tiek su IBM PC tipo kompiuteriais.

Sistema privalo dirbti grafinėje interneto naršyklėje bei turi būti pasiekiama internetu.

Bendradarbiaujančios sistemos.

Sistema turi būti patalpinta į interneto serverį ir iš jo pasiekiama internetu.

Sistema bendradarbiaus su interneto naršykle, kuri privalo turėti nemokamą „Shockwave Flash„ priedą. Pati sistema bus vykdoma Flash MX aplinkoje, nepriklausančioje nuo platformos aplinkos.

Komerciniai specializuoti programų paketai.

Komerciniai specializuoti programų paketai nenaudojami.

Numatoma darbo vietos aplinka.

Numatomai darbo vietai specialių reikalavimų nėra. Fizininės darbo vietos charakteristikos atitinka visų asmeninio kompiuterio darbo vietos charakteristikas.

3.1.4. Vartotojo sąsajos reikalavimai

1. Visos funkcijos turi būti paaiškintos aiškinamaisiais užrašais.
2. Valdymo mygtukai turi būti logiškai išdėstyti bei lengvai pasiekiami.
3. Kai kurios funkcijos turi būti pasiekiamos klaviatūra.
4. Galimybė dirbti tiesiog interneto puslapyje.
5. Suderinamumas su visomis OS.

3.1.5. Funkciniai sistemos reikalavimai

Sistema yra informacinio pobūdžio. Sistema XML faile saugo tekstinius duomenis, kurie yra užkraunami kartu su paveikslais. Sistema turi veikti internete, sistemos funkcijos turi būti valdomos pele, duomenys turi būti saugomi XML faile, tekstas turi būti apsaugotas nuo kopijavimo.

Lentelė 2. funkcinis reikalavimas,

Reikalavimas #:	1	Reikalavimo tipas:	9.1	Įvykis / panaudojimo atvejis #:	3-4
Aprašymas:	Funkcijos valdomos pele				
Pagrindimas:	Valdymo paprastumas				
Šaltinis:	Vartotojas				
Tikimo kriterijus:	Paprastumas				
Priklausomybės	Nėra	Konfliktai:			Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2005 spalio 14d.				

Lentelė 3. funkcinis reikalavimas.

Reikalavimas #:	2	Reikalavimo tipas:	9.1	Įvykis / panaudojimo atvejis #:	3-4
Aprašymas:	Tekstas apsaugotas nuo kopijavimo				
Pagrindimas:	Sistema turi būti saugi				
Šaltinis:	Vartotojas				
Tikimo kriterijus:	Apsauga.				
Priklausomybės	Nėra	Konfliktai:			Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2005 spalio 14d.				

Lentelė 4. funkcinis reikalavimas.

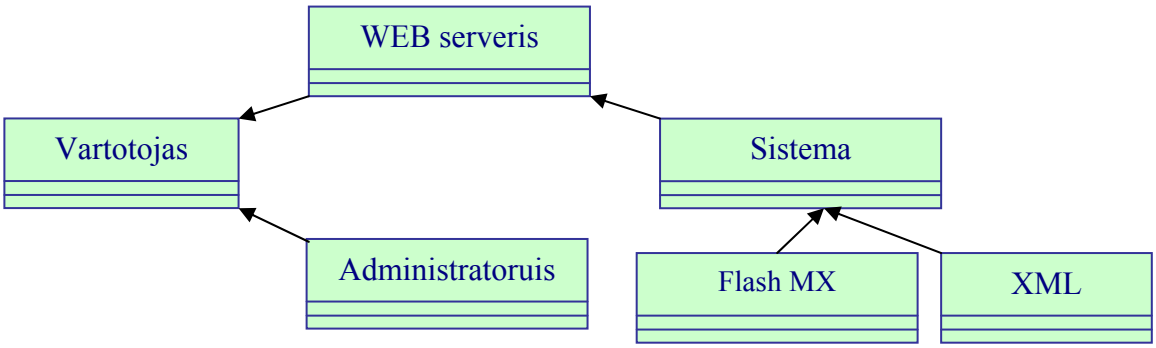
Reikalavimas #:	3	Reikalavimo tipas:	9.1	Įvykis / panaudojimo atvejis #:	1-4
Aprašymas:	Sistema turi veikti internete				
Pagrindimas:	Suderinamumas su visomis platformomis				
Šaltinis:	Vartotojas				
Tikimo kriterijus:	Suderinamumas				

Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:	Nėra		
Istorija:	Užregistruotas 2005 spalio 14d.		

Lentelė 5. funkcinis reikalavimas.

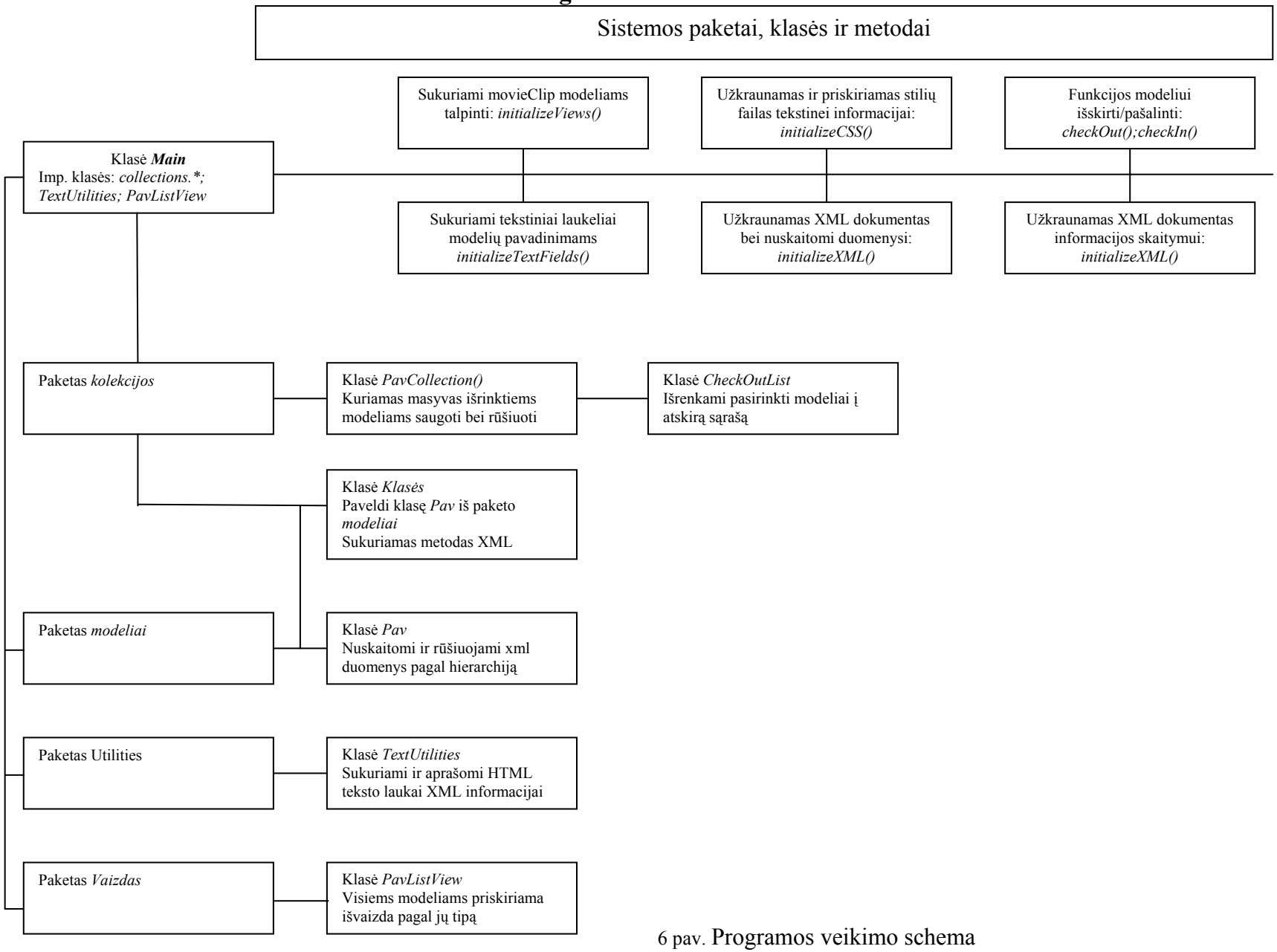
Reikalavimas #:	4	Reikalavimo tipas:	9.1	Įvykis / panaudojimo atvejis #:	1 -2
Aprašymas:	Duomenys turi būti saugomi xml faile				
Pagrindimas:	Lankstus darbas su duomenimis				
Šaltinis:	Vartotojas				
Tikimo kriterijus:	Lankstumas				
Priklausomybės	3	Konfliktai:			Nėra
Papildoma medžiaga:	Nėra				
Istorija:	Užregistruotas 2005 spalio 14d.				

