**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**PROGRAMŲ INŽINERIJOS KATEDRA**

**Andrej Ušaniov**

# MOBILIŲJŲ ĮRENGINIŲ GRAFINĖS VARTOTOJO SĄSAJOS AUTOMATIZUOTAS TESTAVIMAS

**Magistro darbas**

**Vadovas: dr. E. Bareiša**

**KAUNAS, 2005**

**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**PROGRAMŲ INŽINERIJOS KATEDRA**

**TVIRTINU**
**Katedros vedėjas**
**dr. E. Bareiša**
**2005-06-11**

# MOBILIŲJŲ ĮRENGINIŲ GRAFINĖS VARTOTOJO SĄSAJOS AUTOMATIZUOTAS TESTAVIMAS

**Informatikos inžinerijos magistro baigiamasis darbas**

**Kalbos konsultantė**
**Lietuvių kalbos katedros lekt.**
**dr. J. Mikelionienė**
**2005-05-19**

**Vadovas**
**dr. E. Bareiša**
**2005-05-23**

**Recenzentas**
**dr. T. Blažauskas**
**2005-05-20**

**Atliko**
**IFM 9/2 gr. stud.**
**A. Ušaniov**
**2005-05-23**

**KAUNAS, 2005**

Graphical User Interface Automated Testing for Mobile devices

# SUMMARY

Mobile devices such as cell phones and personal digital assistance are widely used in new software products. Testing takes important place in software development process. Constraints of mobile devices (speed, amount of memory, energy, small screen, wide range of platforms) raise new problems for software development process including testing phase. Automated approach of software testing reduces testing time and increases testing range. It is important to distinguish graphic user interface as a special part of testing. The aim of these master theses is to analyze automated testing of GUI for mobile devices, define testing tasks and ways to complete them.

Grafinės vartotojo sąsajos automatizuotas testavimas mobiliems įrenginiams

# SANTRAUKA

Naujuose programiniuose produktuose plačiai taikoma mobilioji įranga (mobilieji telefonai, delniniai kompiuteriai). Programinės įrangos kūrimo procese svarbią vietą užima testavimas. Dabartinių mobiliųjų įranginių apribojimai (darbo sparta, atminties kiekis, energija, ekrano dydis, platformų įvairumas) kelia naujas problemas programinės įrangos kūrimo procesui, tame tarpe ir testavimui. Testavimo proceso automatizavimas leidžia sumažinti bandymų trukmę, padidinti testavimo darbų apimtį. Programinės įrangos testavime mobiliems įrenginiams svarbu išskirti grafinės vartotojo sąsajos testavimą.

Šiame darbe nagrinėjamas grafinės vartotojo sąsajos automatizuotas testavimas mobiliai įrangai, nustatomi tikslai bei galimi keliai jiems pasiekti.

# TURINYS


1. Analitinė dalis ... 1
   1.1. Apibrėžimai ... 1
   1.1. Mobiliųjų įrenginių tipai ... 1
   1.2. Vartotojo sąsajos grafiniai komponentai ... 3
      Komponentai ... 3
      Konstrukcijos ... 6
   1.3. Grafinės vartotojo sąsajos testavimas ... 7
   1.4. Grafinės vartotojo sąsajos automatizuoto testavimo įrankių apžvalga ... 9
      CompuWare TestPartner ... 9
      IBM Rational Test Tools ... 9
      Mercury Interactive Tools ... 10
      SilkTest ... 11
      Atviro kodo grafinės sąsajos testavimo įrankiai ... 11
      Grafinės sąsajos testavimo įrankių apibendrinimas ... 12
   1.5. Grafinės vartotojo sąsajos patiekimo lygiai. ... 13
   1.6. Pakartotinas testavimas ... 14
   1.7. Testavimo proceso žingsniai ... 14
   1.8. Skriptų kodavimas ... 15
      XML kalba ... 15
      XMP sintaksės analizatoriai ... 15
2. Projektinė dalis ... 17
   2.1. Suprojektuota sistema ... 17
      Sistemos aprašymas ... 17
      Tikslas ... 18
      Problemos sprendimas pasaulyje ... 19
      Situacijos Lietuvoje įvertinimas ... 20
      Sprendimo realizacija ... 20
      Duomenų saugyklos realizacija ... 22
   2.2. Architektūra ... 25
      Klasių diagramos ... 25

Išdėstymo diagrama ..... 27
2.3. Diegimas ..... 29
3. Tyrimo dalis ..... 30
3.1. Apibrėžimai ..... 30
3.2. Problemos ..... 31
3.3. Tikslai ..... 32
3.4. Sprendimo būdas ..... 32
3.5. Langų grafo ir operacijų scenarijų pateikimas ..... 34
3.6. Langų grafo ir operacijų scenarijų failų formatai ..... 34
3.7. Testavimo aplinkos veikimas ..... 40
3.8. API ..... 42
4. Eksperimentinė dalis ..... 44
5. Išvados ..... 47
6. Terminų ir santrumpų žodynas ..... 48
7. Literatūra ..... 49
8. Priedai ..... 54
A. Navigacijos grafas ..... 54
B. Navigacijos grafo aprašymas XML kalba. ..... 55
C. Testavimo scenarijaus aprašymas XML kalba. ..... 63
D. Konferencijų medžiaga ..... 65

# PAVEIKSLĖLIŲ SĄRAŠAS

1.1 Pav. Mobiliųjų įrenginių tipai: telefonas (kairėje), delninis kompiuteris (viduryje), pranešimų gaviklis (dešinėje) ..... 2
1.2 Pav. Meniu ..... 3
1.3 Pav. Iššokantis meniu: suglaustas (kairėje), išplėstas (dešinėje) ..... 3
1.4 Pav. Teksto įvedimo laukai ..... 4
1.5 Pav. Jungikliai ..... 4
1.6 Pav. Radijo mygtukai ..... 4
1.7 Pav. Mygtukai ..... 4
1.8 Pav. Progreso indikatoriai ..... 5
1.9 Pav. Dialogo langas ..... 6
1.10 Pav. Mobiliųjų technologijų plitimas mobiliuose įrenginiuose ..... 9
1.11 Pav. Programos pateikimo vartotojui lygiai ..... 14
1.12 Pav. Testavimo proceso žingsniai ..... 15
2.1 Pav. Panaudojimo atvejų diagrama ..... 21
2.2 Pav. Duomenų bazės schema ..... 23
2.3 Pav. Duomenų sinchronizavimo algoritmas ..... 24
2.4 Pav. Įrašo būsenų diagrama ..... 25
2.5 Pav. Pagrindiniai sistemos paketai ..... 25
2.6 Pav. Paketo Common detalizavimas ..... 26
2.7 Pav. Paketo Office detalizavimas ..... 26
2.8 Pav. Paketo Mobile detalizavimas ..... 27
2.9 Pav. Išdėstymo diagrama ..... 28
3.1 Pav. Testavimo aplinkos struktūrinė schema ..... 40
3.2 Pav. Grafinės sąsajos testavimo aplinkos veiklos diagrama ..... 42
3.3 Pav. Testavimo aplinkos programavimo sąsaja ..... 43
4.1 Pav. Klaidų konvergavimas testuojant rankiniu būdu ..... 44
4.2 Pav. Klaidų konvergavimas taikant automatizuotą testavimą ..... 45
4.3 Pav. Rankinis ir automatizuotas regresiniai testavimai ..... 46

# LENTELIŲ SĄRAŠAS

Lentelė 1.1 Mobiliųjų įrenginių tipų palyginimas ..... 2
Lentelė 2.1 Duomenų bazės lentelių aprašymas ..... 23
Lentelė 3.1 Navigacijos grafo ir operacijų scenarijų aprašymo XML etiketės ..... 39

# ĮVADAS

Naujuose programiniuose produktuose plačiai taikoma mobilioji įrangą (mobilieji telefonai, delniniai kompiuteriai). Programinės įrangos kūrimo procese svarbią vietą užima testavimas. Dabartinių mobiliųjų įranginių apribojimai (darbo sparta, atminties kiekis, energija, ekrano dydis, platformų įvairumas) kelia naujas problemas programinės įrangos kūrimo procesui, taip pat ir testavimui. Testavimo proceso automatizavimas leidžia sumažinti bandymų trukmę, padidinti testavimo darbų apimtį. Programinės įrangos testavime mobiliems įrenginiams svarbu išskirti grafinės vartotojo sąsajos testavimą.

Dažnai kuriama programa mobiliai įrangai yra orientuota į taikymą daugelyje mobiliųjų platformų. Tačiau plati mobiliųjų įrenginių įvairovė kelia tam tikrų problemų. Todėl programos kūrėjams tenka atsižvelgti į mobiliųjų įrenginių architektūros bei konfigūracijos skirtumus. Mobiliųjų įrenginių skirtumai komplikuoja grafinės vartotojo sąsajos testavimą. Viename įrenginyje puikiai veikianti vartotojo sąsaja gali būti visiškai nepriimtina kitame įrenginyje.

Kuriant bei testuojant programinę įranga naudojami mobiliųjų įrenginių emuliatoriai, kurie neperteikia visų mobiliųjų įrenginių galimybių, apribojimų bei darbo ypatumų. Todėl yra svarbu išbandyti programos grafinę vartotojo sąsają tiesiogiai mobiliajame įrenginyje ar skirtingų mobiliųjų įrenginių grupėje.

Šiame darbe nagrinėjamas grafinės vartotojo sąsajos automatizuotas testavimas mobiliajai įrangai, nustatomi tikslai bei galimi keliai jiems pasiekti.

# 1. ANALITINĖ DALIS

## 1.1.APIBRĖŽIMAI

**Grafinė vartotojo sąsaja** – tai turinti hierarchinę struktūrą, grafiškai atvaizduota programos dalis, kuri priima vartotojo ir/arba sistemos sukeltus įvykius (*generated evenets*) iš fiksuotos įvykių aibės ir pateikia apibrėžtą (*determined*) grafinį rezultatą. Grafinę vartotojo sąsaja sudaro objektai – grafiniai komponentai. Kiekvienas grafinis komponentas turi fiksuotą savybių aibę. Kiekvienu laiko momentu grafinio komponento savybės turi diskrečią reikšmę. Komponentų savybių reikšmių aibė apibrėžia grafinės sąsajos būseną [12, 13, 33, 44, 51].

**Testavimas** – tai bandymas įsitikinti kad testuojamos programos veikimas atitinka arba neatitinka reikalavimų specifikaciją [5, 51].

**Mobilieji įrenginiai** – tai ypatingai portatyvūs, savarankiškai apdorojantys informaciją bei naudojantys bevieles tinklo ryšio technologijas duomenų mainams įrenginiai. Įrenginys turi būti pakankamai mažas, kad tilptų žmogaus delne [14, 57]. Mobilusis įrenginys turi atitikti šiems reikalavimams:

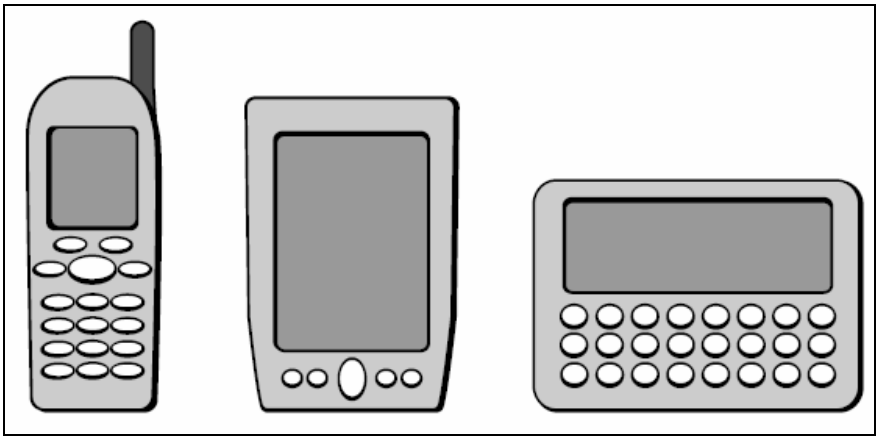
- Kabeliai prie įrenginio jungiami tik pasikrovimui ir sinchronizacijai su kompiuteriu. Kitais atvejais darbas vyksta be kabelių.
- Įrenginys valdymas atliekamas viena ranka. Darbui su įrenginiu, jį nebūtina išdėstyti ant stalo.
- Įrenginys turi palaikyti bevielio tinklo ryšio technologiją arba jų grupę (Wi-Fi, WAP, IrDA, Bluetooth [55])

**Emuliatorius** – speciali aplinka, perteikianti realaus įrenginio funkcionalumą, kuri leidžia išbandyti programos veikimą nenaudojant pačio įrenginio.

## 1.1. MOBILIŲJŲ ĮRENGINIŲ TIPAI

Rinkoje turime daug įvairių mobiliųjų įrenginių, kurie skiriasi savo išvaizda, funkcinėmis galimybėmis bei paskirtimi. Mobilieji įrenginiai skirstomi į trys pagrindines grupes [57] (1.1 Pav.):

- mobilieji telefonai (*cell phones*)
- pranešimų gavikliai (*pagers*)
- delniniai kompiuteriai (*PDAs*)



**1.1 Pav. Mobiliųjų įrenginių tipai: telefonas (kairėje), delninis kompiuteris (viduryje), pranešimų gaviklis (dešinėje)**

Kiekvienai mobiliųjų įrenginių grupei yra būdingos tam tikros savybės. Pateiksime mobiliųjų įrenginių tipų palyginimą (Lentelė 1.1).

**Lentelė 1.1 Mobiliųjų įrenginių tipų palyginimas**

| | **Mobilusis telefonas** | **Delninis kompiuteris** | **Pranešimų gaviklis** |
|---|---|---|---|
| Pirminė paskirtis | Skambučiai | Informacijos saugojimas bei apdorojimas | Elektroninis susirašinėjimas |
| Įvesties būdas | Skaitmenų klaviatūra | Pieštukas, jautrus palietimams ekranas, kartais klaviatūra | klaviatūra |
| Ekrano dydis | Mažas. Dažnai telpa 4 teksto eilutės po 12 simbolių. | Skiriamoji geba 160x160 tašku ir daugiau | Nuo vienos teksto eilutės iš 16 simbolių iki 160x160 skiriamos gebos |
| Forma | Telpa delne | Telpa marškinių kišenėje | Telpa marškinių kišenėje |
| Tinklo ryšio galimybės | Pagrindinis ryšys GSM. Gali palaikyti IrDA, Bluetooth, GPRS, EDGE, WAP | Sinchronizacijos kabelis ir IrDA. Gali palaikyti Bluetooth, Wi-Fi, GPRS | Pagrindinis ryšys GPRS. Gali palaikyti IrDA, Bluetooth |
| Išplečiamumas | Mažas arba nėra | Didelės programinės įrangos išplečiamumo galimybės. Galimybė prijungti aparatūrinius priedus | Mažas arba nėra |

Matome kad įrenginių funkcionalumas bei savybės priklauso nuo įrenginio pirminės paskirties. Tačiau pastaruoju metu rinkoje atsekama tendencija suteikti vienam mobiliajam įrenginiui platų funkcinių galimybių spektrą.

## 1.2. VARTOTOJO SĄSAJOS GRAFINIAI KOMPONENTAI

Nors delniniai/mobilieji įrenginiai skiriasi savo išvaizda bei forma, tačiau visų įrenginių vartotojo sąsajai yra būdingi bendri grafinės vartotojo sąsajos komponentai [42, 57]. Trumpai apžvelgsime grafinius sąsajos komponentus ir pateiksime rekomendacijas jų taikymui.

### Komponentai

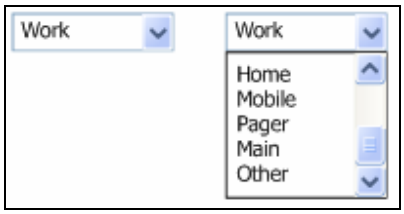
#### 1.2.A.1. Meniu

Visi įrenginiai vienaip ar kitaip palaiko meniu. Meniu - tai aibė komandų, kuri atvaizduojama grafiniu mygtuku, meniu ar įrankių juosta (1.2 Pav.). Meniu komandos skirtos valdyti programą.


**1.2 Pav. Meniu**

#### 1.2.A.2. Iššokantys meniu

Iššokantys meniu (*popup menu*) – tai objekto, kuriam priskirtas iššokantis meniu, galimų pasirinkimo variantų glaustas pateikimas vartotojui. Atvaizdavimui naudojamos rodyklės arba trikampis. Suaktyvinus meniu, vartotojui pateikiamas variantų sąrašas (1.3 Pav.).




**1.3 Pav. Iššokantis meniu: suglaustas (kairėje), išplėstas (dešinėje)**

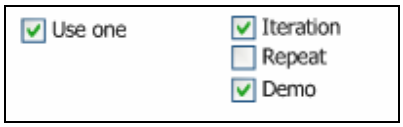
#### 1.2.A.3. Teksto įvedimo laukai

Teksto įvedimo laukai tai stačiakampiai arba taškinės linijos į kur vartotojas gali įvesti tekstą.Teksto įvedimo laukui būdinga etikete – lauko pavadinimas, kuris pateikiamas virš lauko arba kairėje. Dauguma grafinių sąsajų identifikuoja aktyvų įvedimo lauką ryškesniu rėmeliu, spalva ir/arba mirkčiojančiu kursoriumi. Teksto įvedimo laukai gali atvaizduoti vieną ar kelias teksto eilutes (1.4 Pav.). Įvedamam tekstui gali būti taikomi apribojimai: leistina simbolių aibei, įvesties ilgis.


**1.4 Pav. Teksto įvedimo laukai**

### 1.2.A.4. Jungikliai

Jungikliai (*check box*) – tai kontroliniai antspaudai, kurių pagalba įjungiamas arba išjungiama komponentas (1.5 Pav.). Jungikliai gali būti naudojami grupėje ar atskirai. Dažniausiai vienas jungiklis neįtakoja kitų jungiklių.




**1.5 Pav. Jungikliai**

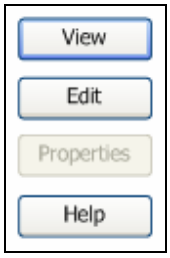
### 1.2.A.5. Radijo mygtukai

Radijo mygtukai (*radio box*) – tai vienas kita paneigiantis mygtukai. Jie visada naudojami grupėje. Vienu metu tik vienas radijo mygtukas iš mygtuku grupės gali būti įjungtas (1.6 Pav.).


**1.6 Pav. Radijo mygtukai**

### 1.2.A.6. Mygtukai

Mygtukas – tai fizinio mygtuko analogas. Mygtuko paspaudimas/bakstelėjimas atitinka veiksmą. Mygtukui gali būti priskirti programuojami mygtukai (*softkeys*), kai įrenginio mygtuko paspaudimas atitinka grafinio mygtuko paspaudimą. Mygtukai gali turėti kelias būsenas: numatytas, leistas, uždraustas (1.7 Pav.). Mygtuko pavadinimas bei kviečiama funkcija gali keistis darbo eigoje.

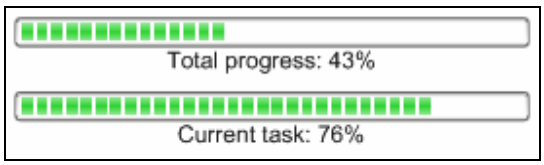



**1.7 Pav. Mygtukai**

### 1.2.A.7. Progreso indikatoriai

Ilgai trunkančių operacijų vykdymo metu svarbu kad programa informuotu vartotoją per grįžtamąjį ryšį (1.8 Pav.). Progreso indikatoriai tai geras būdas realizuoti grįžtamąjį ryšį. Rekomendacijos progreso indikatoriaus naudojimui:

- Naudoti progreso indikatorių kai operacija trunka ilgiau nei 2 sekundės
- Vengti vienas po kito sekančių indikatorių, tai vargina bei klaidina vartotoją. Naudokite vieną indikatorių kad padengti visus vienos sekos procesus.
- Dvigubo progreso indikatoriau naudojimas: vienas indikatorius atspindi visos operacijų sekos progreso būseną, kitas indikatorius atspindi atskiros operacijos progresą.
- Naudokite indikuojamo proceso sustabdymo ar nutraukimo mygtukus.
- Jei procesas trunka ilgiau nei 10 sekundžių, nurodykite praėjusį laiką ir numatytą laiką.




**1.8 Pav. Progreso indikatoriai**

### 1.2.A.8. Reikalavimai grafiniams komponentams

Grafiniams komponentams taikomi šie reikalavimai:

- Grįžtamasis ryšys (*feedback*) – vartotojo informavimas apie naudojamus komponentus. Tam puikiai tinka spalvos pakeitimas, paryškinimas bei formos pasikeitimas.
- Pavadinimas (*naming*) – komponentu pavadinimai turi tiksliai atspindėti komponento veiksmą arba savybę. Reikia vengti šabloniniu pavadinimu, tokių kaip „gerai“, „atlikti“, „taip“, „ne“. Iš komponento pavadinimo vartotojas turi suprasti, kokia yra komponento paskirtis. Taip pat į pavadinimus galima įtraukti daugtaškius „...“ kurie intuityviai informuoja vartotoją, kad suaktyvinus komponentą, jam bus pateiktas dialogas.
- Išdėstymas (*placement*) – svarbi yra komponentų išdėstymo ekrane tvarka. Numatyti komponentai turi būti pateikiami kairiau kitų komponentų. Mygtukai turi

būti patiekiami lango apačioje, kadangi jų išdėstymas lango viršuje „paslėps“ nuo vartotojo lango turinį.

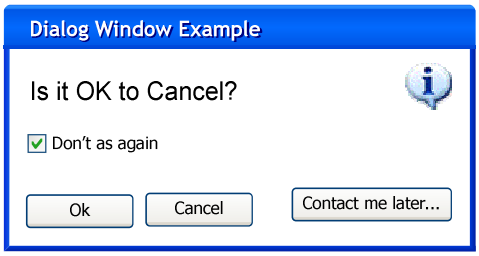
- Būsenos (*state*) – komponentų būsenų sekimas yra svarbus, siekiant efektyviai organizuoti grįžtamąjį ryšį su vartotoju. Komponentai, kurių panaudojimas operacijoje nėra leidžiamas ar apibrėžtas, turi būti uždrausti.

## Konstrukcijos

Grafinės sąsajos konstrukcijos – tai grafinių komponentų rinkiniai. Panašiai kaip sakiniai sudaromi iš žodžių, grafinės konstrukcijos sudaromos iš grafinių komponentų. Apžvelgsime keletą pagrindinių grafinių konstrukcijų [57].

### 1.2.A.9. Dialogai

Dialogas – tai programos langas vaizduojamas programos viršuje (1.9 Pav.). Šis langas iškviečiamas, kai reikalingas vartotojo dėmesys. Dialogo langų paskirtis yra pranešti vartotojui apie klaidą, informuoti apie programos būseną, užklausti vartotoją dėl tolimesnių nurodymų. Dažnai dialogo langas pateikia vartotojui du mygtukus, kurių pagalba jis gali sutikti su pasiūlymu arba jį atmesti. Šie mygtukai standartiškai yra [Gerai] (*OK*) ir [Atmesti] (*Cancel*). Tačiau ypač mobiliuose platformose, nerekomenduojama mygtukų suteikti šabloninius pavadinimus. Mygtukų pavadinimai turi atskleisti vartotojui savo funkcionalumą. Kadangi kartais programos užduodamas klausimas negali būti vienareikšmiškai interpretuotas, ir vartotojui bus neaišku, ką reikš atsakymas „Gerai“.




**1.9 Pav. Dialogo langas**

### 1.2.A.10. Formos, Langai, Vedliai

Formos tai aibė teksto įvedimo bei atvaizdavimo laukų. Dažnai formos laukai yra grupuojami atskiromis kortelėmis. Siekiant patogesnio duomenų įvedimo, šitas procesas atliekamas vedlių (*wizards*). Kiekviename vedlio žingsnyje vartotojas įvedinėja tik vienos grupės reikšmes. Tačiau vedlys taip pat gali būti perteklinis ir nepatogus vartojime. Tai nutinka kai vedlio žingsnių yra daug, o vartotojui tenka įvesti tik keletą reikšmių.

Programose mobiliesiems įrenginiams duomenų įvedimas dažnai atliekamas vedlio. Taip yra todėl kad į mažą mobilaus įrenginio ekraną netelpa visi reikalingi įvedimo laukai. Taip pat reikalavimai darbui su mobiliu įrenginių įneša savo apribojimus – vartotojas turi sugebėti atlikti darbą viena ranka [10]. Vedlys yra realizuojamas kaip programos langų seką.

#### 1.2.A.11. Iškarpinė

Delniniai įrenginiai ir pranešėjai turi iškarpinės savybę (*clipboard*), kuri leidžia atlikti duomenų kopijavimo (*copy*), iškirpimo (*cut*) ir įterpimo (*paste*) operacijas. Iškarpinė yra paranki priemonė duomenų mainams tarp įvairių programų. Mobilieji telefonai neturi tiesioginio iškarpinės palaikymo. Mobilieji telefonai taip pat nepalaiko programų keitimo (*switching*).

## 1.3. GRAFINĖS VARTOTOJO SĄSAJOS TESTAVIMAS

Programinės įrangos kūrimo išlaidų mažinimas ir programinės įrangos kokybės gerinimas yra svarbūs programinės įrangos pramones uždaviniai. Todėl programų gamintojams yra svarbu turėti automatizuotus testavimo įrankius. Šiuolaikiniai automatizuoti testavimo įrankiai leidžia kurti testavimo skriptus, generuoti testavimo atvejus, atlikti vienetų testavimą, baltos ir juodos dėžės testavimą. Šių įrankių paskirtis yra pagreitinti programinės įrangos kūrimo procesą, aptikti daugiau defektų bei sumažinti programų kūrimo kaštus atliekant regresinį testavimą. Įprastai testavimo skriptai sudaromi rankiniu būdu arba naudojant įrašymo procedūrą. Testavimo skriptų sudarymas rankiniu būdu ir/arba įrašymas tai pat gali įnešti klaidų.

Grafinės sąsajos testavimas panašus į komponentų testavimą. Siekiant automatizuoti grafinės sąsajos testavimą, reikia remtis tokia pat logika kaip ir testuojant komponentus: t. y. Automatizuotam testavimo įrankiui pateikiama testuojama programa, gaunamas testavimo rezultatas – defektų sąrašas.

Šiuolaikinių programų vartotojai kviečia programos funkcionalumą per grafinę vartotojo sąsają. Tokiu būdu per grafinės vartotojo sąsajos testavimas gali padengti programą. Programos mobiliajai įrangai taip pat pateikia vartotojui grafinę sąsają. Tačiau mobilieji įrenginiai įneša į grafinę sąsają tam tikrus apribojimus, su kuriais nesusiduria stacionarių kompiuterių vartotojai. Todėl grafinės sąsajos testavimas mobiliai įrangai yra sudėtingesnis nei stacionariems kompiuteriams.

Rankinis testavimas nėra efektyvus. Testavimo automatizavimas leidžia ženkliai padidinti testų skaičių. Tuo pačiu padidėjęs testų skaičius nereikalauja didesnių testavimo resursų. Kiekvienos naujos programos versijos testavimo kaštai mažėja [59].

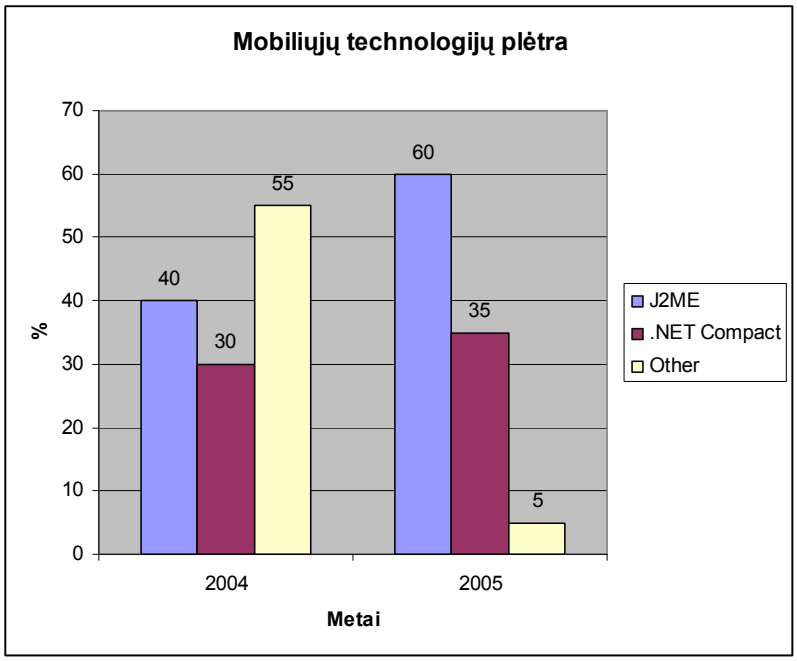
Testuojant programas automatizuotų būdu, testuotojai naudoja skriptus su komandinėm eilutėm. Tokiu būdu galima patikrinti programos atskirų dalių funkcionalumo teisingumą, tačiau testavimas yra visiškai atskirtas nuo grafinės sąsajos. Dabartiniu metu yra paplitęs įrašymo-atkartojimo (*capture/playback*) grafinės sąsajos testavimo būdas. Pirmą kartą testavimas atliekamas rankiniu būdu. Testavimo aplinka įrašo testuotojo atliekamus veiksmus grafinėje sąsajoje: teksto įvedimą iš klaviatūros, palės klavišų paspaudimus bei paspaudimo koordinates. Vėliau, atliekant automatizuotą testavimą, šie įvykiai yra atkartojami. Taip atliekamas regresinis testavimas. Tačiau įrašymo-atkartojimo metodą taikantys testavimo įrankiai nesugeba atpažinti grafinių komponentų (žiūrėkite skyrių 1.2). Dabartinės grafinės sąsajos technologijos reikalauja rankinio testavimo skriptų sudarymo, redagavimo bei rezultatų analizavimo. Iš vienos pusės, grafinės sąsajos testavimo įrankiai suteikia vartotojas galingas priemones atlikti testavimą. Iš kitos pusės, testavimo aplinkos paruošimas reikalauja didelių pastangų [13, 19, 21, 26, 58].

Grafinės vartotojo sąsajos testavimas yra gyvybiškai svarbus todėl, kad jis vykdomas taikant galinio vartotojo požiūrį į programą. Tuo pačiu jis padengia programą, kadangi programos funkcionalumas atvaizduojamas ir valdomas iš grafinės vartotojo sąsajos.

Automatizuotas testavimas leidžia smarkiai sumažinti testavimo kaštus. Taip pat automatizavimas leidžia reguliariai atlikti, to pasėkoje defektai aptinkami anksčiau.

Egzistuoja keletas įrankių, skirtų grafinei vartotojo sąsajai testuoti. Tačiau įrankiai smarkiai priklausomi nuo platformos (šiuo metu dominuojanti stacionarių kompiuterių platforma yra Microsoft Windows). Pritaikyti vienai platformai, jie netinka kitoms. Dažniausiai įrankiai veikia skirtingose aplinkose ir naudojasi skirtingais skriptais. Testavimo skriptas, įrašytas vienoje platformoje, negali būti atkartotas kitoje platformoje. Taip pat skriptas, įrašytas vieno įrankio pagalba, negali būti atkartotas kitame įrankyje [25].

Mobiliųjų įrenginių rinkoje neturime dominuojančios platformos. Tačiau turime plačiai paplitusias technologijas Java ir dotNET (1.10 Pav.). Technologija Java mobiliesiems įrenginiams J2ME 2005 metai palaikys 60% visų mobiliųjų įrenginių. Antroje vietoje po Java yra Microsoft .NET Compact Framework technologija, kuria palaikys 35% mobiliųjų įrenginių. Kiti įrenginiai palaiko kitokias programavimo technologijas arba visai jų nepalaiko [17, 45].




**1.10 Pav. Mobiliųjų technologijų plitimas mobiliuose įrenginiuose**

## 1.4. GRAFINĖS VARTOTOJO SĄSAJOS AUTOMATIZUOTO TESTAVIMO ĮRANKIŲ APŽVALGA

Šiame skyriuje apžvelgsime keletą įvairių gamintojų grafinės sąsajos testavimo įrankių bei jų teikiamą funkcionalumą [2].

### CompuWare TestPartner

TestPartner suteikia galimybę testuoti tiek grafinę sąsaja, tiek ir paprastus komponentus. Testavimui naudojami skriptai, parašyti MicrosoftVisual Basic kalba. Taip pat gali būti atliekamas komponentų testavimas naudojant įrankį. Jei testuotojai turi patirties programavimo srityje, jie gali patys pasirašyti testavimo skriptus. Taip pat galima naudoti TestPartner Visual Navigator, kuris leidžia kurti bei vykdyti grafinės sąsajos testų skriptus. Tačiau testavimo atvejai bei kontroliniai taškai turi būti rašomi testuotojo. Todėl darbui su įrankių reikalingas specialistas, galintis ne tik kurti testavimo atvejus, bet ir mokantis programuoti [15, 25].

### IBM Rational Test Tools

Firma IBM turi atskira savo veiklos atšaką, skirtą programų testavimui. Ši atšaka buvo sukurta įsigyjant įvairių programų įrangos gamintojų produktus. Grafinės sąsajos testavimui naudojami Rational Robot ir Visual Test. [22, 23, 25]

#### 1.4.A.1. Rational Robot

Rational Robot veikia Microsoft Visual Studio 6 pagrindu. Grafinės sąsajos testavimo vykdymui vartotojas turi aktyvuoti įrašymo-atkartojimo registratorių. Registratorius interpretuoja vartotojo veiksmus paversdamas juos į skriptą, koduotą specialia kalba SQABasic. SQABasic sintaksė panaši į Visual Basic. Įrankis palaiko duomenų tipus String, Integer, Long, Variant, Single, Double, Currency, o taip pat ir vartotojo apibrėžtus tipus [22].

#### 1.4.A.2. Rational Visual Test

Rational Visual Test turi vartotojo sąsają, leidžiančia vartotojams kurti testavimo programas ir valdyti testavimo rinkinius. Testavimo programos gali būti kuriamos rankiniu būdu arba naudojantis įrašymo-atkartojimo metodu Scenario Recorder pagalba. Skript kalba vadinama TestBasic. Tačiau testavimo skriptai gali būti taikomi tik programom, sukurtom Microsoft Visual Studio 6 pagrindu. Įrašymo atkartojimo modulis Scenario Recorder dažnai siūlo įvesti duomenis bei nurodyti tikrinimo vietas. Taip pat naudojama daug kietai koduotų (*hard-coded*) grafinės sąsajos koordinačių. Rational Visual Test buvo sukurtas kompanijos Microsoft. Vėliau jį įsigijo Rational Software. Dabar šitas testavimo įrankis priklauso IBM [23].

### Mercury Interactive Tools

Mercury Interactive kompanija siūlo kelis produktus grafinės sąsajos testavimui: WinRunner ir LoadRunner [34, 35].

#### 1.4.A.3. WinRunner

WinRunner tai įrankis, paremtas įrašymo-atkartojimo metodu. Jis skirtas Microsoft Windows platformai bei žiniatinkliui (*web*). Vedlio (*wizard*) pagalba grafinės sąsajos testavimui ruošiamas grafinės sąsajos žemėlapis. Žemėlapio kūrimui gali būti naudojamos objektų arba ekrano analoginės koordinates. Testavimo skriptai koduojami Test Script Language klaba. Įrankis gali naudoti išorinę duomenų bazę. Tačiau duomenų paruošimas atliekamas rankiniu būdu. Kai kuriais atvejais dalis įrankio teikiamų savybių neveikia. Taip pat įrašymo-atkartojimo modulis negali būti taikomas pilno grafinės sąsajos žemėlapio kūrimui [34].

#### 1.4.A.4. LoadRunner

Šis produktas testavimui naudoja skriptus, paruoštus kitais kompanijos Mercury Interactive įrankiais. Įrankio veikimas tapatinamas su virtualiu vartotoju. LoadRunner gerai tinka stresiniam sistemos testavimui. Jis gali emuliuoti kelis šimtus lygiagrečiai prie sistemos prisijungusių

vartotojų. Testavimo metu galima išmatuoti sistemos darbo charakteristikas, nustatyti svarbių verslo procesų siauras vietas (*bottleneck*). Lygiagrečiai su stresiniu sistemos testavimu LoadRunner matuoja atskirų sistemos komponentų darbo charakteristikas. LoadRunner palaiko Microsoft Windows ir Unix platformas [35].

### SilkTest

SilkTest – tai Kompanijos Segue grafinės sąsajos testavimo produktas. Šis įrankis skirtas Microsoft Windows platformas ir leidžia testuoti objektus bei grafinės sąsajos komponentus sukurtus standartinės Microsoft Foundation Class bibliotekos pagrindu [39]. SilkTest veikimas paremtas įrašymo-atkartojimo metodu. Taip pat įrankis teikia keletą vedlių, leidžiančiu rankiniu būdu įtakoti testuojamą programą. Vartotojui pasirinkus grafinį komponentą, įrankis analizuoja jo savybes: nustato komponento atributus bei išdėstymo koordinates. Testavimo skripto sudarymo metu, vartotojo veiksmai apdorojami ir koduojami skripto kalba 4Test. Skripto sudarymo metu, vartotojas turi patvirtinti komponentų grafinius atvaizdavimus (*bitmaps*) bei patikros taškus vedlių pagalba. Praktiškai testavimo duomenys bei kodas yra ruošiami rankiniu būdu.

Dabar SilkTest turi keletą produktų atšakų. Produktai skiriasi palaikomomis technologijoms (Java, .NET) bei testavimo metodais (funkcinis, regresinis, stresinis, apkrautumo) [44].

### Atviro kodo grafinės sąsajos testavimo įrankiai

Dauguma atviro kodo grafinės sąsajos testavimo įrankiu skirti Java programų kūrėjams. Testuotojai kuria naudingus įrankius atviro kodo bendrijoms . Apžvelgsime keletą tokių įrankių [2].

#### 1.4.A.5. Abbot

Abbot - Timothy Wall kompanijos grafinės vartotojo sąsajos testavimo įrankis. Abbot aplinkos pagalba galima atlikti Java programų grafinės sąsajos komponentų testavimą bei funkcinį testavimą . Testavimo skriptai naudojasi specialios Java bibliotekos metodais, kurie leidžia atkartoti vartotojo veiksmus bei analizuoti grafinės vartotojo sąsajos komponentų būsenas. Testavimo skriptai rašomi Java kalba. Taip pat Abbot turi sąsajas, leidžiančias jungti integravimo bei funkcinį testavimą su grafinės sąsajos testavimu. Įrankis yra nemokamas (*freeware*) [56].

#### 1.4.A.6. GUITAR

GUITAR tai UMIACS universiteto iš Maryland kuriamas grafinės vartotojo sąsajos testavimo įrankis [27]. GUITAR pateikia jungtinį grafinės sąsajos testavimo sprendimą. Įrankio veikimas paremtas įvykių apdorojimu (*event-based*) [27, 33]. Įrankį sudaro keli moduliai: testavimo atvejų generatorius [29, 31, 32], skriptų vykdymo modulis [30, 33], regresinio testavimo modulis. Įrenginys bando iš galimų grafinės sąsajos įvykių aibės išskirti pagrindinius testavimo scenarijus: pirmoje stadijoje atliekama grafinės sąsajos dekompozicija - išskiriami atskiri komponentai, vėliau sudaromas komponentų grafas. Iš komponentų grafo nustatomas testavimo scenarijai. Testavimo scenarijų sudarymas paremtas įvykių padengimo kriterijumi [27, 28, 33, 52]

#### 1.4.A.7. qftestJUI

Grafinės sąsajos testavimo įrankis qftestJUI yra sudarytas Java pagrindu. Todėl jis veikia tiek Windows tiek Unix sistemose. Testavimo skriptai taip pat koduojami Java kalba. Java programų kūrėjai dažnai jį naudoja kartu su Java kūrimo įrankiais nuo IBM bei SUN. qftestJUI naudojamas testavimo atveju kūrimui, vykdymui bei tvarkymui [4].

### Grafinės sąsajos testavimo įrankių apibendrinimas

Iš skyriuje 1.4 Grafinės vartotojo sąsajos automatizuoto testavimo įrankių apžvalga patiekti testavimo įrankiai atspindi dabartines grafinės sąsajos testavimo tendencijas. Testavimas atliekamas vykdant testavimo skriptus. Testavimo skriptai gali būti koduojami rankiniu būdu arba įrašomi, testuotojui dirbant su sąsaja taip pat rankiniu būdu. Testavimo skriptų kodavimui naudojamos įvairios kalbos. Keletas skriptų koduojami programavimo kalbomis. Įrašyti testavimo skriptai tai tik vartotojo veiksmų seka. Sprendimą dėl testo rezultatų priima testuotojas.

Testavimo skripto įrašymo metu įrankiai dirba su grafiniu programos atvaizdavimu, fiksuojamos klaviatūros bei pelės įvykių koordinatės, tačiau nenusileidžiama į žemesnį lygį (1.11 Pav.). Kai kurie įrankiai gali dirbti ir grafinių komponentų lygyje. Tuomet testavimo metu įrankiai analizuoja grafinius komponentus ir jų atributus. Testavimo duomenys gali būti nuskaityti iš išorinių duomenų saugyklų. Testavimo atvejų generavimas priklauso tik nuo testuotojų.

Tačiau net atliekant komponentų testavimo, nėra realizuojamas grafinės sąsajos testavimas taikant galinio vartotojo požiūrį. Sąsajos testavimas nėra siejamas su programos funkcionalumo testavimu. Automatizuojamas tik testavimo žingsnis. Kiti žingsniai atliekami rankiniu būdu, kuris

taip pat gali įvelti klaidų. Testuotojo darbas yra orientuotas į programuotojus, kadangi skriptų redagavimui reikalingi programavimo įgūdžiai.

Ne vienas iš įrankių neorientuotas į grafinės sąsajos testavimą mobiliems įrenginiams.

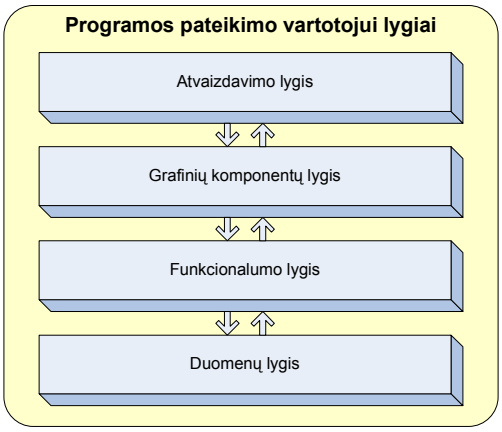
GUITAR įrankis paremtas grafinių įvykių apdorojimu [25, 27]. Taip pat yra automatizuojami testavimo atvejų generavimo bet testavimo analizės žingsniai.