3.1.6. Duomenų srautai



5. pav. sistemos duomenų srautų diagrama

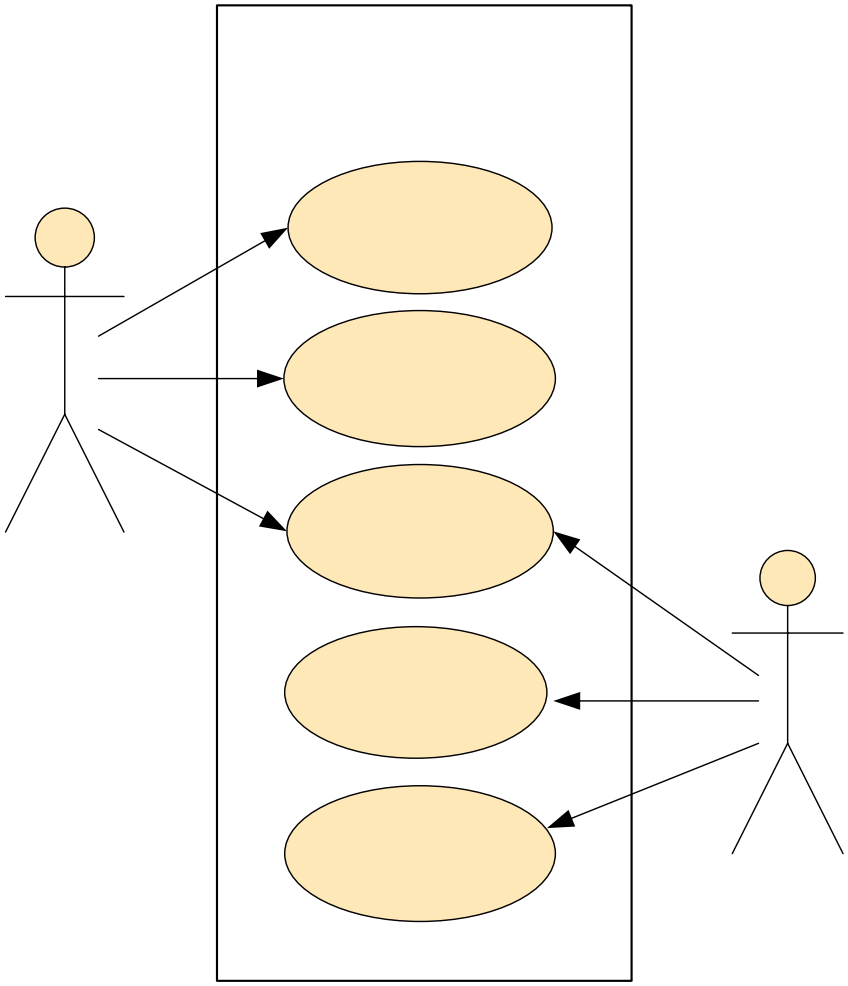
3.2. Programos veikimo schema



6 pav. Programos veikimo schema

3.3. Projektuojamos sistemos architektūra

Kuriamos sistemos panaudojimo atvejų diagrama pavaizduota pirmame paveiksle (1 pav.).



7 pav. Projektuojamos sistemos ER modelis

Lentelė 6 panaudojimo atvejis.

Nr.	1
Pavadinimas:	Prižiūrėti sistemą
Vartotojo/aktoriaus pavadinimas:	Administratorius
Aprašas:	Tikrinamas sistemos veikimas.
Prieš sąlyga:	Sugalvojama kaip bus aprašoma norima specifikuoti sistema, kasi ją sudarys.
Sužadinimo sąlyga:	Sistema turi veikti puikiai.
Po sąlyga:	Sistema tikrinama ir testuojama.

Lentelė 7 panaudojimo atvejis.

Nr.	2
Pavadinimas:	Atnaujinti sistemą
Vartotojo/aktoriaus pavadinimas:	Administratorius
Aprašas:	Administratorius talpina tekstinę informaciją XML formatu bei talpina ją į internetą kartu su nuotraukomis. Taip pat atnaujinama jau pasenusi informacija.
Prieš sąlyga:	Sugalvojama kaip bus aprašoma norima specifikuoti sistema.
Sužadinimo sąlyga:	Administratorius nori atnaujinti sistemą.
Po sąlyga:	Sistema yra atnaujinta.

Lentelė 8 panaudojimo atvejis.

Nr.	3
Pavadinimas:	Rasti informaciją
Vartotojo/aktoriaus pavadinimas:	Vartotojas
Aprašas:	Įsijungęs sistemą internete, vartotojas turi rasti jam tinkamą informaciją.
Prieš sąlyga:	Sugalvojama kaip bus aprašoma norima specifikuoti sistema, kas ją sudarys.
Sužadinimo sąlyga:	Vartotojas nori rasti informaciją.
Po sąlyga:	Informacija sėkmingai rasta.

Lentelė 9 panaudojimo atvejis.

Nr.	4
Pavadinimas:	Išsirinkti informaciją
Vartotojo/aktoriaus pavadinimas:	Vartotojas
Aprašas:	Informacija talpinama į naują langą, kur vėliau gali peržiūrėti išsirinktą informaciją.
Prieš sąlyga:	Sugalvojama kaip bus aprašoma norima specifikuoti sistema, kokie agregatai ją sudarys.
Sužadinimo sąlyga:	Vartotojas nori peržiūrėti išsirinktą informaciją
Po sąlyga:	Vartotojas sėkmingai peržiūri išsirinktą informaciją.

Lentelė 10 panaudojimo atvejis.

Nr.	5
Pavadinimas:	Spausdinti informaciją
Vartotojo/aktoriaus pavadinimas:	Vartotojas, administratorius
Aprašas:	Informacija spausdinama
Prieš sąlyga:	Sugalvojama kaip bus aprašoma norima specifikuoti sistema, kokie agregatai ją sudarys.
Sužadinimo sąlyga:	Vartotojas nori spausdinti išsirinktą informaciją.
Po sąlyga:	Vartotojas sėkmingai spausdina išsirinktą informaciją.

Paketas „Kolekcijos“

Paketą sudaro trys klasės — „PavCollection()“, „CheckOutList()“ ir „Klases()“. Klasė, — „PavCollection()“, kuriamas masyvas išrinktiems modeliams saugoti. „CheckOutList()“ — išrenkami pasirinkti modeliai į atskirą sąrašą. „Klases()“ — paveldi klasę „Pav()“ iš paketo Items, sukuriamas metodas XML duomenims siųsti (setter metodas).

Paketas „Modeliai“

Sudaro klasė „Pav()“ — nuskaitomi ir rūšiuojami XML duomenys pagal hierarchiją.

Paketas „Utilities“

Sudaro klasė „TextUtilities()“ — šioje klasėje sukuriami ir aprašomi HTML teksto laukai XML informacijai.