## 1.5. GRAFINĖS VARTOTOJO SĄSAJOS PATIEKIMO LYGIAI.

Testavimo įrankių apžvalgoje (skyrius 0) nustatėme, kad įrankiai sąveikauja su skirtingais programos pateikimo lygiais (1.11 Pav.). Vieni įrankiai interpretuoja programos grafinę sąsaja kaip grafinį vaizdą, neišskiriant jame grafinių komponentų. Kiti įrankiai remdamiesi konkrečios technologijos programine sąsaja sąveikauja su grafiniais komponentais. Atskirais atvejais mėginama grafinės sąsajos testavimą jungti su funkciniu testavimu, taip pritaikant galinio vartotojo požiūrį į programą.

Išskirsime testavimo įrankių ir programos grafinės sąsajos sąveikavimo lygmenys:

- Atvaizdavimo lygis – programos pateikimas ir interpretavimas kaip grafinio atvaizdavimo, neišskiriant jame grafinių komponentų.
- Grafinių komponentų lygis – iš programos grafinės sąsajos išskiriami grafiniai komponentai. Analizuojamos grafinių komponentų savybės.
- Funkcionalumo lygis – programos grafinė sąsaja analizuojama ne tik kaip grafinių komponentų aibė, bet ir kaip sąsaja, kurios pagalba galima kviesti bei vykdyti testuojamos programos funkcionalumą. Šiame lygyje gali būti vykdomas integravimo testavimas.
- Duomenų lygis – šiame lygyje sekamas ne tik grafinės vartotojo sąsajos kviečiamas bei vykdomas funkcionalumas, taip pat yra atliekamas ir vienetų testavimas.




**1.11 Pav. Programos pateikimo vartotojui lygiai**

## 1.6. PAKARTOTINAS TESTAVIMAS

Pakartotinas testavimas (*Regression testing*) – tai pakartotinas testavimo atvejo vykdymas. Pakartotinai vykdomi visi testavimo atvejai, nepriklausomai nuo to, koks buvo vykdymo rezultatas: sėkmingas ar ne. Pakartotino testavimo metu norima įsitikinti, kad programos pasikeitimai nepakenkė veikiančioms programos dalims.

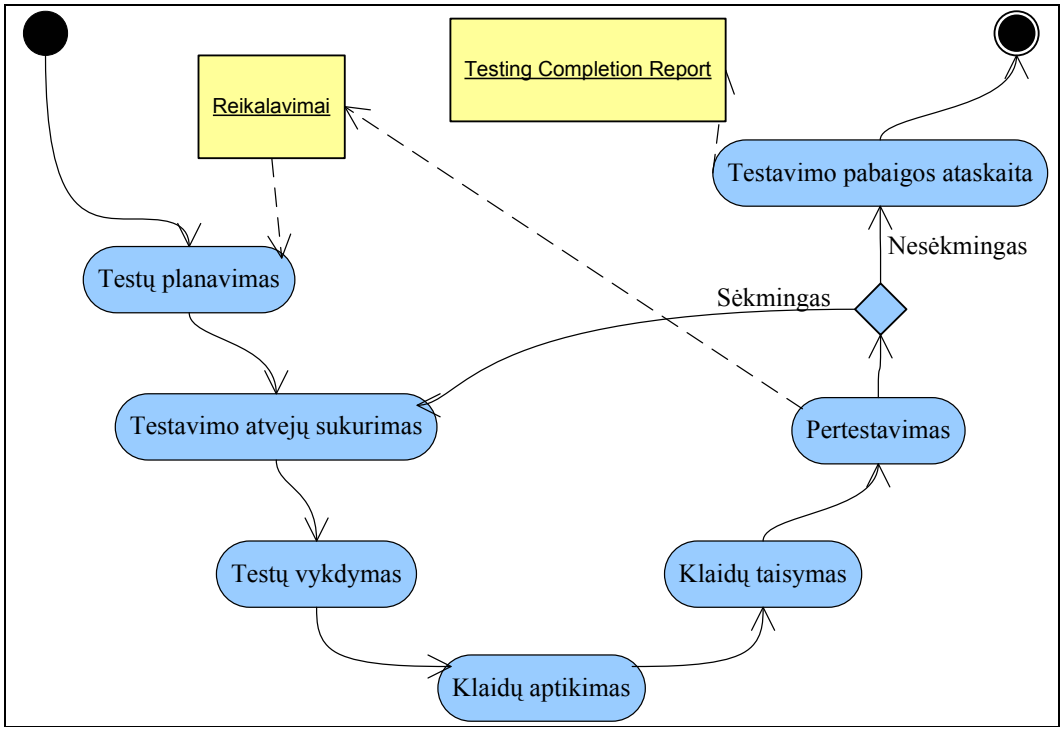
Baigus testuoti programą pagal testavimo skriptą, tikimybė, kad pakartotinai vykdant testavimą pagal tą patį skriptą, yra labai maža. Testavimo skriptai nepadės aptikti defektų programoje jei testavimui bus naudojama ta pati aibė duomenų. Testavimo skriptai padeda aptikti programos defektus, jei jiems vykdyti naudojamos skirtingos duomenų aibės. Testuotojams tenka skirti daug laiko testavimo atvejų bei duomenų parengimui, kad šie galėtų padengti visas programos šakas.

Programos turi tendenciją keistis programos kūrimo proceso metu. Bet kokie programos išeities kodo pakeitimai gali neigiamai paveikti programos funkcionalumą bei charakteristikas. Tai yra pagrindinė priežastis, dėl kurios testai turi būti atkartojami. Taip pat regresinis testavimas atliekamas norint įsitikinti, kad aptiktos klaidos buvo ištaisytos [20, 57].

## 1.7. TESTAVIMO PROCESO ŽINGSNIAI

Programų kūrimo projekto metu išdirbėjai pereina įvairius testavimo žingsnius. Testavimo procese išskiriamos šios fazės: testavimo planavimas, testavimo atvejų bei duomenų paruošimas, testų vykdymas, defektų aptikimas, klaidų taisymas, pertestavimas, testavimo pabaigos ataskaita (1.12 Pav.) [7, 21]. Testavimas yra iteracinis procesas. Pagrindiniai iteracijos žingsniai yra testavimo atvejų bei duomenų kūrimas, testų vykdymas ir testavimo rezultatų

nustatymas. Šių žingsnių automatizavimas leis pagreitinti testavimo darbus, atlikti pakartotiną testavimą, išplėsti testavimo darbų aibę [18].




**1.12 Pav. Testavimo proceso žingsniai**

## 1.8. SKRIPTŲ KODAVIMAS

Testavimo įrankių apžvalgos metu nustatėme, kad testavimo skriptams koduoti naudojamos įvairios kalbos. Tačiau tokių skriptų nepavyksta taikyti kituose įrankiuose. Skriptų kodavimo būdas neturi būti pririštas prie konkretaus testavimo įrankio ir leisti keistis testavimo atvejais bei duomenimis su kitais įrankiais.

### XML kalba

XML yra „žymėjimo“ kalba dokumentams, kuriuose saugoma struktūrizuota tekstinė informacija [1, 3, 6].

### XMP sintaksės analizatoriai

XML sintaksės analizatoriai (*XML parsers*) apdoroja XML dokumentus, nustato jų struktūrą bei duomenų savybes. Dokumento apdorojimo metu išskiriamos atskiros duomenų dalys, kurios vėliau perduodamos kitiems programos komponentams. XML analizatoriai gali atlikti dokumento struktūros teisingumo patikrinimą [6]. Pagal dokumento apdorojimo būda, XML sintaksės analizatoriai skirstomi į dvejas grupes: DOM ir SAX

#### 1.8.A.1. DOM

DOM (*Document Object Model*) tipo XML analizatorių veikiamas paremtas medžio struktūros sąsaja (*tree-based interface*). Dokumento apdorojimo metu jo struktūra nuskaitoma į medžio tipo duomenų struktūrą. Navigacija bei duomenų apdorojimas medžio tipo struktūroje vyksta greitai ir nėra apriboti elementų sekos tvarka.

Metodas reikalauja didelių atminties bei procesoriau resursų. Šis metodas tinka norint nuskaityti duomenys ir pasinaudoti jais kelis kartus. Mobiliuose programose šis metodas gali tapti siaura vieta (*bottleneck*) apdorojant didelius duomenų kiekius.

#### 1.8.A.2. SAX

SAX (*Simple API for XML*) tipo XML analizatorių veikimas paremtas įvykių apdorojimo sąsaja (*event-based interface*). Dokumento apdorojimo metu, analizatoriui aptikus elemento pradžios žymę (*start tag*), iškviečiamas elemento apdorojimo metodas `startElement()`.

Metodas nereikalauja didelių atminties bei procesoriaus resursų. Tačiau dokumento elementai gali būti atliktai tik nuosekliai. Šis metodas yra naudingas, norint apdoroti dokumentą tik vieną kartą.

# 2. PROJEKTINĖ DALIS

Šioje dalyje apžvelgsime magistratūros studijų metu sukurtą programinę įrangą – Mobilaus vadybininko darbo vieta.

Projektas yra sudėtinė magistrinio darbo dalis. Projekto rezultatai buvo pristatyti konferencijose [40, 41, 54, 55, 53].

## 2.1. SUPROJEKTUOTA SISTEMA

Mobiliosios technologijos – tai būdas verslo įmonėm padidinti savo veiklos našumą. Taikant mobiliąsias technologijas kasdieniniame veiklos procese įmonė padidina personalo darbo efektyvumą, sumažina verslo procesų kaštus bei prideda papildomos vertės prie savo produktų, nekeičiant jų kainos. Mobilioji darbo vieta, tai būdas operuoti visa reikalinga informacija realiame laike nutolus nuo darbo vietos. Mobilioji darbo vieta gali būti plačiai taikoma įvairiose veiklos srityse.

Apsiribojant vadybininko veiklos sritimi galima teigti, kad naudojant mobiliąją darbo vietą, mobilusis vadybininkas galės pasiekti visą jam reikiamą informaciją realaus laiko režime, valdyti pavaldų personalą, apdoroti veiklos procesus, pritraukti klientus greičiau patenkinant jų poreikius.

### Sistemos aprašymas

„Mobilaus vadybininko darbo vieta“ programinė įrangos vartotojas gali atlikti visus darbus naudodamasis mobiliuoju įrenginių tokiu kaip kišeninis kompiuteris, mobilusis telefonas ar nešiojamas kompiuteris. Naudodamasis programine įranga bei mobiliuoju įrenginių mobilusis vadybininkas gali gauti, apdoroti bei siusti visą jam reikiamą informacija realiame laike. Mobilusis vadybininkas naudodamasis mobiliuoju įrenginiu galės naudotis programinės įrangos teikiamomis funkcijomis:

- Darbas su ataskaitomis
- Duomenys apie klientus
- Duomenų sinchronizavimas
- Užduočių atmintinės tvarkymas
- Duomenys apie prekes

Produkto pagalba vartotojas gali greitai pasiekti reikalingą informaciją, nepriklausomai nuo jo buvimo vietos bei laiko.

Sprendimas yra skirtas verslo įmonėms, kurios intensyviai bendrauja su savo klientais, ieško naujų klientų bei siekia efektyviau atlikti savo veiklą naudojant mobiliąsias technologijas

**Tikslas**

Dauguma verslo įmonių jau turi informacijos tvarkymo programas, tačiau reta įmonė sugeba sėkmingai konkuruoti be mobiliosios sistemos. Įmonių vadybininkai dažnai susiduria su šiomis problemomis::

- Naujos informacijos stoka
- Pakartotinas duomenų pildymas, gaištantis laiką
- Klaidos atsirandančios informacijos perrašymo metu
- Nesugebėjimas greitai reaguoti į klientų užklausas

Projekto tikslas yra sukurti mobiliosios darbo vietos programinę įrangą. Mobilusis vadybininkas galės atlikti visus darbus naudodamasis mobiliuoju įrenginių tokiu kaip kišeninis kompiuteris, mobilusis telefonas ar nešiojamas kompiuteris. Naudodamasis programine įranga bei mobiliuoju įrenginių mobilusis vadybininkas galės gauti bei apdoroti visą jam reikiamą informacija realiame laike. Bendraujant su klientais mobilusis vadybininkas galės pateikti naujausią informaciją apie konkrečių užsakymų atlikimą. Mobilusis vadybininkas naudodamasis mobiliuoju įrenginiu galės naudotis programinės įrangos teikiamomis funkcijomis:

- Atlikti rezervavimą
- Atlikti užsakymą
- Atlikti pardavimą
- Peržiūrėti rezervavimus
- Peržiūrėti užsakymus
- Peržiūrėti pardavimus
- Peržiūrėti sandelio prekes
- Redaguoti rezervavimą
- Redaguoti užsakymą
- Gauti gamybos padalinio veiklos ataskaitas
- Gauti informaciją apie mokėjimus
- Prognozuoti konkretaus užsakymo atlikimo terminus
- Gauti reikiamą informaciją apie klientą (*Customer Relationship Management*)

Mobiliosios priemonės padės mobiliajam vadybininkui greičiau sužinoti jam reikiamą informaciją, nepriklausomai nuo jo buvimo vietos bei laiko.

Sprendimas yra skirtas verslo įmonėms, kurios intensyviai bendrauja su savo klientais, ieško naujų klientų bei siekia efektyviau atlikti savo veiklą naudojant mobiliąsias technologijas.

## Problemos sprendimas pasaulyje

Yra keletas komercinių realizacijų, kurios mobiliųjų technologijų pagalba įgyvendina mobiliosios darbo vietos principus.

### 2.1.A.1. M–partner

Mobilių agentų informacijos sistema didmeninės prekybos įmonėms ir kitoms organizacijoms, savo veiklą grindžiančiomis reguliariu klientų lankymu užsakymų suformavimui ir kitos informacijos surinkimui [9].

Užsakymo perdavimui į centrinę įmonės duomenų bazę ir naujausios informacijos apie prekes bei klientą gavimui naudojamas bevielis duomenų perdavimas.

Sistemoje yra įdiegtos agento darbo planavimo ir jo kontrolės, marketingo valdymo bei kitos funkcijos, leidžiančios didinti darbo efektyvumą, gerinti paslaugų kokybę ir mažinti įmonės sąnaudas. Sistemą “m-partner” sudaro kliento dalies ir serverio dalies programos, kurios atitinkamai yra įdiegiamos agentų kišeniniuose kompiuteriuose ir įmonės centriniame kompiuteryje. “M-partner” serveris užtikrina duomenų apsikeitimą tarp “m-partner” kliento programų ir įmonės finansinės apskaitos sistemos. Duomenų apsikeitimui tarp “m-partner” kliento ir serverio programų naudojamas bevielis duomenų perdavimas. Duomenų perdavimui naudojama bevielė GPRS duomenų perdavimo technologija. Agentų išsiųsti užsakymai automatiškai įrašomi į apskaitos sistemą. Agentų vadovai, administratoriai ir analitikai gali jungtis prie “m-partner” serverio ir administruoti bei analizuoti jame saugomus duomenis.

### 2.1.A.2. M–Calculatror

Sprendimas gyvybės draudimo agentams. Programa apskaičiuoja draudimo įmoką pagal draudimo agento nurodytus kliento parametrus [8].

M-skaičiuoklė gyvybės draudimo agentams - tai kišeniniame kompiuteryje realizuotas sprendimas agentui operatyviai parinkti kliento poreikius tenkinantį draudimo variantą. Programa apskaičiuoja draudimo įmoką pagal nurodytą draudimo rūšį, įmokų

periodiškumą, draudimo laikotarpį, apdraustojo lytį ir amžių, draudimo sumą bei kitus parametrus.

#### 2.1.A.3. Mobile Order Taker

Sprendimas mobiliajam vadybininkui [10]. Pagrindinė programos užduotis yra užsakymų formavimas. Ši programinė įranga leidžia atlikti šiuos veiksmus:

Gauti įmonės siūlomų produktų sąrašus

Sužinoti apie produktų kainas, vykstančias akcijas bei nuolaidas

Nuotolinis užsakymo sudarymas, operuojant visa reikiama informacija

Klientas gali pasirašyti užsakymą tiesiai delninio kompiuterio ekrane

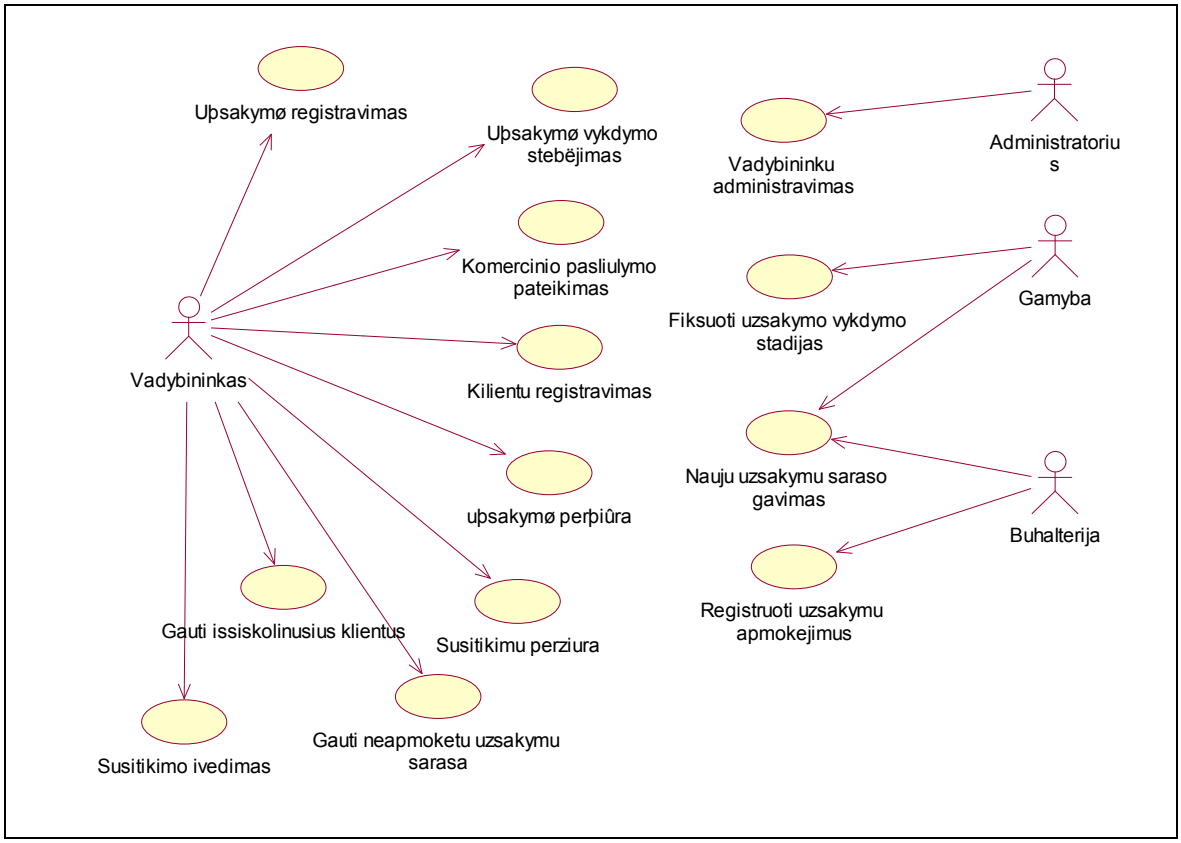
### Situacijos Lietuvoje įvertinimas

Mobiliųjų įrenginių funkcinės galimybės sparčiai plečiasi. Lietuvos rinkoje galima įsigyti naujausius įrenginius, kurių kainos yra priimtinos ne tik verslo įmonėms, bet ir nemažai daliai fizinių asmenų. Nors mobilusis įrenginys tampa vis labiau įprastas mūsų gyvenimas ir iš prabangos atributo pereina į kasdieninį įrankį, tačiau pačio įrenginio funkcinės galimybės, jo dizainas bei firmos, pagaminusios įrenginį, vardas pabrėžia įrenginio savininko prestižą, jo statusą bendruomenėje o įmonės atveju – jos finansines galimybes. Sparti mobiliųjų įrenginių plėtra siejama su bevielių tinklų plėtra. Prognozuojama kad mobilūs tinklai pralenks vielinius tinklus pagal savo panaudojamumą, ypač tai budinga vietovėms, kur įprasti tinklai yra sunkiai sudaromi arba ja visai nėra. Mobiliųjų sprendimų verslui saugumas yra taip pat aktualus kaip ir bet kokiam kitam sprendime.

## Sprendimo realizacija

#### 2.1.A.4. Panaudojimo atvejai

Programinė įranga padės mobiliajam vadybininkui efektyviau atlikti kasdieninį darbą. Programinės įrangos panaudojimo atvejai (*use cases*) pateikti panaudojimo atvejų diagramoje (2.1 Pav.).




**2.1 Pav. Panaudojimo atvejų diagrama**

### 2.1.A.5. Programų sistemos funkcijos

Vadybininkai, naudodamiesi programinę įranga ir delniniais kompiuteriais, galės naudotis sistemos teikiamomis funkcijomis:

- Užsakymų registravimas – Naudodamasis delniniu kompiuteriu vadybininkas galės lengvai užregistruoti naują užsakymą, nurodant visą reikalingą informaciją
- Užsakymo vykdymo stebėjimas – Bet kokiu laiko momentu galima susižinoti apie konkretaus užsakymo būsena ir apie tai informuoti klientą
- Komercinio pasiūlymo pateikimas – Vykstant į susitikimą su klientų galima pasiruošti susitikimui kelyje, gavus visą reikiamą informaciją apie produktų bei paslaugų kainas, nuolaidas, akcijas bei kliento taškus
- Klientų registravimas – Įvesti bei koreguoti kliento duomenys.
- Užsakymų peržiūra – Peržiūrėti užsakymus analizuojant rinkos poreikius.
- Susitikimų peržiūra – Informacijos apie klientą surinkimas (*Customer Relationship Management*).

- Susitikimo įvedimas – Tvarkaraštis (*organizer*), kuriame nurodoma visa svarbi susitikimui informacija.
- Gauti įsiskolinusių klientų sąrašą – Matyti įsiskolinusių klientų sąrašą.
- Gauti neapmokėtų užsakymų sąrašą – Matyti neapmokėtų užsakymų sąrašą.

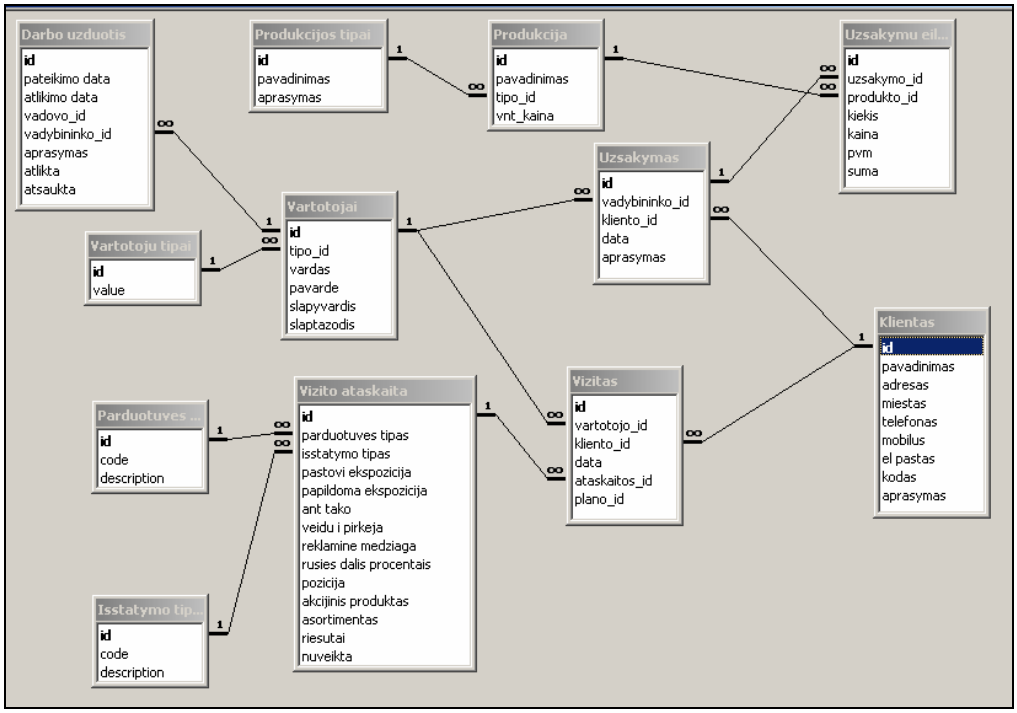
Kiti įmonės darbuotojai galės atlikti tokias funkcijas

- Administruoti mobiliuosius vadybininkus (Administratorius) – Įveda, koreguoja bei šalina sistemos mobiliuosius vadybininkus. Administruoja vadybininkų teises.
- Fiksuoti užsakymo vykdymo stadijas (Gamintojas) – Gamybos metu fiksuoja užsakymo atlikimo stadijas.
- Naujų užsakymų sąrašo gavimas (Gamintojas, buhalteris) – Galimybė matyti visus užsakymus, būti pasiruošusiam naujam darbui.
- Registruoti užsakymų apmokėjimus (Buhalteris) – Registruoti atliktus mokėjimus sistemoje bei buhalterijos modulyje.

### Duomenų saugyklos realizacija

Mobiliuose klientuose bus naudojama Java Record Management System [54]. Sistemos pagrindinės darbo stoties (*server*) duomenų bazės valdymui yra pasirinkta Microsoft SQL Server 2000 Desktop Engine duomenų bazės valdymo sistema.

Projekto duomenų bazės schema pateikta žemiau (2.2 Pav.).




**2.2 Pav. Duomenų bazės schema**

Duomenų bazėje pateiktų esybių aprašymai pateikti lentelėje Lentelė 2.1 Duomenų bazės lentelių aprašymas.

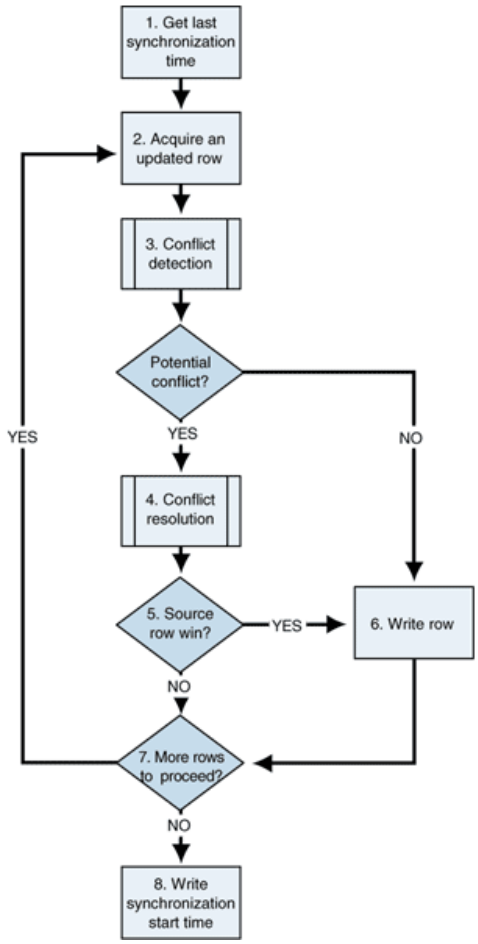
**Lentelė 2.1 Duomenų bazės lentelių aprašymas**

| # | Pavadinimas | Aprašymas |
|---|---|---|
| 1. | Darbo užduotis | Saugoma informacija apie vadybininkui patiektas darbo užduotis |
| 2. | Vartotojo tipas | Saugomi vartotojų tipai |
| 3. | Parduotuvės tipas | Saugomi parduotuvių tipai |
| 4. | Išstatymo tipas | Saugomi prekių išstatymo tipai |
| 5. | Produkcijos tipas | Saugomi produkcijos tipai |
| 6. | Vartotojas | Saugoma informacija apie vartotojus |
| 7. | Vizito ataskaita | Saugomos vadybininkų vizitų ataskaitos |
| 8. | Produkcija | Saugoma informacija apie produkciją |
| 9. | Užsakymas | Saugoma informacija apie sudarytus užsakymus |
| 10. | Vizitas | Saugoma informacija apie vizitus pas klientus |
| 11. | Užsakymo eilutė | Užsakymų eilutės |
| 12. | Klientas | Saugoma informacija apie klientus |

Mobiliuose klientuose bus naudojama Java Record Management System [54].

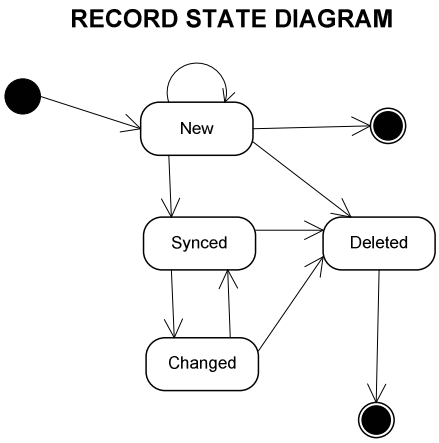
Duomenų sinchronizavimui tarp kliento ir serverio naudojamas „Master-Master Row-Level Synchronization" šablonas [36]. Sinchronizavimo vienetas yra įrašas. Įrašai žymimi sukūrimo/modifikavimo laiku bei vėliavėle „ištrintas". Todėl kiekviena duomenų bazės lentelė turi pora papildomų stulpelių. Sinchronizavimo metu eilutės, pažymėtos

kaip ištrintos („ištrintas“ = TRUE), praleidžiamos. Papildomi stulpeliai neskaitomi. Lyginamos laiko atžymos tarp šaltinio (*source*) ir adresato (*destination*) ir įrašas su vėlesne data laimi. Sinchronizavimo algoritmas pateiktas žemiau (2.3 Pav.).




**2.3 Pav. Duomenų sinchronizavimo algoritmas**

Duomenų sinchronizavimo realizavimui kiekvienas įrašas mobiliajame įrenginyje turi savo būseną. Įrašo būsenų keitimas patiektas būsenų diagramoje (2.4 Pav.).



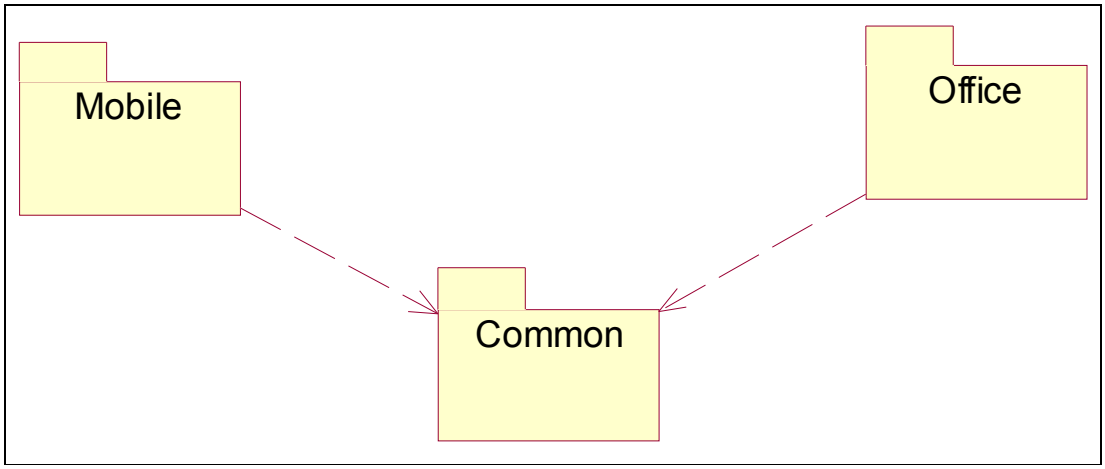

**2.4 Pav. Įrašo būsenų diagrama**

## 2.2. ARCHITEKTŪRA

Šiame skyriuje pateikiamas sistemos išskaidymas į posistemės ir paketus. Detalesnis sistemos aprašymas.

### Klasių diagramos

Sistema susideda iš pagrindinių paketų: Mobile, Common, Office (2.5 Pav.).

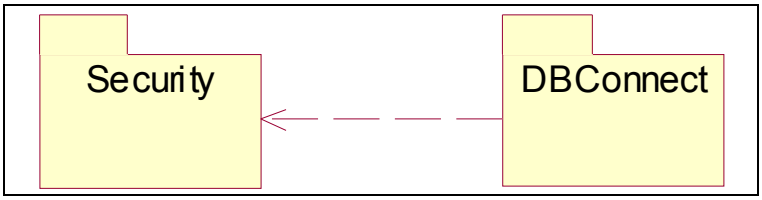



**2.5 Pav. Pagrindiniai sistemos paketai**

#### 2.2.A.1. Paketas Common

Pakete apibendrina visai sistemai bendras funkcijas. Pakete esančios klasės realizuoja operacijas su duomenų baze bei užtikrina sistemos saugumą. Pagal pagrindinius funkcionalumus, pakete išskiriami žemesnio lygio paketai: Security ir DBConnect.

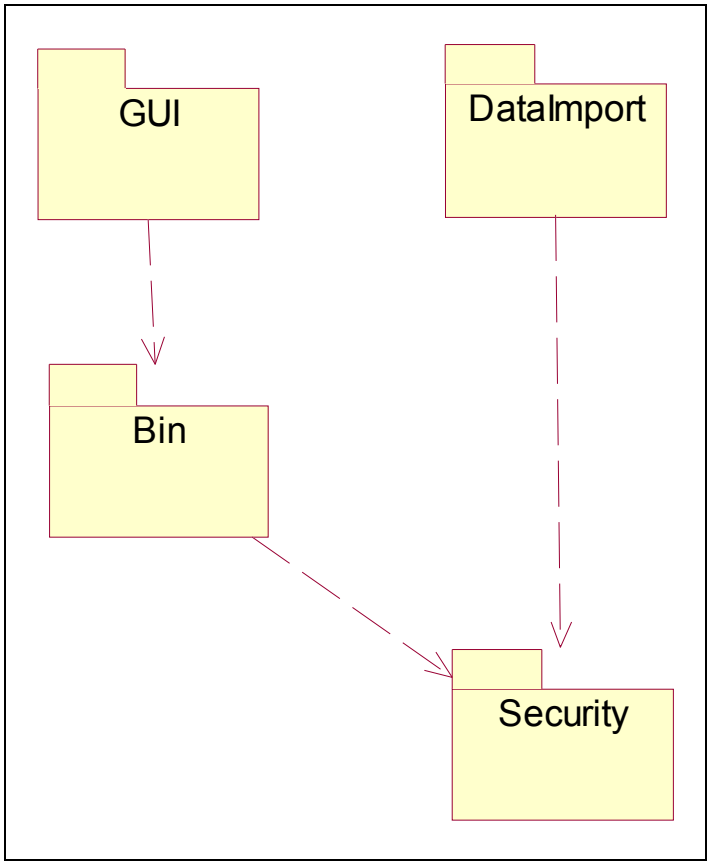
Paketo Common suskirstymas į žemesnio lygio paketus pateiktas žemiau (2.6 Pav.).




**2.6 Pav. Paketo Common detalizavimas**

### 2.2.A.2. Paketas Office

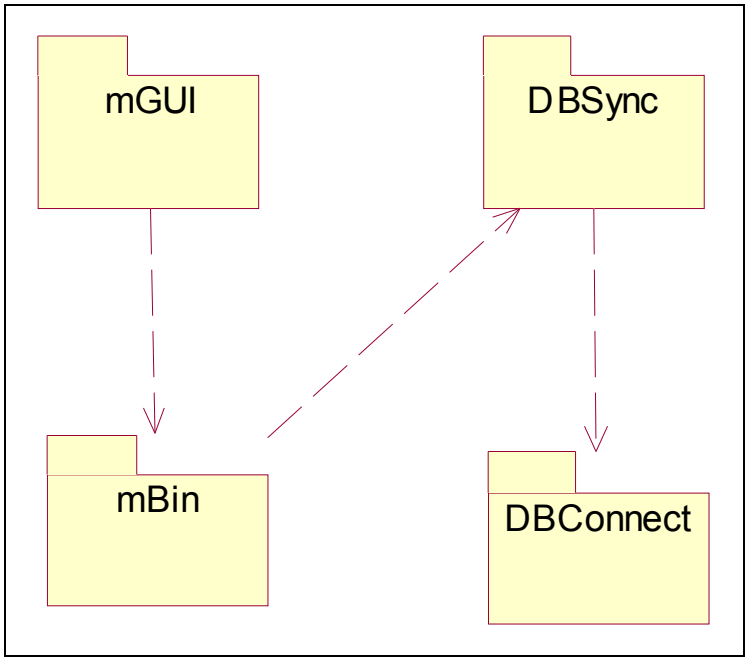
Paketas apima savyje klases bei paketus, kurių funkcionalumas siejasi su stacionaria sistemos dalimi. Pakete pateiktos klasės skirtos bendradarbiavimui su sistemos aktoriais: vadovais ir administratorius. Šios klasės realizuoja grafinė vartotojo sąsają bei patį loginį funkcionalumą. Paketas suskaidytas į žemesnio lygio paketus, kurių klasių funkcionalumo realizacija remiasi pakete Common pateiktomis klasėmis. Paketo Office suskirstymas į žemesnio lygio paketus pateiktas žemiau (2.7 Pav.).




**2.7 Pav. Paketo Office detalizavimas**

### 2.2.A.3. Paketas Mobile

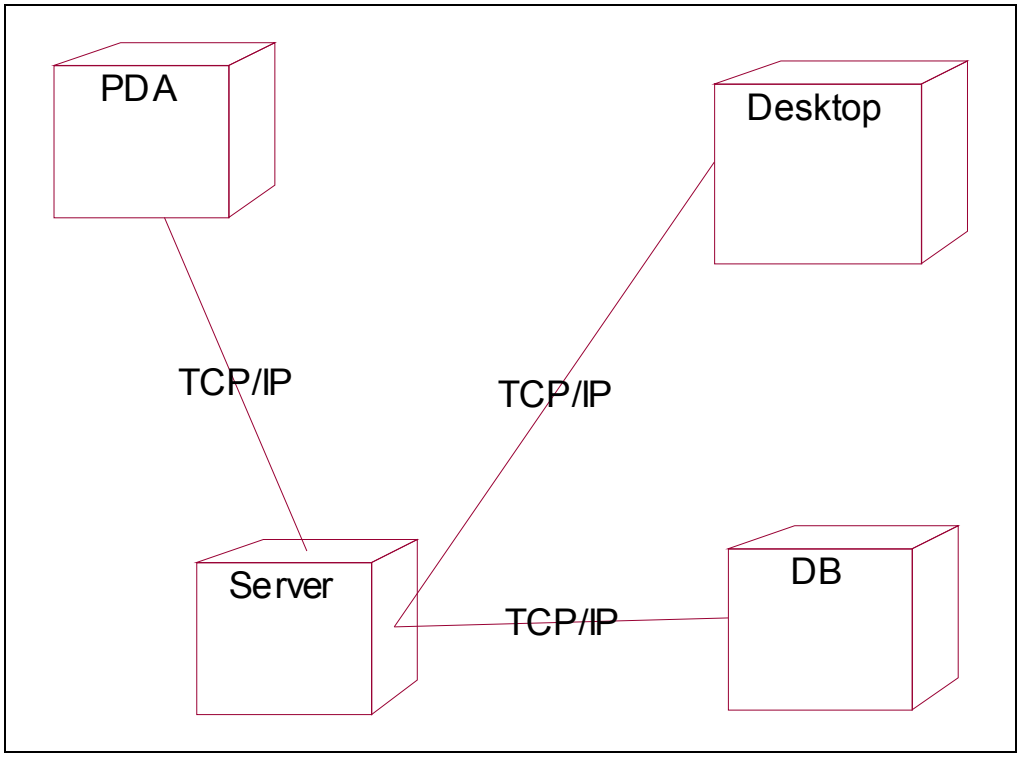
Paketas Mobile atitinka sistemos mobiliosios sistemos dalies funkcijas. Jame pateiktos klasės skirtos duomenų mainams su vartotoju, lokalios duomenų bazės tvarkymo bei sinchronizavimu su pagrindine sistemos duomenų baze. Paketas Mobile suskaidytas į žemesnio lygio paketus, atitinkančius detalesnes mobiliosios sistemos dalies funkcijas. Paketo Mobile suskaidymas į žemesnio lygio paketus pateiktas žemiau (2.8 Pav.).




**2.8 Pav. Paketo Mobile detalizavimas**

## Išdėstymo diagrama

Sistemos dalys (klientas, serveris) yra fiziškai paskirstytos skirtinguose vietose. Žemiau pateikta sistemos išdėstymo diagrama 2.9 Pav..




**2.9 Pav. Išdėstymo diagrama**

### 2.2.A.4. DB

Duomenų bazė diegiama Microsoft Windows XP ® operacinėje sistemoje. Duomenų bazės valdymo sistemai naudojama Microsoft SQL Server Desktop Engine [37].

Minimalus reikalavimai aparatūrai:

- CPU: 133 MHz
- RAM kiekis: 128 MB
- Kietojo disko talpa: 2 GB
- Tinklo korta prisijungimui prie tinklo

### 2.2.A.5. PDA

Sistemos mobilioji kliento dalis, veikianti delniniame kompiuteryje, yra daugiaplatformė. Mobiliojo kliento įrenginio charakteristikos:

| | |
|---|---|
| Procesorius: | 400MHz Intel® XScale™ procesorius |
| Operacinė sistema: | Microsoft® Windows® Mobile 2003 |
| Atmintis: | 64MB SDRAM (56MB vartotojui pasiekiama), 32MB Flash ROM |
| Ekranas: | 3.5" 240 x 320 16-bitų spalvos |

Programinė įranga realizuojama naudojant J2ME platformą. Kompiuteryje turi būti įdiegta JVM [24].

#### 2.2.A.6. Server

Aptarnauja klientų užklausimus ir tarpininkauja jiems dirbant su duomenų baze. Serverio sistemos pusė veiks tame pačiame kompiuteryje kaip ir duomenų bazė. Kadangi serverio sistemos pusė taip pat turi būti nepriklausoma nuo platformos, ji bus realizuota J2SE platformoje. Kompiuteryje turi būti įdiegta JVM.

#### 2.2.A.7. Desktop

Stacionarus sistemos klientas yra surištas su Windows operacine sistema, todėl sistemos funkcijos bus prieinamos per COM komponentus, įdiegtus vartotojo kompiuteryje. Kompiuteris turi būti prijungtas prie tinklo.

## 2.3. DIEGIMAS

Produkto diegimas aprašytas vartotojo vadove. Programos diegimas mobiliajame įrenginyje yra prieinamas sistemos galiniam vartotojui. Dėl mobiliųjų įrenginių apribojimu buvo stengiamasi maksimaliai supaprastinti programinės įrangos diegimo ir atnaujinimo procesą. Todėl diegimas yra atliekamas pasinaudojant mobilaus įrenginio bevielio ryšio tinklo priemonėmis. Diegiant pirmą kartą, vartotojui reikia nurodyti sistemos serverio adresą. Norint atnaujinti programą, vartotojui reikia per mobilaus įrenginio operacinę sistemą aktyvuoti programos atjauninimą. Tolimesnis diegimas vyksta automatiškai.

Serverio diegimas yra smulkiai aprašytas sistemos instaliavimo vadove.