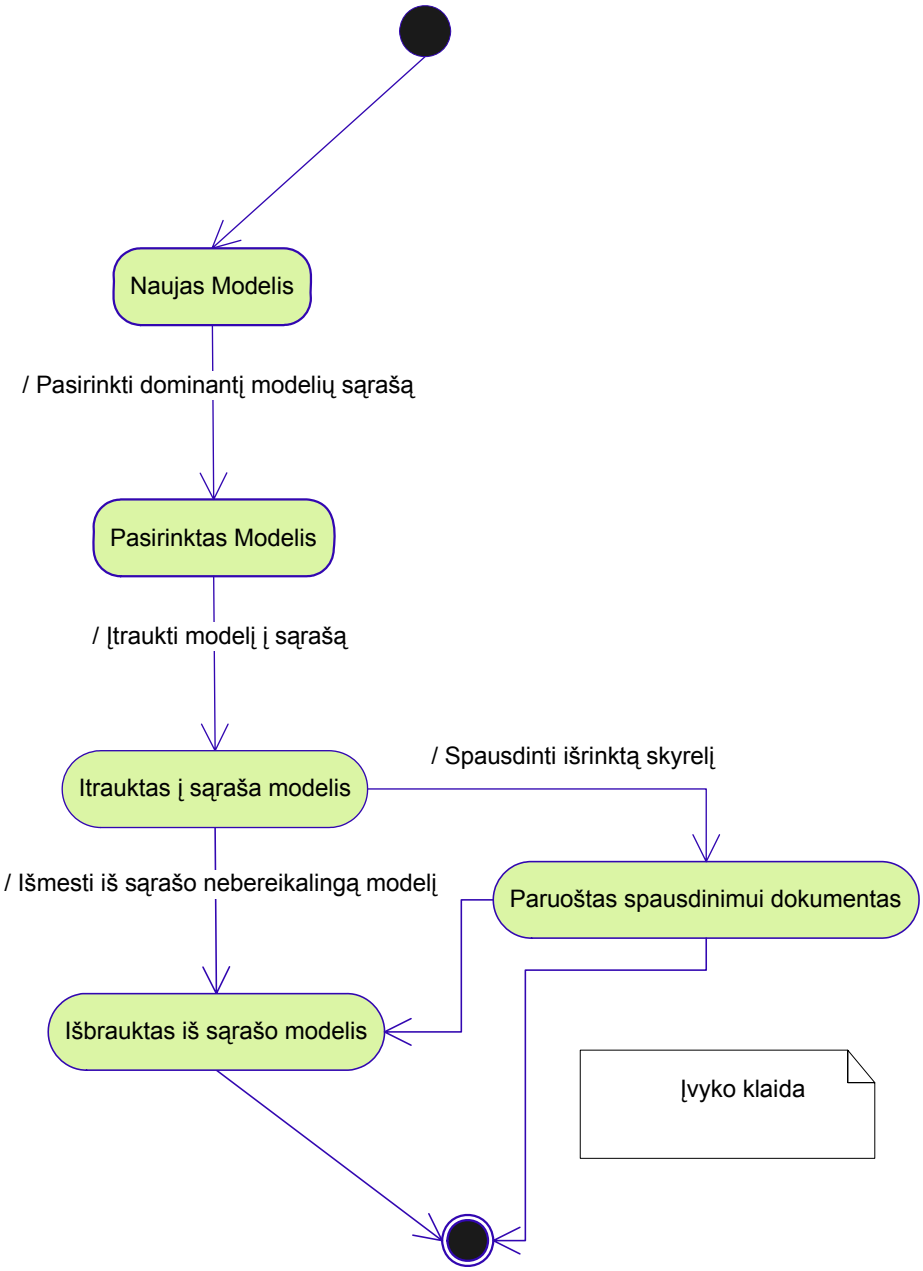
Paketas „Vaizdas“

Šį paketą sudaro klasė „PavListView()“. Šioje klasėje visiems modeliams yra priskiriama išvaizda pagal jų tipą. Taip pat skirta sukurti teksto laukeliui, stilius imamas iš stiliau lapo (styles.css failo), tai yra, šioje klasėje yra nurodomas teksto stilius

Kuriamos sistemos būsenų diagrama.

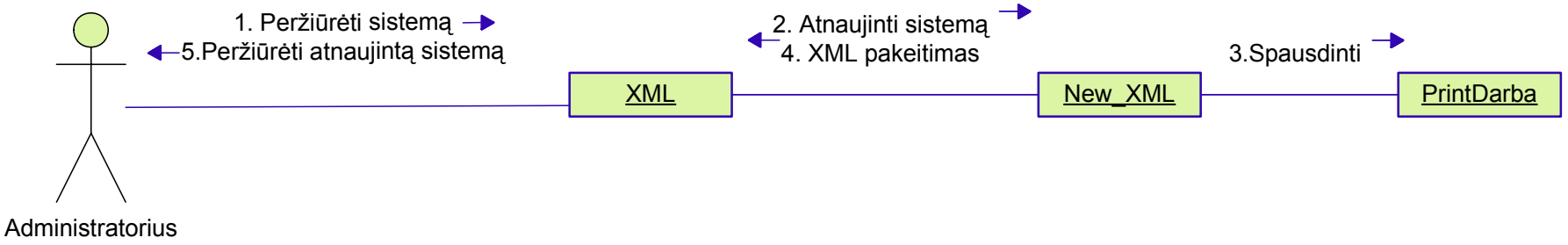
Pateikiama sistemos objektų būsenų diagramos, sistemos elementų bendradarbiavimo bei sekų diagramos.

Būsenų diagrama.

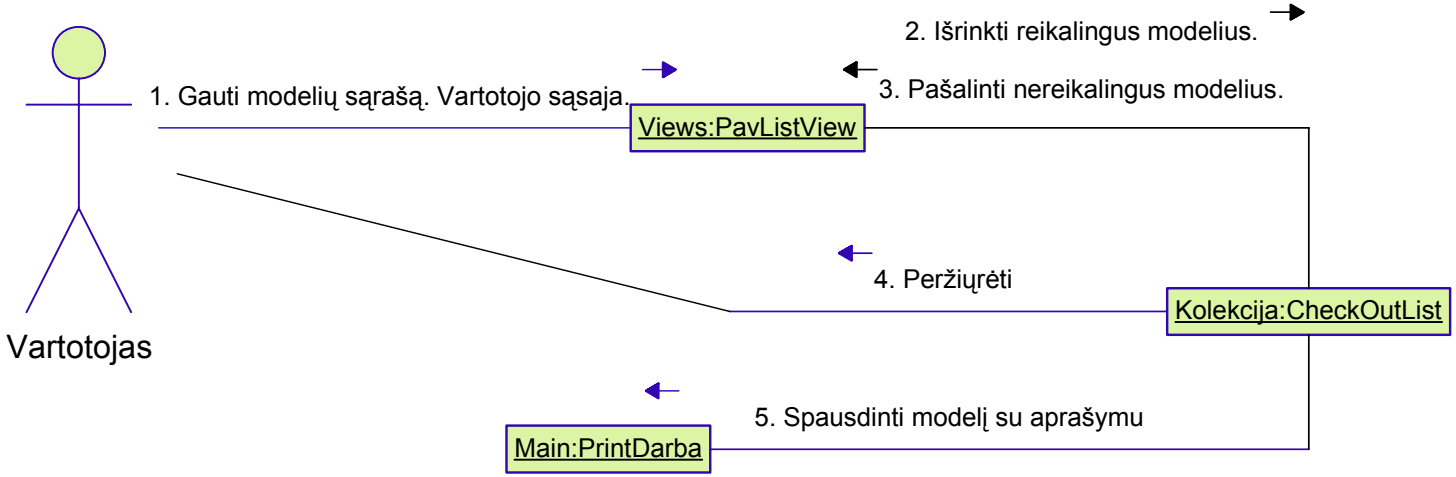


8 pav. Sistemos būsenų diagrama.

Bendradarbiavimo diagrama.