# 3. TYRIMO DALIS

Šioje dalyje apžvelgsime siūlomą grafinės vartotojo sąsajos mobiliems įrenginiams testavimo metodą.

Magistratūros studijų metu buvo vykdomas programinės įrangos mobiliai įrangai kūrimo projektas. Projekto pasėkoje buvo numatytos programų kūrimo procesą gerinančios galimybės. Šių galimybių realizavimui siūlomas grafinės sąsajos mobiliems įrenginiams testavimo metodas.

## 3.1. APIBRĖŽIMAI

- Grafinės sąsajos komponentai skirstomi į šias grupes (skyrius 1.2):
    - Meniu
    - Iššokantys meniu
    - Teksto įvedimo laukai
    - Jungikliai
    - Radijo mygtukai
    - Mygtukai
    - Progreso indikatoriai
    - Dialogai
    - Langai
- Programos lange atvaizduojami grafinės vartotojo sąsajos elementai. Grafiniai elementai tai: mygtukai, įvedimo laukai, pateikimo laukai, paveikslėlių langai, pasirinkimo sąrašai. Lango elementų skaičius yra baigtinis.
- Programos langas gali turėti kelias būsenas. Lango būsena nusako kurie lango elementai turi būti vaizduojami. Lango būsenų skaičius yra baigtinis.
- Iš programos langų sudaromas navigacijos grafas, kuriame mazgas atitinka programos langą, o briauna atitinką perėjimą tarp langų.
- Lango komanda atitinką veiksmą, kuri programos vartotojas gali atlikti programos lange. Lango komandų skaičius yra baigtinis.
- Langų seka LS = $\{l_1, l_2, ... l_n\}$, per kurią vartotojas pereina, atlikdamas operaciją, vadinamas operacijos scenarijumi (uždaras scenarijus kai $l_1 = l_n$, priešingai scenarijus atviras). Tą pačią operaciją gali aprašyti keli scenarijai, tačiau kiekvienu atveju parametrų reikšmės bus skirtingos [12].

## 3.2. PROBLEMOS

Grafinės vartotojo sąsajos ir programos operacijų scenarijų testavimas mobiliems įrenginiams atliekamas rankiniu būdu. Toks testavimo būdas trunka ilgą laiko tarpą. Pakartotinas sąsajos ir scenarijų testavimas praktikoje praktiškai neįmanomas. Programavimo technologijos mobiliems įrenginiams nėra orientuotos į konkrečią aparatūrinę architektūrą ir gali būti taikomos įvairiems mobiliems įrenginiams, palaikantiems mobiliąją technologiją. Mobiliųjų technologijų standartai neapibrėžia kaip aparatūros gamintojai turėtų realizuoti technologijos palaikymą. Todėl grafinė vartotojo sąsaja skirtingų gamintojų įrenginiuose skiriasi viena nuo kitos. Akivaizdu kad, grafinė vartotojo sąsaja bei operacijų scenarijai turi būti išbandyti ne tik emuliatoriuje, bet ir tikrame mobiliame įrenginyje ar jų grupėje.

Mobiliųjų platformų mobiliems įrenginiams realizacijų skirtumai įtakoja operacijų scenarijus. Scenarijuje gali atsirasti papildomi langai. Papildomų langų atsiradimas susijęs su konkretaus mobilaus įrenginio panaudojimu. Todėl operacijų scenarijai turi būti išbandyti visuose įrenginiuose, į kuriuos yra orientuota programa.

Ribotas mobiliųjų įrenginių atminties kiekis, verčia kaupti testavimo atvejų duomenys išorėje. Tai gali būti stalinis kompiuteris, turintis pakankamai resursų testavimo atvejų saugojimui.

Kodo perkodavimas (*obfuscation*). Siekiant sumažinti išeities kodo apimti, atliekamas kodo perkodavimas, kai visu klasių, metodų bei kintamųjų vardai yra pervadinami į sutrumpintus. Toks kodas užima ženkliai mažiau atminties, tačiau yra neskaitomas žmogui. Perkodavimo metu nesudaroma vardų atitikimo lentelė, kurioje būtų nurodyti atitikimai tarp senų bei naujų vardų. Taip pat toks kodas neleidžia pasinaudoti atspindžio technologija (*reflection*).

Šiuo metu programavimo platformos mobiliajai įrangai (J2ME, Mobile.NET) dėl įrenginių apribojimų nepalaiko atspindžio technologijos (*reflection*) [11], kurios pagalba galima kurti objektus žinant jų klasės pavadinimą. Todėl, norint pasiekti programos langus bei jų turinį, reikalinga taikomosios programos sąsaja (API), kuri turės atspindžio technologijos funkcionalumą ir leis apeiti kodo perkodavimo problemas.

Automatizuotam testu atlikimui turi būti naudojama speciali aplinka (*framework*) [43]. Testavimo aplinka atsakinga už testuojamos programos metodų kvietimą. Pagal pateiktą navigacijos grafą bei scenarijus testavimo aplinka atlikinės perėjimus tarp langų, perduodamą testinius duomenys programai bei fiksuodama rezultatus.

## 3.3. TIKSLAI

Mobilios programinės įrangos grafinės vartotojo sąsajos automatizuotam testavimui nustatysime šiuos uždavinius [53]:

- Išbandyti atskiro programos lango funkcionalumą
    - Nustatyti leidžiamų įvesties simbolių aibę
    - Navigacijos galimybės: išėjimas/grįžimas atgal, perėjimas į sekantį langą
- Programos langų grafo išbandymas
    - Automatizuotas scenarijų vykdymas
    - Patikrinti programos langų pasiekiamumą
    - Realaus programos navigacijos grafo atitikimas suprojektuotam
    - Nustatyti projektavimo metu nenumatytus programos langus
- Testavimo duomenų generavimas iš sąsajai suprojektuotų reikalavimų.
- Navigacijos dokumentacijos generavimas pagal realų navigacijos grafą išgautą iš realaus įrenginio
- Išmatuoti vėlinimą – laiko trukmė nuo vartotojo veiksmo pradžios iki atvaizduoto rezultato
- Susiejimas su kitais testavimo būdais: vienetų testavimas, sistemos testavimas, regresinis testavimas

## 3.4. SPRENDIMO BŪDAS

Išorinėje duomenų bazėje yra saugomi programos navigacijos grafas, programos langų aprašymas, testavimo atvejai bei testavimo rezultatų istorija. Išorinės duomenų bazės panaudojimas leidžia išspręsti mobilaus įrenginio riboto atminties kiekio problemą. Iš duomenų saugyklos į testavimo aplinka perduodamas navigacijos grafas bei scenarijai. Kiekvienam navigacijos mazgui perduodami testavimo atvejų duomenys.

Atspindžio problema sprendžiama taikant programinius interfeisus (API). Šių interfeisų pagalba testavimo aplinkoje bus pasiekiami visi programos lango duomenų įvedimo/išvedimo laukai, jų reikšmės bei komandiniai mygtukai. Per programinę sąsaja gauti lango parametrai lyginami su suprojektuotais. Tokiu būdu bus nustatoma ar langas atitinka grafinės vartotojo sąsajos reikalavimus [53]. Taip pat API bus taikomas ir metodams skirtiems atlikti logines operacijas.

Automatizuoto testavimo aplinka – tai savarankiška programa, veikianti mobiliajame įrenginyje. Testavimo aplinka naudoja išorinę duomenų saugyklą testavimo duomenims gauti. Pagal paduotus duomenis: navigacijos grafą, scenarijus ir testinius duomenis, API pagalba kviečiami testuojamosios programinės įrangos metodai. Testavimo aplinka valdys duomenų perdavimą bei fiksuos rezultatus.

Atliekamas kiekvieno vaizduojamo lango bandymas. Tikrinami duomenų įvedimo laukai, leistinos įvesties simbolių aibės. Nustatoma ar visi laukai yra vaizduojami ekrane.

Pagal scenarijų bandoma atlikti perėjimą iš vieno lango į kitą, testavimo aplinka fiksuoja perduotus į metodą duomenys bei gautus rezultatus: loginės operacijos reikšmė bei atvaizduotas langas. Taip pat testavimo aplinka fiksuoja vėlinimą: laiko trukmė per kuria programa reaguoja į vartotojo veiksmus. Perėjimas tarp langų turėtų užtrukti ne ilgiau nei dvi sekundės. Gauti rezultatai perduodami į testavimo orakului. Testavimo orakulas atlieka gautų bei laukiamų rezultatų palyginimą.

Grafinės vartotojo sąsajos automatizuoto testavimo aplinkos struktūrinėje schemoje pavaizduoti duomenų srautai tarp modulių (3.1 Pav.):

1. Testavimo duomenys: navigacijos grafas bei testavimo atvejai. Kiekvienam navigacijos mazgui yra sudaryti testavimo atvejai. Remiantis suprojektuotu navigacijos grafu testavimo aplinka valdys programinę įrangą.
2. Vykdymas. Per API į įvedimo laukus paduodami testavimo duomenys.
3. Rezultatas. Fiksuojama programinės įrangos reakcija į įvestus duomenys. Nustatomas loginės operacijos rezultatas bei sekantis atvaizduotas langas.
4. Testavimo rezultatai grąžinami į testavimo serverį. Testavimo rezultatai nusakys lango duomenų laukus, leidžiamas įvedamų simbolių aibes, galimus perėjimus iš programos lango.
5. Laukiami rezultatai – pagal programos projektą parengti duomenys, nusakantys kiekvieno lango duomenų laukus, leidžiamų įvedimo simbolių aibes, galimus parėjimus iš programos lango.
6. Testavimo rezultatai palyginami su laukiamais rezultatai. Nustatomas realus programos navigacijos medis. Testavimo rezultatai saugojami saugykloje.

Testavimo aplinkos galimybė fiksuoti programos reakciją į paduodamus duomenys bus pritaikyta testavimo scenarijų surašymui, atliekant testavimą rankiniu būdu. Vartotojas dirbs su programa, įvedinės duomenys, o testavimo aplinka fiksuos programos reakciją į vartotojo veiksmus.

## 3.5. LANGŲ GRAFO IR OPERACIJŲ SCENARIJŲ PATEIKIMAS

Pagrindinės testavimo scenarijų ir grafo sudarymo taisyklės [46-50]:

- Testavimo scenarijams bei grafo mazgams aprašyti naudojama XML kalba.
- Grafas aprašomas atskirame faile. Grafo failo pavadinimas sudaromas iš grafo pavadinimo ir plėtinio „xml“, pvz.: `graph1.xml`
- Grafo XML dokumentas apdorojamas DOM tipo XML analizatoriumi. Vienu sykiu visas grafas nuskaitomas į atmintį sudarant medžio tipo struktūrą. Toliau ši struktūra naudojama operacijų scenarijų bei testinių duomenų kūrimui.
- Kiekvieno scenarijaus aprašymui naudojamas atskiras failas. Scenarijaus failo pavadinimas sudaromas iš scenarijaus pavadinimo ir plėtinio „xml“, pvz.: `scenario1.xml`

Scenarijau XML dokumentas apdorojimas SAX tipo XML analizatoriumi. Scenarijuje patiektos operacijos vykdomos tik vieną kartą. Todėl scenarijaus apdorojimui puikiai tiks SAX analizatoriaus funkcionalumas.

## 3.6. LANGŲ GRAFO IR OPERACIJŲ SCENARIJŲ FAILŲ FORMATAI

Šiame skyriuje pateikiamas navigacijos grafui ir operacijų scenarijams aprašymui XML klaba naudojami elementai [12, 46-50].

**a. Grafas**

Grafo aprašymas žymimas etiketėmis `<graph>` ir `</graph>`. Grafui suteikiamas pavadinimas, kuris žymimas etiketėmis `<graphname>` ir `</graphname>`. Grafo langų sąrašas žymimas etiketėmis `<nodelist>` ir `</nodelist>`.

Grafo aprašymo pavyzdys:

```
<graph>
      <graphname>...</graphname>
      <nodelist>
            <node>...</node>
            ...
      </nodelist>
</graph>
```

**b. Langas**

Grafas susideda iš mazgų aprašymų. Mazgo aprašymas žymimas etiketėmis <node> ir </node>. Programos grafinės sąsajos atveju mazgas atitinka langą.

Kiekvieną mazgą vienareikšmiškai nustato unikalus kodas – mazgo ID, kuris žymimas etiketėmis <nodeid> ir </nodeid>. Šakninio mazgo ID = 0. Kitų langų ID didinamas vienetų. Papildomai mazgui suteikiamas pavadinimas, žymimas etiketėmis <nodename> ir </nodename>. Mazgas gali turėti kelias būsenas. Mazgo būsenų sąrašas žymimas etiketėmis <statelist> ir </statelist>. Mazge atvaizduojamų elementų sąrašas žymimas etiketėmis <itemlist> ir </itemlist>.

Mazgo komandų sąrašas žymimas etikėtėmis <cmdlist> ir </cmdlist>.

Mazgo aprašymo pavyzdys:

```
<node>
      <nodeid>...</nodeid>
      <nodename>...</nodename>
      <statelist>
            <state>...</state>
            ...
      </statelist>
      <itemlist>
            <item>...</item>
            ...
      </itemlist>
      <cmdlist>
            <cmd>...</cmd>
            ...
      </cmdlist>
</node>
```

**c. Būsena**

Kiekvienas langas turi savo būsenų aibę. Būsenos aprašymas žymimas etiketėmis <state> ir </state>. Kievieną būseną vienareikšmiškai nustato unikalus kodas – būsenos ID, kuria žymimas etiketėmis <stateid> ir </stateid>. Papildomai būsenai suteikiami pavadinimas, žymimas etiketėmis <statename> ir </statename>, ir aprašas, žymimas etiketėmis <description> ir </description>.

Būsenos aprašymo pavyzdys:

```
<state>
      <stateid>...</stateid>
      <statename>...</statename>
```

```
        <description>...</description>
    </state>
```

**d. Elementas**

Lange atvaizduojami grafiniai elementai. Elemento aprašymas žymimas etiketėmis `<item>` ir `</item>`. Kiekvieną elementą vienareikšmiškai nustato unikalus kodas – elemento ID, kuris žymimas etiketėmis `<itemid>` ir `</itemid>`. Papildomai elementui yra suteikiamas pavadinimas, žymimas etiketėmis `<itemname>` ir `</itemname>`. Elemento tipas žymimas etiketėmis `<type>` ir `</type>`. Elementui gali būti suteikta numatyta reikšmė. Numatytos reikšmės aprašymas žymimas etiketėmis `<defaultvalue>` ir `</ defaultvalue >`. Elementui gali būti taikomi apribojimai. Apribojimų sąrašo aprašymas žymimas etiketėmis `<constraintlist>` ir `</constraintlist>`.

Elemento aprašymo pavyzdys:

```
<item>
        <itemid>...</itemid>
        <itemname>...</itemname>
        <type>...</type>
        <defaultvalue>...</ defaultvalue >
        <constraintlist>
                <constraint>...</constraint>
                ...
        </constraintlist>
</item>
```

**e. Apribojimas**

Grafiniam elementui gali būti taikomi įvesties apribojimai. Apribojimo aprašymas žymimas etiketėmis `<constraint>` ir `</constraint>`. Kiekvieną apribojimą vienareikšmiškai nustato unikalus kodas – apribojimo ID, kuris žymimas etiketėmis `<constraintid>` ir `</constraintid >`. Apribojimo tipas žymimas etiketėmis `<type>` ir `</type>`. Apribojimo reikšmė žymima etiketėmis `<value>` ir `</value>`.

Apribojimo aprašymo pavyzyds:

```
<constraint>
        <constraintid>...</ constraintid >
        <type>...</type>
        <value>...</value>
</constraint>
```

**f. Komanda**

Lango komandų pagalba kviečiamos programos funkcijos. Komandos aprašymas žymimas etiketėmis <cmd> ir </cmd>. Kiekvieną komandą vienareikšmiškai nustato unikalus kodas – komandos ID, kuris žymimas etiketėmis <cmdid> ir </cmdid>. Papildomai komandai suteikiamas pavadinimas, žymimas etiketėmis <cmdname> ir </cmdname>. Prieš vykdant komandą, jai perduodami parametrai. Komandos parametrų sąrašas žymimas etiketėmis <paramlist> ir </paramlist>.

Komandos aprašymo pavyzdys:

```
<cmd>
      <cmdid>...</cmdid>
      <cmdname>...</cmdname>
      <paramlist>
            <param>...</param>
            ...
      </paramlist>
</cmd>
```

**g. Parametras**

Komandai gali būti perduodami parametrai. Komandos parametrų skaičius yra baigtinis. Parametro aprašymas žymimas etiketėmis <param> ir </param>. Kiekvieną parametrą vienareikšmiškai nustato unikalus kodas – parametro ID, kuris žymimas etiketėmis <paramid> ir </paramid>. Papildomai parametrui suteikiamas pavadinimas, žymimas etiketėmis <paramname> ir </paramname>. Parametro tipas žymimas etiketėmis <type> ir </type>.

Parametro aprašymo pavyzdys:

```
<param>
      <paramid>...</paramid>
      <paramname>...</paramname>
      <type>...</type>
</param>
```

**h. Scenarijus**

Scenarijaus aprašymas žymimas etiketėmis <scenario> ir </scenario>.

Scenarijų vienareikšmiškai nustato unikalus kodas – scenarijaus ID, kuris žymimas etiketėmis <scnrid> ir </scnrid>. Papildomai scenarijui suteikiamas pavadinimas, žymimas etiketėmis <scnrname> ir </scnrname>. Scenarijus susideda iš žingsnių. Žingsnių sąrašo aprašymas žymimas etiketėmis <steplist> ir </steplist>.

Scenarijau aprašymo pavyzdys:

```
<scenario>
<scnrid>...</scnrid>
<scnrname>...</scnrname>
<steplist>
      <step>...</step>
      ...
</steplist>
</scenario>
```

**i. Žingsnis**

Scenarijaus žingsnis atitinka operaciją, kuria atliekama pereinant iš vieno lango į kitą. Kiekvienas žingsnis žymimas etiketėmis `<step>` ir `</step>`. Kiekvienas scenarijaus žingsnis yra indeksuojamas. Indeksavimui žymėti naudojamos etiketės `<index>` ir `</index>`. Žingsniui nurodomi mazgo/lango ID, žymimas etiketėmis `<nodeid>` ir `<nodeid>`, mazgo komandos ID, žymimas etiketėmis `<cmdid>` ir `</cmdid>`, komandai perduodamų parametrų sąrašas, žymimas etiketėmis `<paramlist>` ir `</paramlist>`. Kiekvienas parametrui žymėti naudojamos etiketės `<param>` ir `</param>`. Parametrui nurodomas ID, žymimas etiketėmis `<paramid>` ir `</paramid>`, ir riekšmė, žymima etiketėmis `<value>` ir `</value>`.

Žingsnio aprašymo pavyzdys:

```
<step>
      <index>...</index>
      <nodeid>...</nodeid>
      <cmdid>...</cmdid>
      <paramlist>
            <param>
                  <paramid>...</paramid>
                  <value>...</value>
            </param>
            ...
      </paramlist>
</step>
```

Žemiau pateiktos visų navigacijos grafo ir operacijų scenarijuose naudojamos etiketės (Lentelė 1.1).

**Lentelė 3.1 Navigacijos grafo ir operacijų scenarijų aprašymo XML etiketės**

| # | Grupė | Pavadinimas | Pradžios etiketė | Pabaigos etiketė |
|---|---|---|---|---|
| 1. | Grafas | Grafas | <graph> | </graph> |
| 2. | | Pavadinimas | <graphname> | </graphname> |
| 3. | | Langų sąrašas | <nodelist> | </nodelist> |
| 4. | Langas | Langas | <node> | </node> |
| 5. | | ID | <nodeid> | </nodeid> |
| 6. | | Pavadinimas | <nodename> | </nodename> |
| 7. | | Būsenų sąrašas | <statelist> | </statelist> |
| 8. | | Elementų sąrašas | <itemlist> | </itemlist> |
| 9. | | Komandų sąrašas | <cmdlist> | </cmdlist> |
| 10. | Būsena | Būsena | <state> | </state> |
| 11. | | ID | <stateid> | </stateid> |
| 12. | | Pavadinimas | <statename> | </statename> |
| 13. | | Aprašymas | <description> | </description> |
| 14. | Elementas | Elementas | <item> | </item> |
| 15. | | ID | <itemid> | </itemid> |
| 16. | | Pavadinimas | <itemname> | </itemname> |
| 17. | | Tipas | <type> | </type> |
| 18. | | Numatyta reikšmė | <defaultvalue> | </defaultvalue> |
| 19. | | Apribojimų sąrašas | <constraintlist> | </constraintlist> |
| 20. | Apribojimas | Apribojimas | <constraint> | </constraint> |
| 21. | | ID | <constraintid> | </constraintid> |
| 22. | | Tipas | <type> | </type> |
| 23. | | Reikšmė | <value> | </value> |
| 24. | Komanda | Komanda | <cmd> | </cmd> |
| 25. | | ID | <cmdid> | </cmdid> |
| 26. | | Pavadinimas | <cmdname> | </cmdname> |
| 27. | | Parametrų sąrašas | <paramlist> | </paramlist> |
| 28. | Parametras | Parametras | <param> | </param> |
| 29. | | ID | <paramid> | </paramid> |
| 30. | | Pavadinimas | <paramname> | </paramname> |
| 31. | | Tipas | <type> | </type> |
| 32. | Scenarijus | Scenarijus | <scenario> | </scenario> |
| 33. | | ID | <scnrid> | </scnrid> |
| 34. | | Pavadinimas | <scnrname> | </scnrname> |
| 35. | | Žingsnių sąrašas | <steplist> | </steplist> |
| 36. | Žingsnis | Žingsnis | <step> | </step> |
| 37. | | Indeksas | <index> | </index> |
| 38. | | Lango ID | <nodeid> | </nodeid> |
| 39. | | Komandos ID | <cmdid> | </cmdid> |

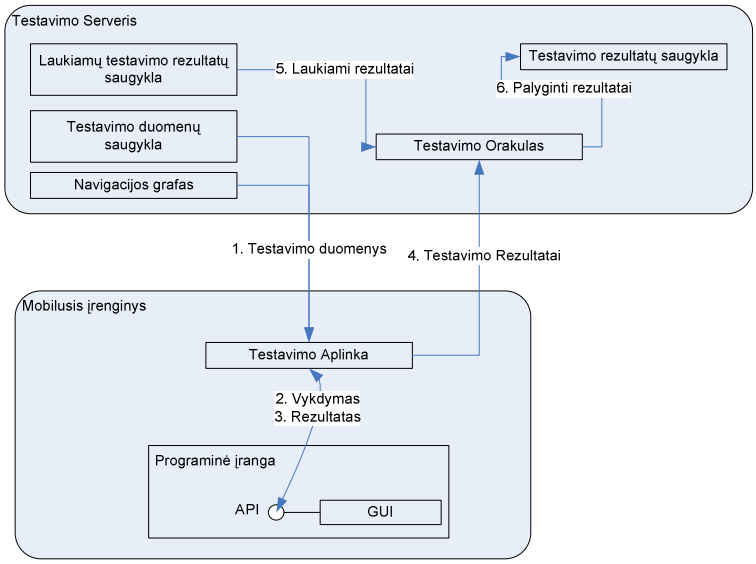
| # | Grupė | Pavadinimas | Pradžios etiketė | Pabaigos etiketė |
|---|---|---|---|---|
| 40. | | Parametrų sąrašas | <paramlist> | </paramlist> |
| 41. | | Parametras | <param> | </param> |
| 42. | | Parametro ID | <paramid> | </paramid> |
| 43. | | Parametro reikšmė | <value> | </value> |

## 3.7. TESTAVIMO APLINKOS VEIKIMAS

Testavimo aplinka (*framework*) yra paskirstyta tarp mobilaus įrenginio ir serverio (3.1 Pav.). Serverio duomenų saugykloje kaupiami visi duomenys testavimo duomenys: navigacijos grafas, testavimo scenarijai. Duomenų atskyrimas nuo mobilaus įrenginio atliekamas dėl kelių priežasčių:

- Mobilaus įrenginio atmintis yra ribota
- Keli mobilūs įrenginiai testuojami pagal bendrus scenarijus
- Kelių įrenginių testavimo rezultatų lyginimas

Mobiliajame įrenginyje veikianti testavimo aplinkos dalis priima testavimo duomenis iš serverio. Testavimo duomenys siunčiami dalimis. Pagal priimtus duomenys vykdomas testavimas.

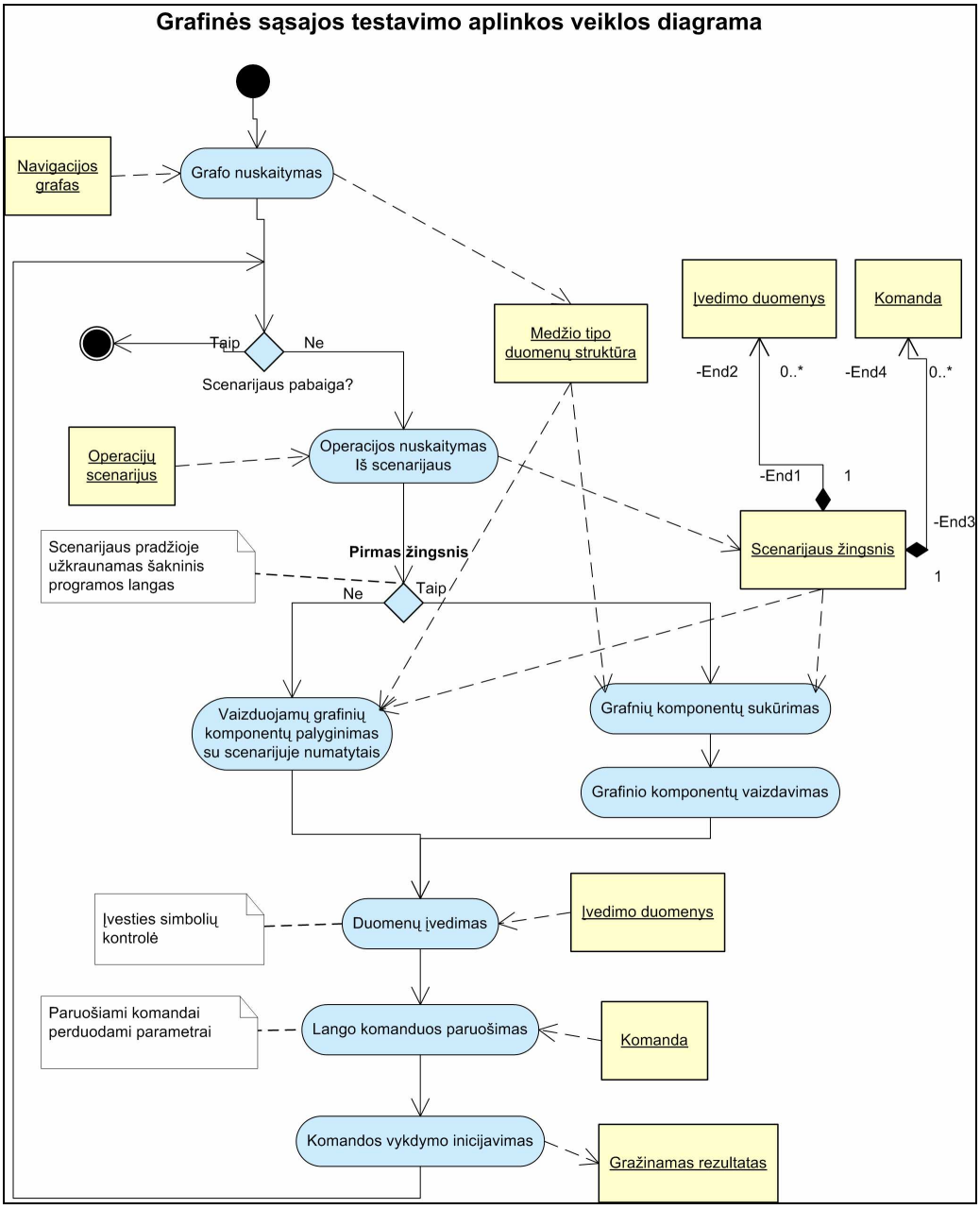



**3.1 Pav. Testavimo aplinkos struktūrinė schema**

Testavimo aplinkos veikimas (3.2 Pav.):

1. Serverio pasiruošimas testavimui. Nuskaitomas navigacijos grafas. Paruošiamas testavimo scenarijus.

2. Scenarijaus persiuntimas į mobilųjį įrenginį.
3. Mobiliajame įrenginyje analizuojamas scenarijus. Iš scenarijaus išskiriamos atskiros operacijos.
4. Mobilusis įrenginys pagal scenarijaus operacijoje naudojamą `<nodeid>` užlausia mazgo aprašymo iš serverio. Mazgo aprašymas atrenkamas iš navigacijos grafo ir grąžinamas į mobilųjį įrenginį.
5. Scenarijaus žingsnio atvaizdavimas. Užkraunami grafiniai komponentai. Tikrinamas jų funkcionalumas. Fiksuojami rezultatai.
6. Paruošiama užkrauto lango komandą. Komandos tikrinimas.
7. Parametrų komandai paruošimas. Komandos parametrų patikrinimui naudojamas grafo lango aprašymas.
8. Inicijuojamas komandos su nuskaitytais parametrais vykdymas.
9. Fiksuojamas komandos rezultatas. Komandos pasėkoje vykdomos loginės operacijos, kurių rezultatas atvaizduojamas grafiškai.
10. Atliekamas patikrinimas, ar langas, į kurį perėjo programa, atitinka scenarijuje numatytam langui.
11. Kartojami žingsniai 3-10 kol pereinamos visos scenarijaus operacijos.
12. Scenarijaus testavimo rezultatai persiunčiami į serverį. Testų orakulas atsakingas už rezultatų apdorojimą.




**3.2 Pav. Grafinės sąsajos testavimo aplinkos veiklos diagrama**

## 3.8. API

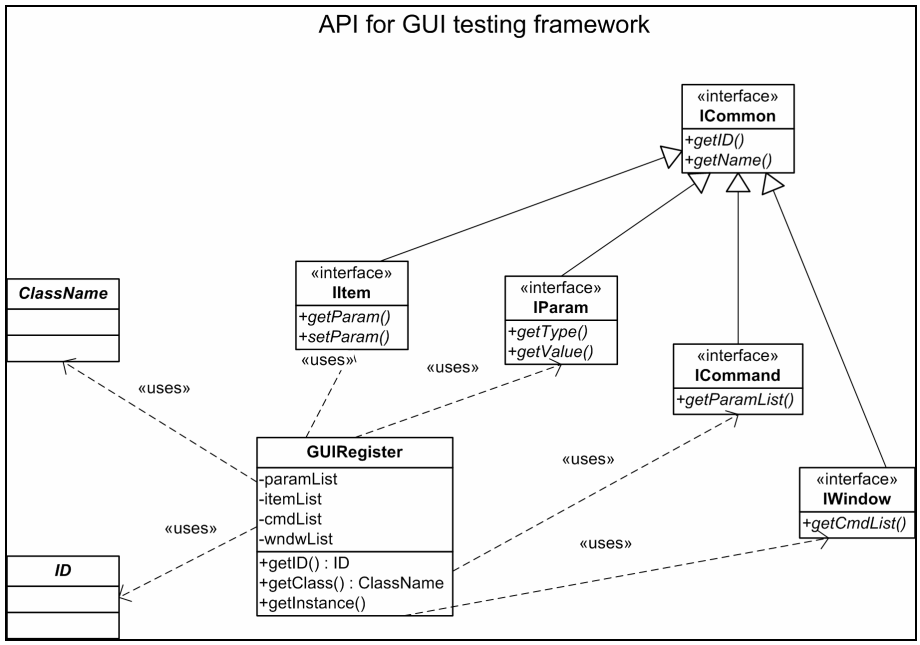
Tam kad atlikti grafinės sąsajos automatizuotą testavimą funkciniame lygyje, reikia kad jos grafiniai komponentai paveldėtų programinę sąsają (*API*) (3.3 Pav.). Per API galime gauti šiuos parametrus:

- Kiekvieno lango identifikacijos kodą (*id*), pavadinimą (*name*), galimų komandų sąrašą (*cmdlist*).

- Kiekvienos komandos identifikacijos kodą (*id*), pavadinimą (*name*), parametrų sąrašą (*paramlist*)
- Kiekvieno parametro identifikacijos kodą (*id*), pavadinimą (*name*), tipą (*type*), reikšmę (*value*).

Pagrindinė problemą yra kodo perkodavimas, kurio negalime išvengti dėl mobiliųjų įrenginių apribojimų. Kodo perkodavimo metu visų klasių pavadinimai pakeičiami nesudarant vardų atitikimo lentelės. Tai sudaro problemą, jei mes norime kurti objektą pagal jo klasės pavadinimą.

Kodo perkodavimo problemai išvengti yra siūloma naudoti objektų identifikacijos kodo susiejimą su perkoduotu klasės pavadinimu. Testavimo aplinkos mobiliajame įrenginyje inicijavimo metu turi būti kietais suprogramuotas grafo objektų susijimas su jų ID. Kai mobiliajame įrenginyje bus vykdomas testavimo aplinkos inicijavimas, visų klasių pavadinimai jau bus perkoduoti. Todėl perkodavimo metu mes sužinosime naujus klasių pavadinimus. Objekto ID ir klasių susijimo lentelių apdorojimui bus naudojamas grafinių komponentų registras.



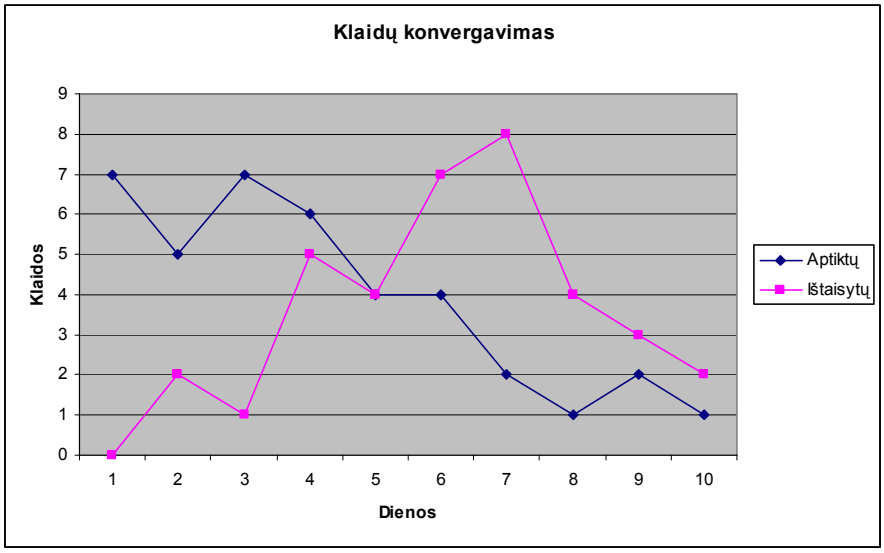

**3.3 Pav. Testavimo aplinkos programavimo sąsaja**

# 4. EKSPERIMENTINĖ DALIS

Šioje dalyje pateikiama siūlomo metodo apžvalga.

Problemos dėl automatizuoto testavimo taikymo kyla dėl to, kad automatizuojama tik viena iš testavimo fazių. Testuotojų darbas tampa sudėtingesniu dėl to, kad reikia prisitaikyti prie automatizuotai testuojamosios fazės: kurti skriptus ir duomenis. Iš siūlomo testavimo grafo, pagal parametrų galimų simbolių aibes, galima generuoti testavimo duomenis.

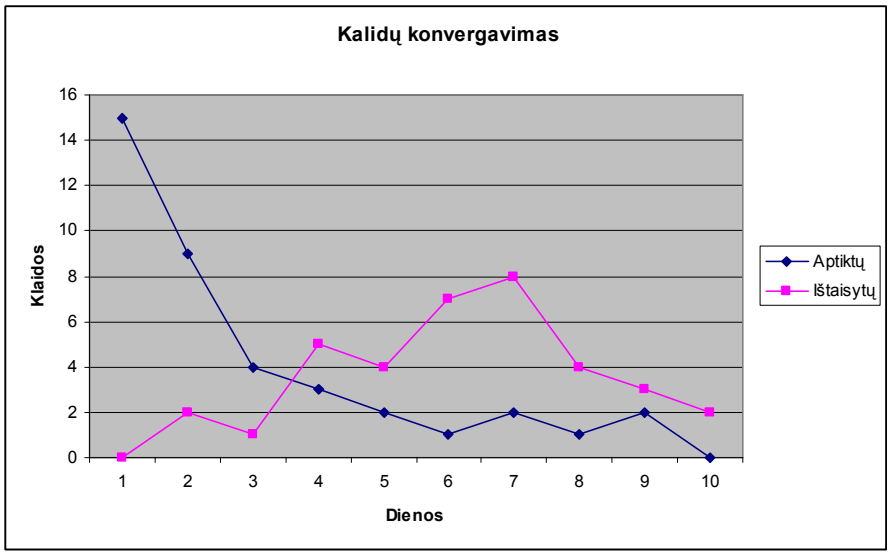
Vykdant testavimą rankiniu būdu, aptiktų ir ištaisytų defektų skaičius kitimas pateiktas žemiau grafe. Neištaisytų klaidų kiekio mažėjimas vadinamas klaidų konvergavimų [38].




**4.1 Pav. Klaidų konvergavimas testuojant rankiniu būdu**

Testavimo pradžioje testuotojai aptinka daugiau klaidų, nei programuotojai sugeba ištaisyti. Kažkuriuo laiko momentu programuotojai taiso daugiau klaidų nei testuotojai jų aptinka. Ištaisytų klaidų kiekis didesnis nei aptiktų klaidų kiekis parodo kad klaidų lygis pradėjo konverguoti ir kūrimo procese prasidėjo stabilizacijos fazė.

Automatizavus testavimą, didesnis klaidų kiekis būtų aptiktas testavimo fazės pradžioje. Vėliau būtų aptinkama mažiau defektų. Todėl stabilizavimo fazė prasidėtų anksčiau.




**4.2 Pav. Klaidų konvergavimas taikant automatizuotą testavimą**

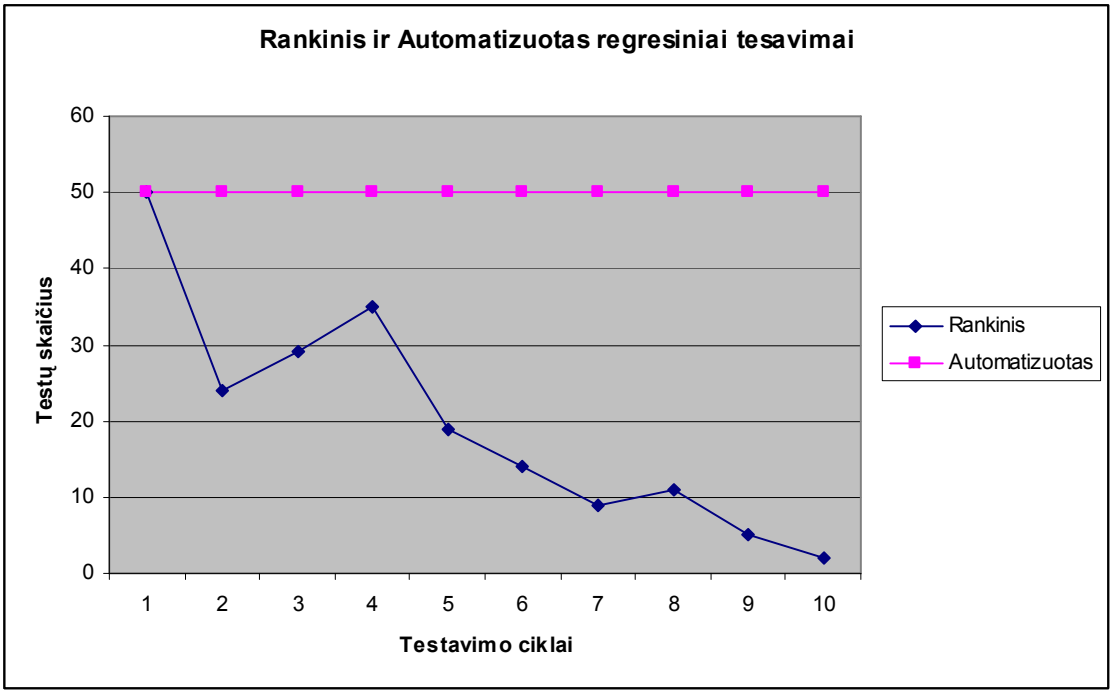
Iš grafikų (4.1 Pav. ir 4.2) matosi, kad testuojant automatizuotu būdu stabilizacinė fazė prasideda anksčiau. Testuojant rankiniu būdu stabilizacinė fazė pasiekiama po 5 dienos, o testuojant automatiniu būdu po 3dienu.

Automatizuotas testavimas taip pat ir užtikrins aukštesnį testavimo efektyvumą. Testavimo efektyvumas – tai aptiktų defektų kiekis iš galimų defektų aibės. Neaptiktų defektų kiekis – tai dažniausiai defektai, apie kuriuos sužinoma iš programos užsakovo. Taip pat lieka dalis defektų kuriuos nepavyko aptikti. Tačiau jų dalis yra santykinai maža ir neturi didelės įtakos nustatant efektyvumą [16].

Testavimo efektyvumui išreikšti naudojama tokia formulė

$$Efektyvumas = \frac{AptiktosKlaidos}{AptiktosKlaidos + NeaptiktosKlaidos}$$

Rankinis testavimas riboja testavimo komandos galimybes. Praktikoje testavimo komandos darbo trukmė ir resursai yra apriboti. Todėl pakartotinas testavimas atliekamas ne visoms testuojamo produkto dalims. Automatizuotas testavimas leidžia praktiškai neapsiriboti testų kiekiu.




**4.3 Pav. Rankinis ir automatizuotas regresiniai testavimai**

Pagal pasiūlyta testavimo metodą yra aprašytas projekto navigacijos grafas bei scenarijus.

Realaus navigacijos grafo schema, navigacijos grafo atvaizdavimas XML formatu ir testavimo scenarijaus atvaizdavimas XML formatu patiekti prieduose.

# 5. IŠVADOS

1. Projektavimo fazėje reikia vertinti kuriamo produkto testavimo būdus.
2. Automatizuotas testavimas leidžia atlikti regresinį testavimą.
3. Automatizuotas regresinis testavimas praktiškai neriboja vykdomų testavimo atvejų kiekio. Tai leidžia padidinti testavimo efektyvumą, nedidinant resursų.
4. Automatizuotos testavimo aplinkos sukūrimo ir pritaikymo kaštai pateisinami pakartotino testavimo teikiamais privalumais.
5. Atskiros testavimo fazės automatizavimas negarantuoja viso testavimo proceso efektyvumo didėjimo.