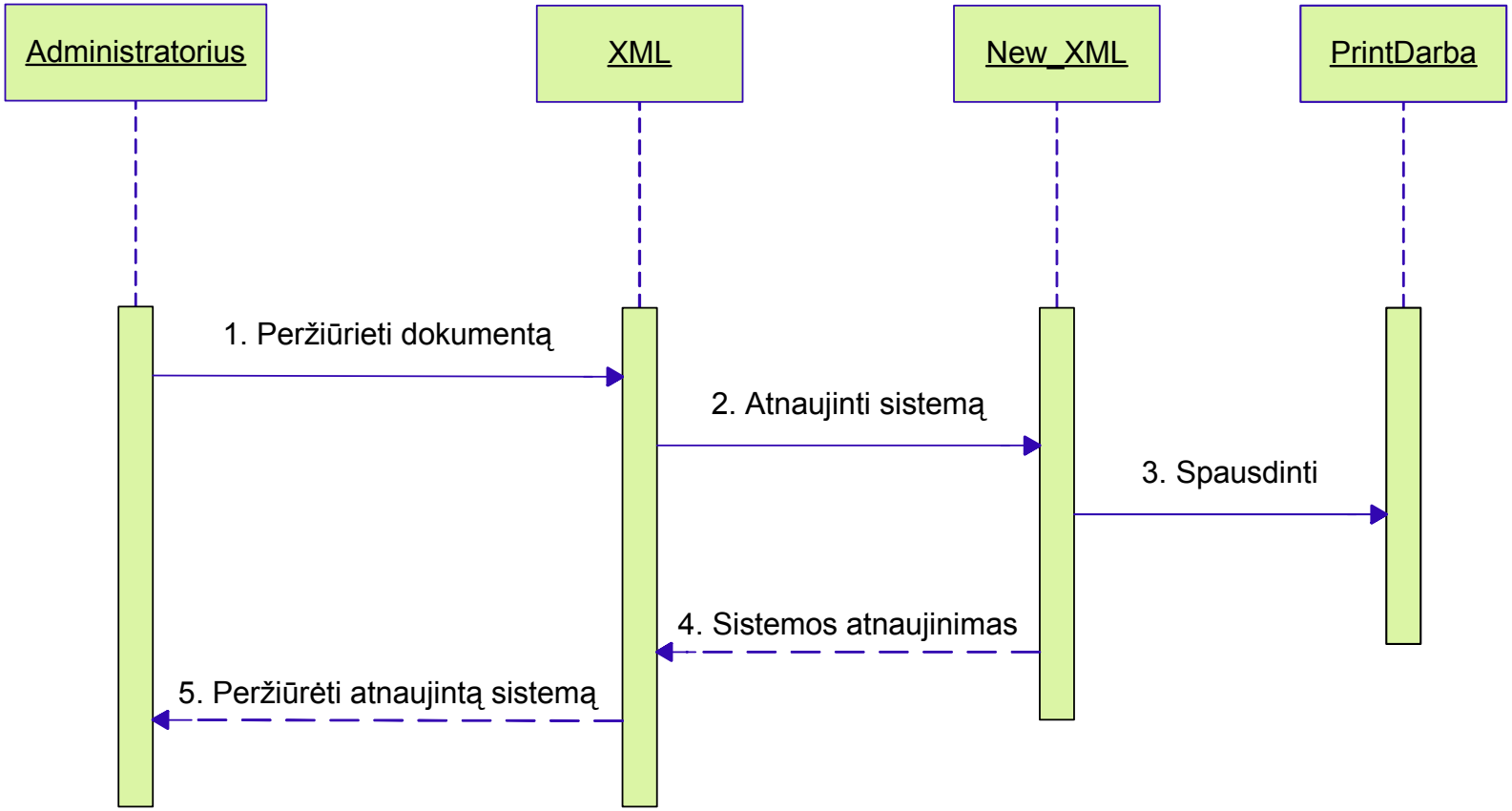


9 pav. Sistemos bendradarbiavimo schema

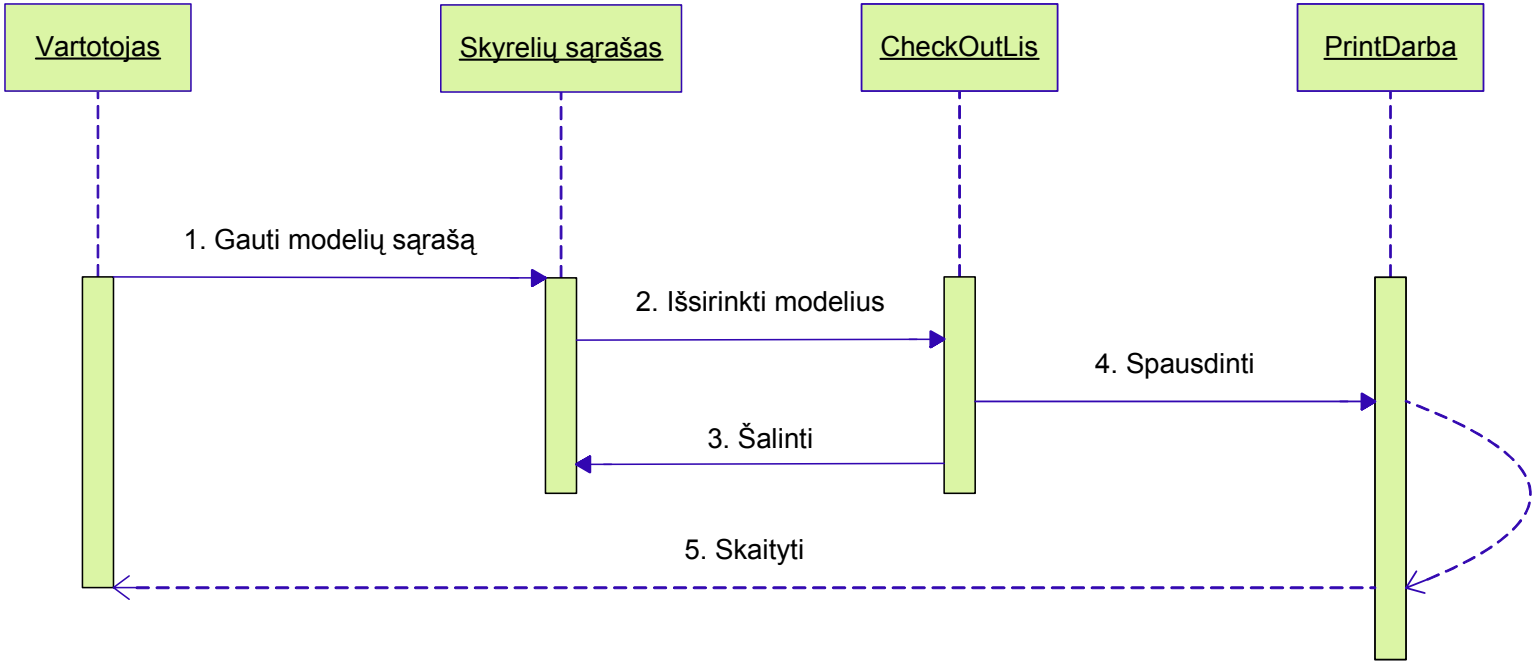


10 pav. Sistemos bendradarbiavimo schema

Sekų diagrama.



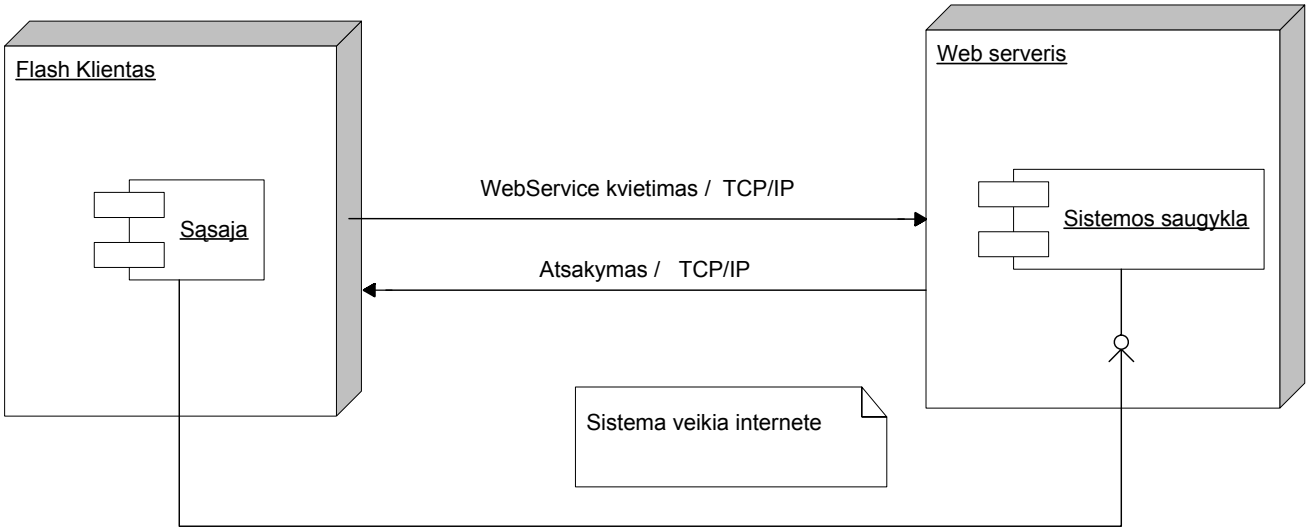
11 pav. Kuriamos sistemos sekų diagrama



12 pav. Kuriamos sistemos sekų diagrama

Projektuojamos sistemos išdėstymo vaizdas

Paveikslėlyje pateikta sistemos išdėstymo diagrama.



13 pav. Sistemos išdėstymo vaizdas

Web klientas.

Web klientas naudosis sistema per tinklo naršyklę. Sistemos veikimas nepriklauso nuo kliento platformos ar operacinės sistemos. Sistemos paslaugoms pasiekti web klientas privalo turėti shockwave flash priedą.

Web klientams rekomenduojama

Minimalus CPU: 300 MHz

Minimalus RAM kiekis: 128 MB

Minimalus laisvos disko vietos dydis 10 MB

Web serveris

WEB servisas diegiamas Microsoft Windows 2003 Server kompiuteryje. Jam įdiegti dar naudojama Microsoft Infomation Services 6.0 web serverio versija.

Minimalus CPU: 600 MHz

Minimalus RAM kiekis: 256 MB

Minimalus Disko dydis 4 GB

Serveriui reikalingas tinklo palaikymas. Web servisas teikia paslaugas naudojant HTTP protokolą, o HTTP naudoja TCP/IP protokolą perduoti duomenis žemiausiame lygyje.

WEB servisas realizuojamas .NET platformoje ir naudoja Microsft .NET Framework virtualia mašiną.

Kompiuteryje turi būti įdiegta Microsoft .NET Framework.

Duomenų vaizdas

Bibliotekos duomenys saugomi XML faile (biblioteka.xml). Siame faile yra saugomi modelių bei iškarpų aprašymai.

3.4. Programinių modulių specifikacija

XML dokumentų užkrovimas. Action script 2.0 turi XML klasę. Pasinaudoję XML konstruktoriumi galime sukurti šios klasės instanciją savo programoje.

```
var xmlNaujasKintamasis:XML = new XML();
```

Pavyzdyje kintamajam xmlNaujasKintamasis yra priskiriamas XML duomenų tipas bei jam priskiriamas XML objektas.XML kintamiesiems parametrai nėra būtini, tačiau gali būti siunčiami konstruktoriui.

Siekiant eliminuoti tarpus, naujų eilučių simbolius iš xml dokumento – priskiriamas ignoreWhite žymė kintamąjam.

```
xmlNaujasKintamasis.ignoreWhite = true;
```

Svarbus veiksmas kraunant XML dokumentus – onLoad įvykio prižiūrėtojas, startuojantis tuojau pat po XML failo užkrovimo.

```
xmlNaujasKintamasis.onLoad = function(bSekmingas:boolean):Void {
    if(bSekmingas){
            // tolesnis darbas su XML }
};
```

Aukščiau pateikta funkcija siunčia boolean reikšmę, nusakančią apie sėkmingą XML dokumento užkrovimą. Šioje funkcijoje talpinamas kodas, „apdorojantis“ XML failą.