# 6. TERMINŲ IR SANTRUMPŲ ŽODYNAS

| # | Trumpinys | Apibrėžimas |
|---|---|---|
| 1. | MSDE | Microsoft SQL Server 2000 Desktop Engine |
| 2. | JRE | Java Runtime Environment |
| 3. | jKVM | Java Kilobite Virtual Macine |
| 4. | PDA | Personal Digital Assistant |
| 5. | DB | Database |
| 6. | DBMS | Database Management System |
| 7. | J2ME | Java Micro Editon |
| 8. | JVM | Java Virtual Machine |
| 9. | MIDP | Mobile Information Device Profile |
| 10. | CLDC | Connected Limited Device Configuration |
| 11. | XML | eXtensible Markup Language |
| 12. | SAX | Simple API for XML |
| 13. | DOM | Document Object Model |
| 14. | API | Application programming interface |

# 7. LITERATŪRA


1. Extensible Markup Language [Interaktyvus]. [žiūrėta 2005-04-15]. Prieiga per internetą: http://www.w3.org/XML/.
2. Testingfaqs.org. GUI Test Drivers [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.testingfaqs.org/t-gui.html.
3. Kas yra XML? [Interaktyvus]. [žiūrėta 2005-04-20]. Prieiga per internetą: http://www.xml.lt/.
4. Quality First Software GmbH. qftestJUI - The Java GUI Testtool [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: www.qfs.de/en/qftestJUI.
5. Software Testing Dictionary [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.geocities.com/xtremetesting/TestingDictionary.html.
6. Software AG. XML Glossary [Interaktyvus]. [žiūrėta 2005-05-12]. Prieiga per internetą: http://www.softwareag.com/xml/about/glossary.htm.
7. Enterprise Integration Toolkit. Testing Guide [Interaktyvus]. [žiūrėta Prieiga per internetą: http://www.eitoolkit.com/tools/implementation/build/testing_guide.doc.
8. Sidabrinis tinklas. M–Calculatror [Interaktyvus]. [žiūrėta 2003-11-15]. Prieiga per internetą: http://www.stinklas.lt/mobilus.htm.
9. Sidabrinis tinklas. M–partner [Interaktyvus]. [žiūrėta 2003-11-15]. Prieiga per internetą: http://www.stinklas.lt/mobilus.htm.
10. Adnconsulting. Mobile Order Taker [Interaktyvus]. [žiūrėta 2003.11.27]. Prieiga per internetą: http://www.adnconsulting.com/Ingles/mobileordertaker.htm.
11. Ancona, M., Cazzola, W. Implementing the essence of reflection: a reflective run-time environment// Symposium on Applied Computing archive. Proceedings of the 2004 ACM symposium on Applied computing: tarptautinės konferencijos pranešimų medžiaga, [2004 m.], p. 1503-1507.
12. Baniulis, K., Tamulynas, B. Duomenų Struktūros. Kaunas: Technologija, 2000. pp. 102-136
13. Berber, S. Automated Testing for Embedded Devices [Interaktyvus]. [žiūrėta 2004-10-15]. Prieiga per internetą: http://www.perftestplus.com.
14. Chin. Mobile technologies and interactive applications [Interaktyvus]. [žiūrėta 2005-05-11]. Prieiga per internetą: http://www.chin.gc.ca/English/Digital_Content/Tip_Sheets/Wireless/glossary.html.

15. Compuware. TestPartner Product Preview. Automated, repeatable testing with TestPartner [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.compuware.com/products/qacenter/2925_ENG_HTML.htm.
16. Craig, R., Jaskiel, S. Measuring Test Effectiveness Actech House Publisher, 2002. ISBN: 1580535089.
17. Desk, J.N. Java Is Now "World's Most Ubiquitous Mobile Development Platform" [Interaktyvus]. [žiūrėta 2005-05-13]. Prieiga per internetą: http://java.sys-con.com/read/78499.htm.
18. Digia. Digia Test Automation Services [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.digia.com/digia/devhome.nsf/891315221476DFFDC2256E9E002AF423/$file/Digia_Test_Automation_Services.pdf.
19. Dustin, E. Lessons in Test Automation: A Manager's Guide to Avoiding Pitfalls When Automating Testing [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.informit.com/articles/article.asp?p=21467&rl=1.
20. Dustin, E. Effective software testing: 50 specific ways to improve your testing. Boston: Addison-Wesley, 2003. ISBN: 0-201-79429-2. p. 215
21. Herbert, M. The practical organization of automated software testing [Interaktyvus]. [žiūrėta 2005-01-16]. Prieiga per internetą: http://www.automated-testing.com/PATfinal.htm.
22. Ibm. Rational Robot Product Overview [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www-306.ibm.com/software/awdtools/tester/robot/.
23. Ibm. Rational Visual Test [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www-306.ibm.com/software/awdtools/visualtest/support/index.html.
24. Javamobiles. List of JVM for PDA [Interaktyvus]. [žiūrėta 2004-10-12]. Prieiga per internetą: http://www.javamobiles.com/jvm.html.
25. Kanglin, L., Mengqi, W. Effective GUI test automation: developing an automated GUI testing tool. Boston: Sybex, 2005. ISBN: 0-7821-4351-2.
26. Linz, T., Daidl, M. How to automate testing of graphical user interfaces [Interaktyvus]. [žiūrėta 2005-02-19]. Prieiga per internetą: http://www.imbus.de/forschung/pie24306/gui/aquis-full_paper-1.3.shtml.
27. Memon, A. Home page of GUTAR – a GUI testing framework [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.cs.umd.edu/~atif/guitar.html.

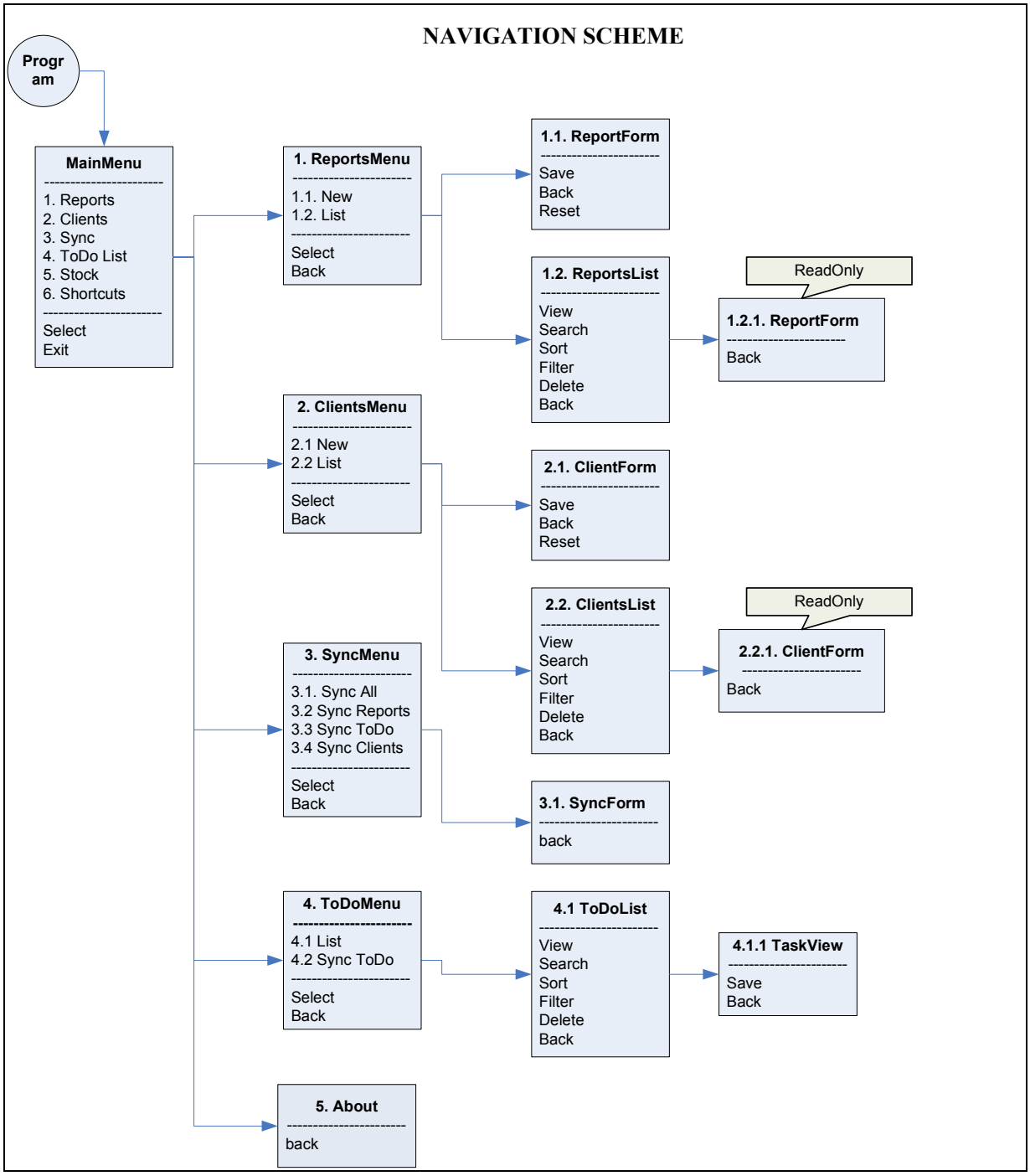
28. Memon, A., Pollack, M., Soffa, M. Hierarchical GUI Test Case Generation Using Automated [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.computer.org/portal/site/transactions/menuitem.a66ec5ba52117764cfe79d108bcd45f3/index.jsp?&pName=tse_home/&.
29. Memon, A., Pollack, M., Soffa, M. Using a Goal-driven Approach to Generate Test Cases for GUIs// The 21st International Conference on Software Engineering: tarptautinės konferencijos pranešimų medžiaga, [Los Angeles, USA, 1999 m. gegužės 16-22], p.
30. Memon, A., Pollack, M., Soffa, M. Automated Test Oracles for GUIs// Eighth International Symposium on the Foundations of Software Engineering (FSE'2000): tarptautinės konferencijos pranešimų medžiaga, [San Diego, CA, 2000 m. lapkričio 6], p.
31. Memon, A., Pollack, M., Soffa, M. Plan Generation for GUI Testing// The Fifth International Conference on Artificial Intelligence Planning and Scheduling: tarptautinės konferencijos pranešimų medžiaga, [Breckenridge, CO, USA, 2000 m. balandžio 15 – 17], p.
32. Memon, A., Pollack, M., Soffa, M. A Planning-Based Approach to GUI Testing// 13th International Software/Internet Quality Week (QW2000): tarptautinės konferencijos pranešimų medžiaga, [San Francisco, CA, USA, 2000 m. gegužės 30 - liepos 2], p.
33. Memon, A., Soffa, M., Pollack, M. Coverage Criteria for GUI Testing// 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE-9): tarptautinės konferencijos pranešimų medžiaga, [2001 m. spalio 14], p.
34. Mercury Mercury. WinRunner Overview [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.mercury.com/uk/products/quality-center/functional-testing/winrunner/.
35. Mercury, Interactive. LoadRunner Overview [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.mercury.com/uk/products/performance-center/loadrunner/.
36. Microsoft. Master-Master Row-Level Synchronization [Interaktyvus]. [žiūrėta 2004.05.15]. Prieiga per internetą: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpatterns/html/DesSynchronization.asp.
37. Microsoft. Microsoft SQL Server 2000 Desktop Engine [Interaktyvus]. [žiūrėta 2005-01-19]. Prieiga per internetą: http://www.microsoft.com/sql/msde/productinfo/overview.asp.

38. Microsoft. MSF process model [Interaktyvus]. [žiūrėta 2004-08-20]. Prieiga per internetą: http://www.msdb.ru/Downloads/Msdn/Msf/MSF_process_model_rus.doc.
39. Msdn. Microsoft Foundation Class Library [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vcmfc98/html/mfchm.asp.
40. Packevičius, Š., Ušaniov, A. Vartotojo sąsajos delniniams kompiuteriams kūrimo principai// Informacinė Visuomenė ir Universitetinės Studijos: konferencijos pranešimų medžiaga: tarptautinės konferencijos pranešimų medžiaga, [Kaunas, 2004 m. balandžio 15], p. 323-327.
41. Packevičius, Š., Ušaniov, A., Bareiša, E. Programinės įrangos testavimas mobiliuose įrenginiuose naudojant nuotolinį modulių testavimą// Informacinės Technologijos: konferencijos pranešimų medžiaga: tarptautinės konferencijos pranešimų medžiaga, [Kaunas, 2005 m. balandžio 29], p. 177-180.
42. Piroumian, V. Wireless J2ME platform programming. Moscow: Sun Microsystems Press, 2002. ISBN: 0-13-044914-8.
43. Satoh, I. A Testing Framework for Mobile Computing Software// IEEE Transactions on Software Engineering. ISSN 2003, p. 1112-1121.
44. Segue. Functional & Regression Testing. SilkTest [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://www.segue.com/products/functional-regressional-testing/index.asp.
45. Sholler, D. dotNet seen gaining steam in dev projects [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2860227,00.html.
46. Slovinski, J. Advanced UI Design Using XML and XSL - Part 1: Folder Tree Creation [Interaktyvus]. [žiūrėta 2005-05-13]. Prieiga per internetą: http://www.15seconds.com/issue/010921.htm.
47. Slovinski, J. Advanced UI Design Using XML and XSL - Part 2: Custom Context Menu Creation [Interaktyvus]. [žiūrėta 2005-05-13]. Prieiga per internetą: http://www.15seconds.com/issue/010927.htm.
48. Slovinski, J. Advanced UI Design Using XML and XSL - Part 3: Folder Tree Administration [Interaktyvus]. [žiūrėta 2005-05-13]. Prieiga per internetą: http://www.15seconds.com/issue/011113.htm.

49. Slovinski, J. Advanced UI Design Using XML and XSL - Part 4: Folder Tree Drag and Drop [Interaktyvus]. [žiūrėta 2005-05-13]. Prieiga per internetą: http://www.15seconds.com/issue/011129.htm.
50. Slovinski, J. Advanced UI Design Using XML and XSL - Part 5: Progress Indicator Creation [Interaktyvus]. [žiūrėta 2005-05-13]. Prieiga per internetą: http://www.15seconds.com/issue/011212.htm.
51. Sommerville, I. Software engineering. Boston: Addison-Wesley, 2004. ISBN: 0321210263. p. 759
52. Umiacs University of Maryland Institute for Advanced Computer Studies. [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://umiacs.umd.edu/.
53. Ušaniov, A., Packevičius, Š. Grafinės vartotojo sąsajos automatizuotas testavimas mobiliems įrenginiams su apribojimais// Informacinės Technologijos: konferencijos pranešimų medžiaga: tarptautinės konferencijos pranešimų medžiaga, [Kaunas, 2005 m. balandžio 29], p. 180-184.
54. Ušaniov, A., Packevičius, Š., Bareiša, E. Duomenų saugyklų taikymas mobiliose sistemose// Informacinės technologijos - 2005: tarptautinės konferencijos pranešimų medžiaga, [Kaunas, m. sausio 29], p. 508-513.
55. Ušaniov, A., Packevičius, Š., Bareiša, E. Tinklo ryšio priemonės mobiliuose įrenginiuose// Informacinės technologijos - 2005: tarptautinės konferencijos pranešimų medžiaga, [Kaunas, 2005 m. vasario 28], p. 767-770.
56. Wall, T. Getting Started with the Abbot Java GUI Test Framework [Interaktyvus]. [žiūrėta 2005-05-14]. Prieiga per internetą: http://abbot.sourceforge.net/.
57. Weiss, S. Handheld usability. London: John Wiley & Sons Ltd, 2002. ISBN: 0-470-84446-9.
58. Zambelich, K. Totally data-driven automated testing [Interaktyvus]. [žiūrėta 2005-02-18]. Prieiga per internetą: http://www.sqa-test.com/articles.html.
59. Ематин, В., Закис, А., Новичков, А., Шкляева, Н., Подоляк, О. Автоматизация процесса тестирования при помощи методологии и инструментальных средств IBM Rational [Interaktyvus]. [žiūrėta 2005-05-10]. Prieiga per internetą: http://content.mail.ru/arch/12788/597372.html.

# 8. PRIEDAI

## A. Navigacijos grafas



## B. Navigacijos grafo aprašymas XML kalba.

```
<?xml version="1.0"?>
<graph>
	<graphname>Mobilaus vadybininko darbo vieta</graphname>
	<nodelist>
		<node>
			<nodeid>0</nodeid>
			<nodename>MainMenu</nodename>
			<itemlist>
				<item>
					<itemid>1</itemid>
					<itemname>MainMenu</itemname>
					<itemtype>List</itemtype>
					<defaultvalue>"Reports", "Clients", "Sync", "ToDo
List", "Stock", "Shortcut"</defaultvalue>
				</item>
			</itemlist>
			<cmdlist>
				<cmd>
					<cmdid>1</cmdid>
					<cmdname>Select</cmdname>
					<paramlist>
						<param>
							<paramid>1</paramid>
							<paramname>MenuItem</paramname>
							<type>String</type>
						</param>
					</paramlist>
				</cmd>
				<cmd>
					<cmdid>2</cmdid>
					<cmdname>Exit</cmdname>
				</cmd>
			</cmdlist>
		</node>
		<node>
			<nodeid>1</nodeid>
			<nodename>ReportsMenu</nodename>
			<itemlist>
				<item>
					<itemid>1</itemid>
					<itemname>ReportsMenu</itemname>
					<itemtype>List</itemtype>
					<defaultvalue>"New", "List"</defaultvalue>
				</item>
			</itemlist>
			<cmdlist>
				<cmd>
					<cmdid>1</cmdid>
```

```
                        <cmdname>Select</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>MenuItem</paramname>
                                        <type>String</type>
                                </param>
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>2</cmdid>
                        <cmdname>Back</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>2</paramid>
                                        <paramname>ParentWndw</paramname>
                                        <type>Displayable</type>
                                </param>
                        </paramlist>
                </cmd>
        </cmdlist>
</node>
<node>
        <nodeid>2</nodeid>
        <nodename>ClientsMenu</nodename>
        <itemlist>
                <item>
                        <itemid>1</itemid>
                        <itemname>ClientsMenu</itemname>
                        <itemtype>List</itemtype>
                        <defaultvalue>"New", "List"</defaultvalue>
                </item>
        </itemlist>
        <cmdlist>
                <cmd>
                        <cmdid>1</cmdid>
                        <cmdname>Select</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>MenuItem</paramname>
                                        <type>String</type>
                                </param>
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>2</cmdid>
                        <cmdname>Back</cmdname>
                        <paramlist>
                                <param>
```

```
                                                <paramid>2</paramid>
                                                <paramname>ParentWndw</paramname>
                                                <type>Displayable</type>
                                        </param>
                                </paramlist>
                        </cmd>
                </cmdlist>
        </node>
        <node>
                <nodeid>3</nodeid>
                <nodename>SyncMenu</nodename>
                <itemlist>
                        <item>
                                <itemid>1</itemid>
                                <itemname>SyncMenu</itemname>
                                <itemtype>List</itemtype>
                                <defaultvalue>"Sync All", "Sync Reports", "Sync
ToDo", "Sync clients"</defaultvalue>
                        </item>
                </itemlist>
                <cmdlist>
                        <cmd>
                                <cmdid>1</cmdid>
                                <cmdname>Select</cmdname>
                                <paramlist>
                                        <param>
                                                <paramid>1</paramid>
                                                <paramname>MenuItem</paramname>
                                                <type>String</type>
                                        </param>
                                </paramlist>
                        </cmd>
                        <cmd>
                                <cmdid>2</cmdid>
                                <cmdname>Back</cmdname>
                                <paramlist>
                                        <param>
                                                <paramid>2</paramid>
                                                <paramname>ParentWndw</paramname>
                                                <type>Displayable</type>
                                        </param>
                                </paramlist>
                        </cmd>
                </cmdlist>
        </node>
        <node>
                <nodeid>4</nodeid>
                <nodename>ToDoMenu</nodename>
                <itemlist>
                        <item>
```

```
                        <itemid>1</itemid>
                        <itemname>ToDoMenu</itemname>
                        <itemtype>List</itemtype>
                        <defaultvalue>"List", "Sync ToDo"</defaultvalue>
                </item>
        </itemlist>
        <cmdlist>
                <cmd>
                        <cmdid>1</cmdid>
                        <cmdname>Select</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>MenuItem</paramname>
                                        <type>String</type>
                                </param>
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>2</cmdid>
                        <cmdname>Back</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>2</paramid>
                                        <paramname>ParentWndw</paramname>
                                        <type>Displayable</type>
                                </param>
                        </paramlist>
                </cmd>
        </cmdlist>
</node>
<node>
        <nodeid>5</nodeid>
        <nodename>About</nodename>
        <itemlist>
                <item>
                        <itemid>1</itemid>
                        <itemname>Text</itemname>
                        <itemtype>String</itemtype>
                        <constraintlist>
                                <constraint>
                                        <constraintid>1</constraintid>
                                        <type>ReadOnly</type>
                                </constraint>
                        </constraintlist>
                </item>
        </itemlist>
        <cmdlist>
                <cmd>
                        <cmdid>1</cmdid>
```

```
                        <cmdname>Back</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>ParentWndw</paramname>
                                        <type>Displayable</type>
                                </param>
                        </paramlist>
                </cmd>
        </cmdlist>
</node>
<node>
        <nodeid>11</nodeid>
        <nodename>ReportForm</nodename>
        <itemlist>
                <item>
                        <itemid></itemid>
                        <itemname></itemname>
                        <itemtype></itemtype>
                        <defaultvalue></defaultvalue>
                        <constraintlist>
                                <constraint>
                                        <constraintid></constraintid>
                                        <type></type>
                                        <value></value>
                                </constraint>
                        </constraintlist>
                </item>
        </itemlist>
        <cmdlist>
                <cmd>
                        <cmdid>1</cmdid>
                        <cmdname>Save</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>NewData</paramname>
                                        <type>Record</type>
                                </param>
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>2</cmdid>
                        <cmdname>Back</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>ParentWndw</paramname>
                                        <type>Displayable</type>
                                </param>
```

```
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>3</cmdid>
                        <cmdname>Reset</cmdname>
                </cmd>
        </cmdlist>
</node>
<node>
        <nodeid>12</nodeid>
        <nodename>ReportsList</nodename>
        <itemlist>
                <item>
                        <itemid>1</itemid>
                        <itemname>ReportsList</itemname>
                        <itemtype>List</itemtype>
                </item>
        </itemlist>
        <cmdlist>
                <cmd>
                        <cmdid>1</cmdid>
                        <cmdname>View</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>MenuItem</paramname>
                                        <type>String</type>
                                </param>
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>2</cmdid>
                        <cmdname>Search</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>SearchItem</paramname>
                                        <type>String</type>
                                </param>
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>3</cmdid>
                        <cmdname>Sort</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>SortType</paramname>
                                        <type>String</type>
                                </param>
```

```
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>4</cmdid>
                        <cmdname>Filter</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>Filter</paramname>
                                        <type>String</type>
                                </param>
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>5</cmdid>
                        <cmdname>Delete</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>MenuItem</paramname>
                                        <type>String</type>
                                </param>
                        </paramlist>
                </cmd>
                <cmd>
                        <cmdid>6</cmdid>
                        <cmdname>Back</cmdname>
                        <paramlist>
                                <param>
                                        <paramid>1</paramid>
                                        <paramname>ParentWndw</paramname>
                                        <type>Displayable</type>
                                </param>
                        </paramlist>
                </cmd>
        </cmdlist>
</node>
<node>
        <nodeid>121</nodeid>
        <nodename>ReportForm</nodename>
        <itemlist>
                <item>
                        <itemid>1</itemid>
                        <itemname>data</itemname>
                        <itemtype>Record</itemtype>
                </item>
        </itemlist>
        <cmdlist>
                <cmd>
                        <cmdid>1</cmdid>
```

```
                                        <cmdname>Back</cmdname>
                                        <paramlist>
                                                <param>
                                                        <paramid>1</paramid>
                                                        <paramname>Parent</paramname>
                                                        <type>Displayable</type>
                                                </param>
                                        </paramlist>
                                </cmd>
                        </cmdlist>
                </node>
        </nodelist>
</graph>
```