Kaip ir kiekviena action script 2.0 funkcija, taip ir onLoad metodas turi būti iškviestas bei jam siunčiamas xml failo pavadinimas bei kelias iki jo.

```
xmlNaujasKintamasis.load(„naujasXmlDokumentas.xml“);
```

XML užkrovimas buvo realizuotas Main klasėje.

```
private var _xmlKlases:XML;
_xmlKlases = new XML();
private function initializeXML():Void {
                var oClass:Object = this;
                _xmlKlases.ignoreWhite = true;
                _xmlKlases.onLoad = function(bSuccess:Boolean):Void {
                  oClass._lbOnlineBranch.dataProvider = this.firstChild;
                };
                _xmlKlases.load("tekstas" + tekstoNr + ".xml");
   }
```

Darbas su XML duomenimis. Užkrovus XML dokumentą jis turi būti apdorojamas programiškai siekiant atskirti reikalingus hierarchijos lygmenis. Visi XML objektai sudaryti iš XMLNode objektų, bei jų savybių naudojamų informacijai išskirti.

Naudojami XMLNode objektų savybės: *firstChild, lastChild, nextSibling, previousSibling, parentNode, attributes, nodeName* bei *nodeValue.*

firstChild ypatybė gražina šakninius xml dokumento duomenis. Siekiant gražinti pirmą šakninio įrašo vaiką, naudojame dvi *firstChild* ypatybes.

Pvz. *xmlKintamasis:XMLNode = this.firstChild.firstChild*.

lastChild ypatybė gražina paskutinį šakninio įrašo vaiką.

Siekiant gražinti toliau esantį to paties hierarchinio lygmens įrašą, naudojamas *nextSibling* atributas.

```
var XNSaknis:XMLNode = this.firstChild;
var xnPirmasIrasas:XMLNode = xnSaknis.firstChild;
trace(xnPirmasIrasas.nextSibling);
```

Atributas *nodeName* gražina esamo hierarchinio lygmens pavadinimą, įrašytą tarp „<“ ir „>“ skiriamųjų skliaustų.

nodeValue atributas naudojamas tik su tekstu, todėl neveikia su atskirais elementais. Pavyzdžiui: *trace(xnPavadinimas.firstChild.nodeValue);*

Ciklas, gražinantis reikalingą informaciją iš XML:

```
for (var i:Number = 0; i < this.firstChild.childNodes.length; i++) {  //gražinamas modelių kiekis
    xnModelis = this.firstChild.childNodes[i];       //gražinamas modelis
    mdTipas = new Pav(xnPav.attributes.title, xnPav.attributes.rusis,
    xnPav.firstChild.nodeValue);                     //gražinama reikšmė
    trace(mdTipas.toString());                       //išvedama XML informaciją lange
}
```

Aprašomas „setter“ (siuntimo) metodas XML dokumento apdorojimui naudoti.

```
public function set dataProvider(xnKlases:XMLNode):Void {
    var bkItem:Pav;
    for(var i:Number = 0; i < xnKlases.childNodes.length; i++) {
                bkItem = new Pav();
                bkItem.dataProvider = xnKlases.childNodes[i];
                _aPavs.push(bkItem);
}
```

Meniu pasirinkimai, paaiškinimai bei tekstas yra suskirstyti į modelius, kurių išvaizda (teksto dydis, šriftas ir pan.) yra keičiama „css“ stilių failu. Išvaizdos priskyrimas modeliams pateiktas apačioje.

```
public function set model(bcModel:PavCollection):Void {
                _bcModel = bcModel;
                _bcModel.addEventListener("update", this);
    }
```

Patikusio skyrelio pavadinimas gali būti įtraukas į pasirinkimų sąrašą, bei pašalintas iš jo. Sąrašas rūšiuojamas pagal abėcėlę.

```
public function addItem(bkItem:Pav):Void {
        _aPavs.push(bkItem);                    //skyrelių pridėjimas sąraše
        dispatchEvent({type: "update", target: this});
}

public function removeItemAt(nIndex:Number):Void {
        _aPavs.splice(nIndex, 1);               // skyrelių
išmetimas
        dispatchEvent({type: "update", target: this});
}

public function get collection():Array {
        _aPavs.sort();                          // rūšiuojami skyreliai
        return _aPavs;
}
```

Metodas, priskiriantis pavyzdžiams identifikacinius numerius.

```
public function findPavID(bkItem:Pav):Number {
    for(var i:Number = 0; i < _aPavs.length; i++) {
        if(_aPavs[i] == bkItem) {
        return i;
        }
```

3.5. Suprojektuotos sistemos testavimas

3.5.1 Vartotojo sąsajos testavimas

Sukurtos sistemos vartotojo sąsajos testavimą atlikau dviem etapais:

1. Pirmame etape tikrinau visų sistemos mygtukų bei funkcijų veikimą, funkcijų pavadinimų teisingumą bei suprantamumą.
2. Antrąjį testavimo etapą atliko tretieji asmenys. Vartotojai naudojosi sistema, taip buvo nustatyta ar vartotojo sąsaja aiški, o taip pat ar sistema veikia korektiškai.

Abu vertinimo kriterijai buvo įvertinti gerai. Vartotojo sąsaja yra labai aiški, vartotojui nebuvo sunku susigaudyti programos funkcijų panaudojime . Suprojektuotos sustemos veikimo teisingumas vertinamas taip pat labai gerai. Testavimo metu vartotojas nesusidūrė su jokiomis problemomis.

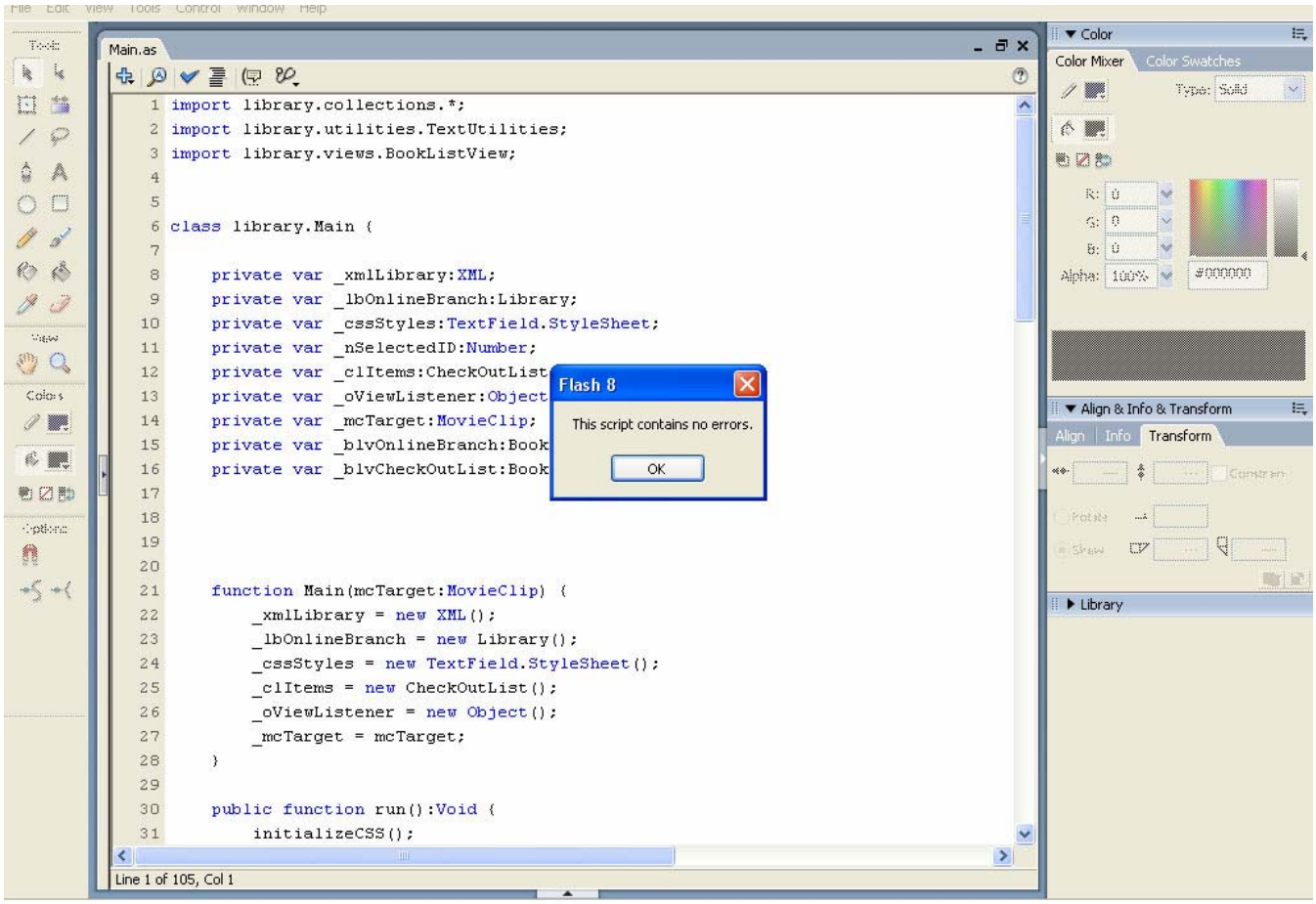
3.5.2. Programos funkcijų testavimas

Kad galėtume nustatyti bei ištaisyti sukurtos sistemos klaidas ir defektus atlikau programinės įrangos testavimą. Problemoms nustatyti naudojausi dviem sistemos testavimo būdais:

- Statinis (automatizuota statinė analizė) (14 pav.),
- Dinaminis sistemos tikrinimas (programinės įrangos testavimas).

Automatizuotos statinės analizės metu atlikau sistemos programinio kodo teisingumo patikrinimą. Šis teisingumo patikrinimas buvo atliktas Macromedia Flash MX 2004 kompiliatoriumi. Kompiliatorius patikrinimo metu radęs klaidingą programos kodo vietą nurodo įvykdytos klaidos eilutės vietą bei išveda į ekraną pranešimą su klaidos aprašymu. Programos kodas buvo testuojamas iki tol, kol kompiliatorius nebeaptiko nei vienos kodo sintaksės ar kitokio pobūdžio klaidos. Automatizuotos statinės analizės metu kompiliatoriaus aptiktos klaidos buvo ištaisytos ir pašalintos.

Tikrinimo fazėje vykdžiau *komponentų testavimą*: testavimas stambinančiu integravimo testavimo principu. Visi komponentai buvo integruojami iki, kol buvo sukurta visiškai sukomplektuota programa. Testavimo metu tikrinau, kaip veikia visų programos modulių komponentai: duomenų iš XML failo skaitymas, duomenų išvedimas iš XML į ekraną, pelės klavišo paspaudimą, klaviatūros klavišo paspaudimą, mygtukų reakciją į pelės paspaudimą, psausdinimas. Tirkinimo metu buvo ustatyta, jog daugiau nei 95% visų elementų veikia teisingai. Patikrinimo metu aptiktos klaidos: pasirinkus modelio rūšį modelių sąrašas atsiveria tik iš antro pelės klavišo paspaudimo. Surastos klaidos buvo pašalintos ir tai užtikrino visų formų elementų korektišką veikimą.



14 pav. Automatizuotos statinė analizė

Sistemos testavimo metu sistema buvo patalpinta į interneto serverį. Į tą patį katalogą buvo patalpinti ir XML failai, kurių informacija buvo keičiama. Pakeitus xml faile esančią informaciją buvo stebimas informacijos pateikimas sistemoje. Bandžiau XML failuose įrašyti skirtingomis raidėmis prasidedančius skyrelius, kurie yra išdėstomi pagal abėcėlę. Visuomet skyreliai buvo išdėstomi korektiškai. Išsirinkus norimą skyrelį buvo stebimas teisingas jo pavadinimas išrinktų skyrelių sąraše. Taip pat išrinkus kelis skyrelius, šie buvo surikiuojami pagal abėcėlę patogiai peržiūrai. Iš pasirinktų skyrelių sąrašo pavadinimai turi būti pašalinami pelės paspaudimu. Paspaudus pele ant skyrelio pavadinimo turi būti pašalinamas tik pasirinktas skyrelis. Bet kurio bandymo metu skyreliai buvo pašalinami teisingai. Bandant sistemą XML failai buvo atnaujinami 15 kartų. Visus kartus skyrelių aprašymai buvo pateikiami tiksliai pagal vartotojo paspaustas nuorodas (skyrelių pavadinimus).

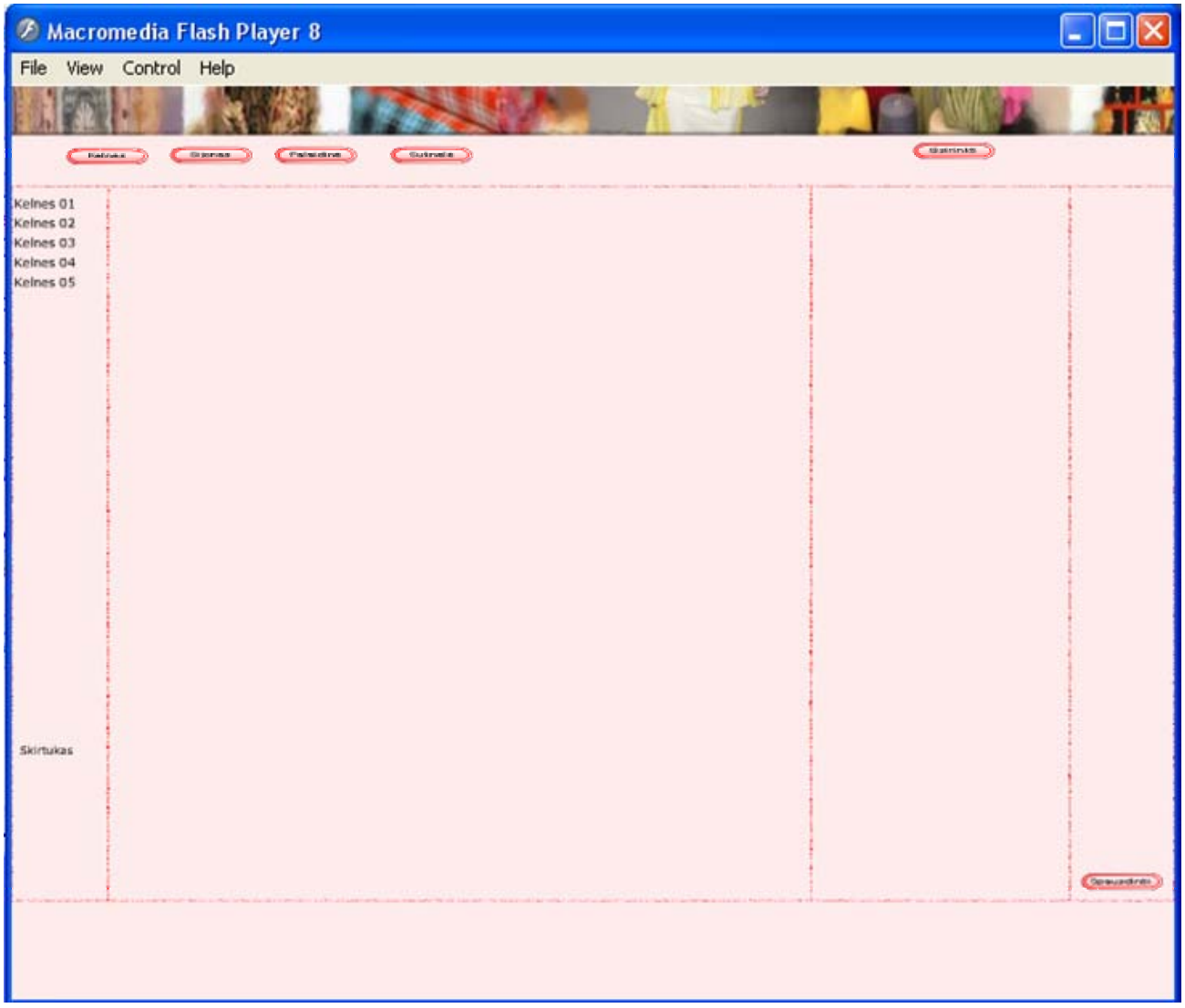
Testavimo metu bandžiau keisti failo *styles.css* duomenis. Pakeitus šrifto dydį, patį šriftą ar jo spalvą – sistemos pateikiamo teksto šriftas taip pat pasikeisdavo. Teksto aprašymo, išrinktų skyrelių bei skyrelių sąrašo šriftas pasikeičia tiksliai pagal *styles.css* faile nurodytus parametrus.

Paspaudus ant norimo skyrelio – buvo išvedamos tam skyreliui paruoštos nuotraukos.

4. Produkto kokybės kriterijai

Vienas iš svarbiausių produkto kokybės kriterijų yra paprasta ir suprantama vartotojo sąsaja. Jei vartotojo sąsaja bus neaiški – programa gali tapti nevartojama. Taip pat labai svarbu kaip veikia sistema. Sistema turi pateikti vartotojo išsirinktą informaciją, pareikalavus išrinkti ją bei atspausdinti. Sukurtą sistemą tobulinsiu, jeigu sistema bus populiari tarp vartotojų, ar programos užsakovas pareikalaus naujų funkcijų, arba tiesiog nepatenkintas esamomis.

Sistemos vartotojo sąsaja buvo kuriama naudojant Action Script 2.0 programavimo kalbą. Visos sistemos komandos yra paprastos ir valdomos pele arba klaviatūros klavišais. Vartotojas neturi galimybės keisti programos nustatymų. Duomenų bazė sistemoje nėra naudojama, nes visi programos duomenys yra laikomi XML dokumente. Kadangi visos funkcijos pasirenkamos iš galimų funkcijų sąrašo, todėl duomenų kontrolės sistemoje nėra. Vartotojo pranešimai nėra naudojami, nes nenumatytų įvykių testuojant nepasitaikė.



15 pav. Sistemos pagrindinis langas

4.1. Sistemos valdymas

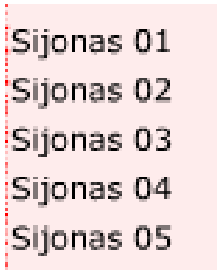
Sukurta sistema yra valdoma pele. Pelės pagaba vartotojas turi pasirinkti norimą modelio rūšį ir tada peržiūrėti modelių sąrašą, išsirinkti modelį, prie kurio nori vėliau sugrįžti ar atspausdinti.

Norint išeiti iš sistemos reikia atlikti šiuos veiksmus: jei programa yra paleidžiama iš internetonio puslapio – sistema uždaroma išjungiant naršyklės langą, jeigu programa paleidžiama vykdomuoju „projector“ failu – iš jos išeinama paspaudus išjungimo mygtuką dešiniajame viršutiniame lango kampe arba pasirinkus „file“ meniu ir „exit“ punktą.

4.2. Modelių pasirinkimas, peržiūra ir spausdinimas

Lango viršutinėje dalyje matome modelių pasirinkimo mygtukus (16 pav.), kuriais galime pasirinkti norimą modelio rūšį. Visi sistemos mygtukai yra užkraunami dinamiškai iš XML failo.

16 pav. Modelių pasirinkimo mygtukai



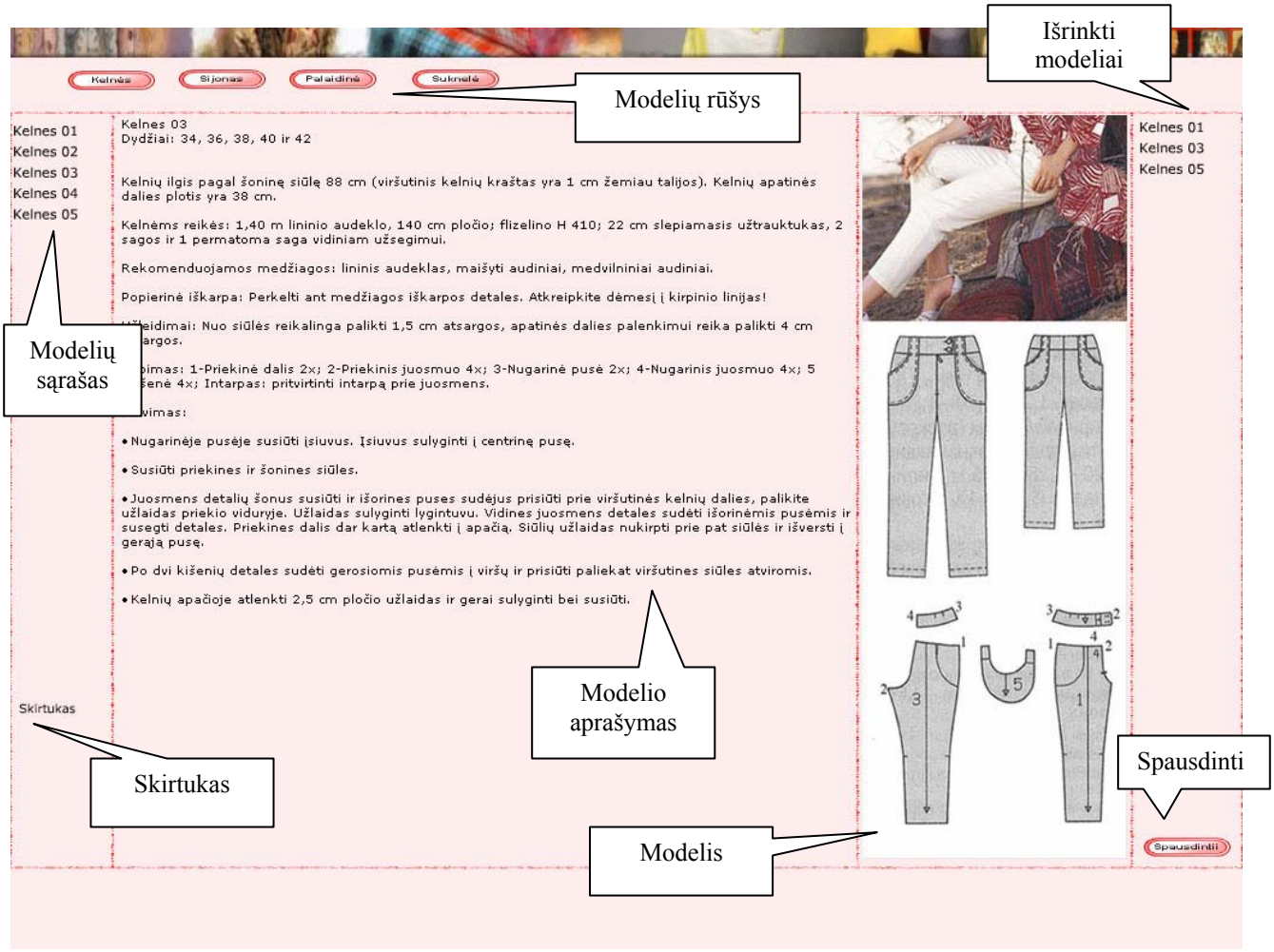
17 pav. modelių sąrašas

Išsirinkus norimą modelių rūšį dešinėje sistemos lango pusėje matome esamų tos rūšies modelių sąrašą (17 pav.). Pelės pagalba, paspaudus ant modelio pavadinimo, galima peržiūrėti visus sąraše pateiktus modelius.

Modelio išskyrimas atliekamas mygtuko „Skirtukas“ pagalba. Pasirinkus patinkantį modelį vartotojas gali jį įsikelti į išskirtų modelių sąrašą esantį sistemos lango dešiniajame šone. Modeliai yra rūšiuojami pagal alfabetą.

18 pav. Išsirinktų modelių sąrašas

Sistema turi galimybę atspausdinti patikusį modelį. Spausdinimas vyksta paspaudus mygtuką „Spausdinti“. Taip pat reikia, jog spausdintuvo parametruose vartotojas pasirinktų lapo padėtį „*Landscape*“.



19 pav. Sistemos langas su išsirinktu modeliu

5.Vartotojo dokumentacija

Programinės įrangos reikalavimai:

- Operacinė sistema: MsWindows, Linux (su grafiniu apvalkalu), MacOS.
- Interneto naršyklė su shockwave flash priedu.

Minimalūs techninės įrangos reikalavimai:

- Centrinis procesorius – Pentium III ar alternatyvus.
- Operatyvinė atmintis – 32MB.
- Vaizdo posistemė – 1024x728 raiška, bent 16 bit spalvų.
- Internetas.

5.1. Vartotojo vadovas

1. Įsitikinkite, kad esate grafinėje operacinės sistemos aplinkoje.
2. Pasileiskite interneto naršyklę.
3. Surinkite interneto adresą http://www.geocities.com/simkaite
4. ***Modelio peržiūra:***
    a. Užsikrovus programai iš meniu juostos pasirinkite norimą rūbo modelį, pavyzdžiui *<kelnės>*.
    b. Atsivėrusiame sąraše pelės pagalba pasirinkite įvairius modeliu, juos peržiūrėkite.
5. ***Modelių išrinkimas:***
    a. Meniu juostoje pasirinkus norimą modelį mygtuko *<Išsirinkti modelį>* pagalba įkelkite jį į išsirinktų modelių formą
6. ***Modelio šalinimas iš išrinktųjų sąrašo:***
    a. Norėdami pašalinti modelį iš formos *<Išsirinkti modeliai>,* reikia pelės pagalba spausti ant nebereikalingo modelio pavadinimo.
7. ***Modelio spausdinimas:***
    a. Išsirinkus norimą modelį iš meniu justos išsirinkite mygtuką spausdinti.

5.2. Administratoriaus dokumentas

Programą sudaro 36 failai.

Index.html – xHTML failas, startuojantis po interneto adreso surinkimo.

Failai su *.swf galūne – sukompiliuoti Macromedia Flash MX 2004 Action Script 2.0 failai.

Failai su *.jpg galūne – grafiniai modelių failai.

Visi failai turi būti patalpinti į šakninį (root) interneto puslapio katalogą.

Norint pakeisti tekstinę informaciją – reikia keisti xml failuose esančius įrašus, grafinius failus, arba styles.css faile esančius teksto išvedimo parametrus.

6. Produkto kokybės įvertinimas

Vartotojo sąsaja buvo kuriama kuo aiškesnė, kad nepatyręs vartotojas galėtų nesunkiai perprasti programos valdymą. Kuriant vartotojo sąsają stengiausi išlaikyti sąsajos pastovumą, maksimaliai išlaikyti tokią pačią navigacijų, meniu formatą.

Sukurta sistema veikia tiesiog interneto puslapyje. Ji patalpinta adresu: http://www.geocities.com/simkaite. Testavimas buvo atliktas įvairių išsilavinimų ir sričių žmonių. Nemažą susidomėjimą parodė žmonės, kurie domisi ar tik pradeda domėtis rūbų siuvimu.

Sistemą testavę vartotojai įvertino programos savybes įvertino aukščiau nei 80 balų, kas byloja apie labai gerą reikalavimų įvykdymą bei vartotojų lūkesčių pateisinimą.

Nesėkmingo pasinaudojimo programa atsirasdavo tik dėl Shockwave Flash priedo nebuvimo ar dėl kompiuterio techninės ir programinės įrangos netvarkingumo. Sistema puikiai dirbo tiek Windows, tiek Linux ar MacOSX aplinkose.

Programos greitaveika tiesiogiai atitinka vartotojo centrinio procesoriaus spartą. Bandant ją kompiuteriu, neatitinkančiu minimalių reikalavimų, sistemos greitaveika buvo sumažėjusi, rezultatų pateikimas ekrane buvo tik patenkinamas, tačiau programa puikiai dirbo.

Visi sistemą vertinę vartotojai pripažino, jog programos funkcijos yra intuityviai suvokiamos.

IŠVADOS

1. XML sparčiai plintanti ir besivystanti duomenų atvaizdavimo ir aprašymo kalba, kuri leidžia patogiai saugoti struktūrizuotą tekstinę informaciją ir pateikti ją internete.
2. Sistemos kodas yra parašytas Action Script 2,0 programavimo kalba išnaudojant visas darbo su XML galimybes.
3. Lietuvos rinkoje nėra informacinės sistemos, skirtos pradedančioms arba esamoms siuvėjoms, kurios rastų informaciją apie įvairius drabužių modelius bei sužinotų įvairių jų siuvimo subtilybių. Taigi sistema buvo pritaikyta šiai vartotojų grupei.
4. Sukurtos sistemos administravimas/atnaujinimas yra labai paprastas. Visi teksto atributai gali būti keičiami paprastai, kadangi yra saugomi „*style*“ faile.
5. Programoje yra naudojami skirtukai, kurie leidžia vartotojui išsirinkti jam labiausiai patikusius modelius. Išsirinkti modeliai yra rūšiuojami pagal abėcėlę. Itin paprastas išsirinkto turinio atspausdinimas.
6. Sistemoje naudojama šimtaprocentinė teksto ir paveikslų apsauga nuo kopijavimo, kadangi jie yra išvedami iš atskirų failų. Visi duomenų dokumentai gali būti patalpinti bet kurioje vietoje internete.
7. Visa tekstinė informacija yra saugoma XML dokumente. Sistemoje naudojami paveikslai saugomi JPG formatu, ir į sistemą išvedami dinamiškai programos paleidimo metu.

LITERATŪRA

1. Adobe Flex 2.0 ActionScript and MXML Language Referente. [interaktyvus]. 2006, lpvas [žiūrėta 2006 – 03- 24]. Prieiga per internetą: http://livedocs.macromedia.com /labs/1/flex20beta2/langref//index.html
2. Baniulis K., Tamulynas B. Objektinis programavimas. Iš Duomenu struktūros [interaktyvus]. [žiūrėta 2006-03-24]. Priega per internetą: http://vejas.pit.ktu.lt/ ~kazysba/ds/ds96r/objekt/t10_obj.htm
3. Bleyle J., Burger M., Diezel K., Gowin S., Harris D., Herbert B., Nelson B., Ong S. Macromedia FlashMX 2004. [interaktyvus]. 2003, spalis [žiūrėta 2005-12-12; 2005-12-13; 2006-01-06; 2006-02-24; 2006-03-16; 2006-04-25]. Prieiga per internetą: http://livedocs.macromedia.com/flash/mx2004/main_7_2/wwhelp/wwhimpl/js/html/wwhelp.htm
4. Dwyer D. *Object-oriented programming in Action Script 2.0*. September, 2002.
5. Jocobson D.; Jacobson J. *Flash and XML. A Developer's Guide.* Addison Wesley, 2001.
6. Macromedia inc. *Macromedia Flash MX Action Script 2.0 refernce guide*. Macromedia 2003.
7. McLaughlin B. *Java and XML*. United States of America, O'Reilly & Associates, Inc., 2001.
8. Ostreika A. *Programavimo Visual Basic pagrindai*. Kaunas, 2003.
9. Riškus A. *Programavimas Java. Pirmoji pažintis*. Kaunas, Technologija, 2003.
10. Starkus B. *Visual Basic 6 Jūsų kompiuteryje*. Kaunas, 2002.
11. Tapper J.; Talbot J.; Haffner R. *Object-Oriented Programming with Action Script 2*.0. New Riders Publishing, 2004.
12. T. Ray Erik. *Learning XML*. United States of America, O'Reilly & Associates, Inc., 2001.
13. Wadler Philip. An Introduction to XML and Web Technologies. [interaktyvus]. 2006, spalis [žiūrėta 2006-04-12; 2006-04-13; 2006-04-25; 2006-05-15]. Prieiga per internetą: http://www.brics.dk/ixwt/
14. Sun Developer Network. Working with XML: The Java/XML Tutorial. [interaktyvus]. 2006, [žiūrėta: 2006-04-13; 2006-04-25]. Prieiga per internetą: http://java.sun.com/xml/tutorial_intro.html

Macromedia Flash and XML technology integration using object oriented programming

Summary

Todays modern man doesn even imagine his life without computer. Since first personal computer was created, software architect companys, such as „Microsoft“, „Apple“ and others are trying to create more functional, easy to use and effective operating systems or software.

When building internet applications, Macromedia Flash gives more flexibility ant functionality than Visual Basic, C or even Java. Programming process in Action script 2.0 became very convenient , easy and fully object orriented.

With each new Flash version, designers are trying to create more convenient and functionall programming language. With Action script 2.0, Macromedia is trying to rival the proffessional application languages. This programming language ensures programming comfort and functionality.

My **primary task** is to create action script application that reads all data from xml documents. The system mus work in internet. It has to work directly in web browser for maximum convenience. The administration of the program is very simple and straight forward. Even unexpierenced user must lightly understand and use all application possibilities.

System requirements:

Application must be very simple to use

All buttons and picks must be obvious even for unexpierenced users

System must be stable

System must have a good protection on copying data

System must be available in internet

Sutrumpinimų ir terminų žodynas

API — Application programming interface. API yra sąsaja, kuri patiekia kompiuteris, aplikacija arba biblioteka

AWT — abstract window tolkit.

CSS — Cascading Style Sheets (CSS). CSS yra paprastas mechanizmas, skirtas interneto tinklapiu stiliui aprašyti.

DOM — Document Object Model (DOM). Dokumento objektinis modelis - tai sąsaja, nepriklausoma nei nuo platformos, nei nuo programavimo kalbos, suteikianti programoms ir skriptams galimybę dinamiškai pasiekti ir atnaujinti duomenis, struktūrą bei štilių dokumentuose.

DTD — dokumento tipo paskirtis DTD (Document Type Definition).

HTML — HyperText Markup Language

JVM — Javos virtuali mašina, sluoksnis tarp programos ir operacinės sistemos leidžiantis tą pačią programą vykdyti įvairiose operacinėse sistemose (Java Virtual Machine).

OO — objektiskai orientuotas.

OP — objektinis programavimas.

SAX — paprastas XML API, dazniausiai skirtas tik JAVA. SAX pirmiausia buvo pritaikytas XML ir JAVA, tai „deFacto“ standartas.

SGML — Standard Generalized Markup Language/ standartiškai apibrėžta žymių kalba.

URI — interneto resurso identifikatorius (dažniausiai URL).

W3C — The World Wide Web Consortium / www konsorciumas.

WML — Wireless Markup kalba, sukurta XML pagrindu. Ji yra pagrindine kalba, kuria kuriami WAP tinklapiai mobiliesiems telefonams.

XHTML — Extensible HyperText Markup Language.

XML — duomenų aprašymo standartas (eXtensible Markup Language).

XSL — Extensible Stylesheet Language. XSL tai rekomendacijų šeima, nusakanti XML dokumento transformacijas bei patiekimą.

PRIEDAI

1 Priedas

Programinio produkto kokybės įvertinimas:

Testuojama sritis	Testuotojas	Vertinimas [0-100balų]
Sėkmingas pasinaudojimas	1	97
	2	99
	3	94
	Vidurkis	96,67
Aiškumas	1	95
	2	83
	3	96
	Vidurkis	91
Saugumas	1	100
	2	100
	3	100
	Vidurkis	100
Naudojimo paprastumas	1	80
	2	79
	3	97
	Vidurkis	85,33
Naudojimo patogumas	1	100
	2	88
	3	79
	Vidurkis	89
Greitaveika	1	70
	2	80
	3	95
	Vidurkis	81,67

Vertinime dalyvavę testuotojai:

Pavardė, vardas	Organizacija, pareigos
Renaldas Zajančkauskas	IFM-9/3
Evaldas Bartauskis	RR - 3/2
Ramunė Bakalienė	NMA prie ŽŪM IT departamentas

2 Priedas

Programinio produkto kokybės Įvertinimas

Testuojama sritis	Testuotojas	Vertinimas [0-100balų]
Sėkmingas pasinaudojimas	1	100
	2	100
	3	100
	Vidurkis	100
Aiškumas	1	85
	2	100
	3	90
	Vidurkis	91,67
Saugumas	1	100
	2	100
	3	100
	Vidurkis	100
Naudojimo paprastumas	1	100
	2	90
	3	100
	Vidurkis	96,67
Naudojimo patogumas	1	93
	2	90
	3	97
	Vidurkis	93,33
Greitaveika	1	85
	2	100
	3	90
	Vidurkis	91,67
Funkcionalumas	1	85
	2	80
	3	85
	Vidurkis	83,33

Vertinime dalyvavę testuotojai:

Pavardė, vardas	Organizacija, pareigos
Vida Zajančkauskienė	Namų šeimininkė
Regina Bartauskienė	IĮ „Elegancija“ vadovė
Jolanta Lengvenytė	LŽŪU studentė