## C. Testavimo scenarijaus aprašymas XML kalba.

```
<?xml version="1.0"?>
<scenario>
	<scnrid>01</scnrid>
	<scnrname>ViewReport</scnrname>
	<steplist>
		<step>
			<index>1</index>
			<nodeid>0</nodeid>
			<cmdid>1</cmdid>
			<paramlist>
				<param>
					<paramid>1</paramid>
					<value>Reports</value>
				</param>
			</paramlist>
		</step>
		<step>
			<index>2</index>
			<nodeid>1</nodeid>
			<cmdid>1</cmdid>
			<paramlist>
				<param>
					<paramid>1</paramid>
					<value>List</value>
				</param>
			</paramlist>
		</step>
		<step>
			<index>3</index>
			<nodeid>12</nodeid>
			<cmdid>1</cmdid>
			<paramlist>
				<param>
					<paramid>1</paramid>
					<value>View</value>
				</param>
			</paramlist>
		</step>
		<step>
			<index>4</index>
			<nodeid>121</nodeid>
			<cmdid>1</cmdid>
			<paramlist>
				<param>
					<paramid>1</paramid>
					<value>ReportList</value>
```

```
                        </param>
                    </paramlist>
                </step>
                <step>
                    <index>5</index>
                    <nodeid>12</nodeid>
                    <cmdid>2</cmdid>
                    <paramlist>
                        <param>
                            <paramid>1</paramid>
                            <value>ReportsMenu</value>
                        </param>
                    </paramlist>
                </step>
                <step>
                    <index>6</index>
                    <nodeid>1</nodeid>
                    <cmdid>2</cmdid>
                    <paramlist>
                        <param>
                            <paramid>1</paramid>
                            <value>MainMenu</value>
                        </param>
                    </paramlist>
                </step>
        </steplist>
</scenario>
```

## D. Konferencijų medžiaga

# DUOMENŲ SAUGYKLŲ TAIKYMAS MOBILIOSE SISTEMOSE

**Andrej Ušaniov, Packevičius Šarūnas, Eduardas Bareiša**

*Kauno Technologijos universitetas, Informatikos fakultetas, Programų Inžinerijos Katedra, Studentų g. 50, LT - 3031 Kaunas*

Mobiliųjų įrenginių platus taikymas bei vartojimas informaciniuose sistemose buvo beprasmis dėl pačių įrenginių ribotų galimybių bei mobiliųjų duomenų saugyklų nebuvimo. Informacinės sistemos su mobiliais įrenginiais negalėdavo suteikti savo vartotojams pridėtinės vertės atitinkančios informacinės sistemos kainos. Tačiau situacija pasikeitė, atsiradus priemonėm, leidžiančios organizuoti mobiliąsias duomenų saugyklas. Šiuo metu mobilios informacinės sistemos kūrėjai susiduria su jų poreikius atitinkančios mobilios Duomenų Bazių Valdymo Sistemos pasirinkimo problema.

Pranešime apžvelgiamos duomenų saugyklų mobiliuose įrenginiuose organizavimo būdai bei technologijos: Java Record Management System, Microsoft Pocket PC platformos duomenų bazės, Palm OS platformos duomenų bazės, Išorinių duomenų bazių panaudojimas

## 1. Įžanga

Mobiliųjų įrenginių platus taikymas bei vartojimas informaciniuose sistemose buvo beprasmis dėl pačių įrenginių ribotų galimybių bei mobiliųjų duomenų saugyklų nebuvimo. Informacinės sistemos su mobiliais įrenginiais negalėdavo suteikti savo vartotojams pridėtinės vertės atitinkančios informacinės sistemos kainos. Tačiau situacija pasikeitė, atsiradus priemonėm, leidžiančios organizuoti mobiliąsias duomenų saugyklas. Šiuo metu mobilios informacinės sistemos kūrėjai susiduria su jų poreikius atitinkančios mobilios Duomenų Bazių Valdymo Sistemos pasirinkimo problema.

Kuriant programinė įranga mobiliems įrenginiams, duomenų bazė mobiliame įrenginyje numatoma norint tiekti vartotojui priėjimą prie jam reikiamų duomenų, greitą duomenų pasiimimą net ir dirbant atsijungus nuo tinklo. Tokio sprendimo atveju būna reikalinga r duomenų sinchronizacijos su pagrindine duomenų baze mechanizmai. Projektuojant tokia programinę įrangą tenka pasirinkti duomenų bazės valdymo sistemą mobiliam įrenginiui. Mobilaus įrenginio techniniai parametrai ir jo programinė įranga apriboja projektuotojo galimybes pasirinkti tinkamą duomenų bazės valdymo sistemą mobiliam įrenginiui.

## 2. Metrikos

Norint įvertinti duomenų bazių valdymo sistemų mobiliems įrenginiams galimybes ir jų tinkamumą kuriant programinę įrangą mobiliems įrenginiams buvo vertinami tokie parametrai, jie pateikti sekančioje lentelėje.

**1 lentelė. Duomenų bazių valdymo sistemų mobiliems įrenginiams palyginimo kriterijai.**

| Kriterijus | Aprašymas |
|---|---|
| Platforma | Kokiuose delniniuose įrenginiuose galima naudoti. (PocketPC, PalmOS, Windows CE ir pan.) |
| Atminties sunaudojimas | Kiek reikia atminties mobiliame įrenginyje, kad suktųsi duomenų bazės valdymo sistema. |

| | |
|---|---|
| Duomenų bazės dydis | Kokio dydžio galima sukurti duomenų bazę mobiliame įrenginyje. |
| Duomenų bazės tipas | Reliacinė, failų sistema paremta. |
| Ryšys su kitomis DB | Ar yra galimybė sinchronizuoti duomenis su duomenų bazėmis esančiomis serveriuose. |
| Sinchronizavimo protokolas | Koks protokolas naudojamas sinchronizuoti duomenų bazes tarp mobilių įrenginių ir serverių. SyncML, ActiveSync ir pan. |
| Sinchronizavimo konfliktų sprendimas programiškai | Ar yra galimybės programiškai išspręsti sinchronizavimo metu iškilusias problemas. T.y. ar yra galimybė rašyti kokius plug-ins sinchronizavimo protokolui. |
| Sinchronizavimo konfliktų sprendimo vieta. | Serveryje, mobiliame įrenginyje. Programuojama atskirai. |
| Duomenų bazių kiekis | Kiek duomenų bazių galiam sukurti mobiliame įrenginyje. |
| Lentelių kiekis | Kiek galima sukurti lentelių duomenų bazėje mobiliame įrenginyje. |
| Laukų kiekis | Kiek galima sukurti lentelėje laukų. |
| Stored Procedūrų palaikymas | Ar galima naudoti Stored procedūras duomenų bazėje mobiliame įrenginyje. |
| SQL palaikymas | Kuria SQL versiją palaiko duomenų bazė mobiliame įrenginyje. |
| Palaikomi duomenų tipai. | Kokius duomenų tipus palaiko mobili duomenų bazė. |
| Duomenų bazės valdymo įrankiai | Ar yra duomenų bazės valdymo įrankiai pasiekiami mobiliame įrenginyje. |
| Programavimo priemonės. | Ar yra bibliotekos skirtos ADO.NET, ADO, ODBC, JDBC, native biblioteka. |
| Index palaikymas | Ar palaiko duomenų bazė indeksavimą. |
| Foreighn key palaikymas | Ar palaiko duomenų bazė Foreighn key. |
| Primary Key Palaikymas | Ar palaiko duomenų bazė Primary key. |
| Sinchronizavimas iš kelių serverių. | Ar galiam sinchronizuoti duomenis iš kelių serverių į ta pačia duomenų bazę mobiliame įrenginyje. |
| Sinchronizavimo stebėjimas | Ar yra programinės priemonės stebėti sinchronizavimo progresą. |
| Sinchronizavimas laukų. | Ar galima sinchronizuoti tik kelis laukus lentelėje vietoj visos eilutės. |
| Programų kiekis | Ar gali kelios programos mobiliame įrenginyje pasiekti tą pačią duomenų bazę. |
| Saugumas | Ar yra priemonės apsaugoti duomenų bazę mobiliame įrenginyje (šifravimas, autentifikavimas, autorizavimas) |
| Kaina | Kieka kainuoja licenzija vienam mobiliam įrenginiui. |

3. Duomenų bazės mobiliuose įrenginiuose

Analizuojamos tokios duomenų bazių valdymo sistemos mobiliems įrenginiams:

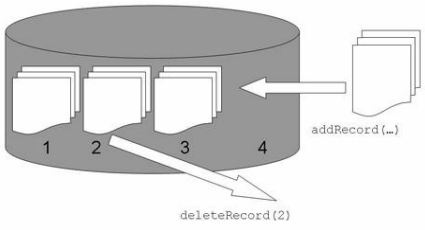
- Java Record Management System
- Microsoft SQL CE Server

- Oracle Database Lite
- DB2 Everyplace

4. Java Record Management System

Kaip ir daugeliui desktop-based programų, MIDletams (Java mobiliosios programos) reikalinga pastovi duomenų saugojimo priemonė. Skirsime duomenys į dvi grupes: vartotojo duomenys ir programos duomenys. Vartotojo duomenų saugojimas yra labai svarbus. Tačiau mobiliųjų įrenginių galimybės, lyginant su desktop sistemomis, yra labai ribotos. Dėl šių apribojimų mobilus įrenginiai neturi įprastos failų sistemos koncepcijos. J2ME (Java 2 Micro Edition) bazinės klasės skirtos darbui su duomenų saugykla, neatvaizduoja duomenų paskirstymo į vartotojo ir programos grupes.

Record Management System (RMS) tai įrašų pagrindu veikianti duomenų bazė. Įrašas – tai susijusios informacijos rinkinys apie esybę/objektą. Kiekvienas įrašas turi vienodą aibę laukų, kurių ilgiai yra fiksuoto dydžio. Įrašų skaitymas/rašymas atliekamas per unikalų įrašų identifikatorių recordId, kuris yra pirminis raktas. RMS sistema yra atsakinga už recordId valdymą: naujų išskirimą, unikalumo palaikymą. Įrašų skaitymas vyksta ne baitų lygmenys, o įrašų lygmenyje, t.y. per vieną kartą galima įrašyti arba nuskaityti tik vieną įrašą. Įrašo turinys nėra svarbus RMS sistemai, ji mato įrašus kaip baitų masyvą. Vienintelis dalykas kuris rupi RMS tai recordId. Įrašų koncepcija RMS sistemoje pavaizduota 1 paveikslėlyje.




**1 pav. Įrašų koncepcija RMS sistemoje.**

MIDletai yra pristatomi MIDletų rinkiniuose. Viename MIDletų rinkinyje yra mažiausiai vienas MIDletas. MIDletui sukūrus įrašų saugyklą, ji yra prieinama visiems MEDletų rinkinio MIDletams. Įrašų saugyklos pavadinimas turi būti unikalus MIDletų rinkinyje. MIDletas gali pasiekti įrašų saugyklą tik savo MIDletų rinkinyje.

Programos kūrėjai tūrėtų skirti dėmesio RMS veikimo greitaveikos analizei. Kadangi priklausomai nuo J2ME platformos realizacijos, skirtingos RMS funkcijos veikia skirtingu greičiu. Pvz. Kai kuriose sistemose dirbant su dideliais duomenų kiekais ir norint atlinkti tam

tikrų įrašų modifikacija yra efektyviau perskaityti visą saugyklos (RecordStore) turinį, ištrinti saugyklą ir sukurti naują, kurioje užsaugoti modifikuotus įrašus vietoj to, kad atnaujinti(update) įrašus skirtus modifikavimui.

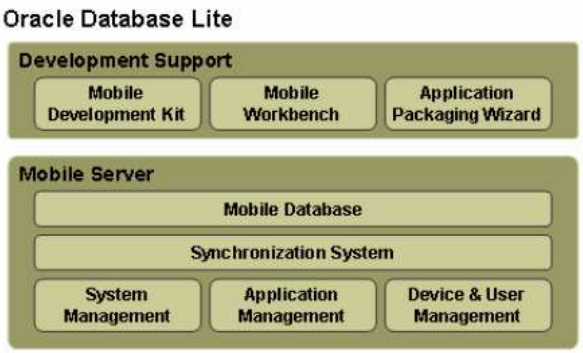
5. Microsoft SQL CE Server

SQL Server CE yra maža duomenų bazė skirt avis dažniau kuriamoms programoms kurios išplečia duomenų valdymo galimybes iki mobilių įrenginių. SQL Server CE yra pilnateisis narys tarp SQL Server 2000 šeimos produktų. Turi įrankius skirtus DB valdymui, programavimo sąsajas (API), ir SQL sintakse, kuri yra pažystama visiems programuotojams dirbantiems su SQL Server produktais. SQL CE yra vienintelis produktas iš SQL Server 2000 šeimos, kuris teikia reliacinės duomenų bazės galimybes Windows CE paremtiems įrenginiams. SQL CE turi:

- optimizuojantį užklausų procesorių
- Transakcijų palaikymą.
- Duomenų tipų palaikymą.
- Mažai naudoja sistemos resursų.
- Remote Data Access ir Merge Repclication per HHTP protokolą.

6. Oracle Database Lite

Oracle Database Lite yra priedas prie Oracle duomenų bazės naudojamas dažnam kūrimui ir diegimui didelės svarbos, mission-critical programoms skirtoms mobiliems įrenginiams. Oracle Database Lite naudoja duomenų sinchronizavimą, kad patikimai ir saugiau apsikeistų duomenimis tarp Oracle Database ir nutolusios aplinkos. Darbuotojai gali naudotis kompanijos informacija ir atlikti jiems riekiamas funkcijas būdami atsijungę nuo kompanijos duomenų bazės. Kompanijos naudojamos Oracle Database Lite gali apdidinti darbuotojų produktyvumą, sumažinti darbų sąnaudas, ar padidinti klientų pasitenkinimą. Sekančiame paveikslėlyje pateikta Oracle Database Lite sistemos vaizdas.




**2 pav. Oracle Database Lite sistemos sudėtis.**

7. DB2 Everyplace

IBM DB2 Everyplace yra IBM produktas teikiantis duomenų bazės paslaugas mobiliems įrenginiams. DB2 Everyplace susideda iš trijų pagrindinių komponentų, tokių kaip duomenų bazės variklis, kuris sukasi mobiliame įrenginyje, sinchronizavimo serveris ir pagrindinė duomenų bazė, kuri sukasi desktop PC arba enterprise serveryje.

DB2 Everyplace leidžia vartotojams įsidėti fragmentus pagrindinės duomenų bazės į savo mobilius įrenginius ir sinchronizuoti juos su pagrindine duomenų baze. Duomenų bazės variklis mobiliame įrenginyje yra gana nereikus resursams jam užtenka net 200KB atminties.

Prie DB2 Everyplace teikiamas DB2 Everyplace's MAB įrankis leidžiantis sukurti programas, naudojančias DB2 Everyplace duomenų bazę nerašant kodo. Šis įrankis sugeneruoja programą kuri gali būti pritaikyta konkrečiam mobiliam įrankiui arba tiesiog J2SE kodas, kuris gali būti vykdomas betuokiame įrenginyje.

Duomenų sinchronizavimams vyksta naudojantis HTTP protokolu, sinchronizavimo serveryje yra java servletas, kursi gali būti patalpintas betuokiame servletus palaikančiame serveryje (Tomcat, WebSphere ir pan.).

DB2 Everyplace teikia įvairias priemones dirbti su duomenų baze, tokais kaip C/C++, Java, Visual Basic, .NET bibliotekas.

8. Palyginimo rezultatai

Išanalizavimus duomenų bazių valdymo sistemas skirtas mobiliems įrenginiams palyginimo rezultatai yra pateikti žemiau esančioje 2 lentelėje.

**2 lentelė. Duomenų bazių valdymo sistemų mobiliems įrenginiams palyginimo kriterijai.**

| DBMS / Kriterijus | Java Record Management System | Microsoft SQL CE Server | Oracle Database Lite | DB2 Everyplace |
|---|---|---|---|---|
| Platforma | PocketPC, PalmOS, Symbian, kitos | Windows CE, Pocket PC, Windows Mobile | Windows CE, Pocket PC, Windows Mobile, Windows NT/XP/2000/98, PalmOS, Linux | Linux, Neutrino, PalmOS, Symbian, Windows, Windows CE, Windows Pocket PC |
| Atminties sunaudojimas. | Priklauso nuo JVM realizacijos | Nedidelis | 150KB – 1MB | 137 KB |
| Duomenų bazės dydis | Ribojamas mobilaus įrenginio laisvos atminties kiekiu | Iki 2 GB | Iki 4 GB | Ribojamas mobilaus įrenginio laisvos atminties kiekiu |
| Duomenų bazės tipas | Įrašų pagrindu | Reliacinė. | Reliacinė. | Reliacinė |
| Ryšys su kitomis DB | Nėra | Tik Microsoft SQL Server. | Oracle | Per “DB2 Everplace’s synchronization server” su **DB2**, Oracle, Microsoft SQL server, Domino and Exchange |
| Sinchronizavimo protokolas | Atskirai programuojamas | Vidinis (RDA, Merge Repclication). | Publish/Subscription modelis. Push modelis. | SyncML |
| Sinchronizavimo konfliktų sprendimas programiškai | Atskirai programuojamas | Yra | Yra | Yra |
| Sinchronizavimo konfliktų sprendimo vieta. | Programos projektuotojas sprendžia | Mobiliame įrenginyje, programiškai. | Sisteminė ir programuojama. | Sisteminė ir programuojama. |
| Duomenų bazių kiekis | Ribojamas mobilaus įrenginio laisvos | Neribotas. | Neribotas. | Neribotas |

| | | | | |
|---|---|---|---|---|
| | atminties kiekiu | | | |
| Lentelių kiekis | Ribojamas mobilaus įrenginio laisvos atminties kiekiu | Neribotas. | Neribotas. | Neribotas |
| Laukų kiekis | Ribojamas mobilaus įrenginio laisvos atminties kiekiu | Neribotas. | Neribotas. | Neribotas |
| Stored Procedūrų palaikymas | Nėra | Nėra | Yra. | Yra |
| SQL palaikymas | Nėra | Yra. | Yra. (SQL-92) | Yra. (SQL-99) |
| Palaikomi duomenų tipai. | Byte | INT, FLOAT, CHAR, VARCHAR. | Oracle DB tipai. | IBM DB2 tipai |
| Duomenų bazės valdymo įrankiai | Nėra | Yra (SQLCE Query) | Yra. | Yra |
| Programavimo priemonės. | J2ME API | ADOCE, ADO.NET, ADO. | JDBC, ADOCE, ADO, ADO.NET. | JDBC, ADO.NET |
| Index palaikymas | Nėra | Nėra. | Yra | Yra |
| Foreighn key palaikymas | Nėra | Yra. | Yra | Yra |
| Primary Key Palaikymas | Yra | Yra. | Yra | Yra |
| Sinchronizavimas iš kelių serverių. | Programuojamas atskirai | Nėra. | Nėra. | Nėra. |
| Sinchronizavimo stebėjimas | Nėra | Nėra. | Nėra. | Nėra. |
| Sinchronizavimas laukų. | Programuojamas atskirai | Nėra. Tik eilutės. | Nėra. | Yra |
| Programų kiekis | Neribotas | 1 programa prie 1 duomenų bazės. | Neribotas. | Neribotas. |
| Saugumas | Programuojamas atskirai | Duomeny bazės šifravimas. Autentifikavimas. | Autentifikavimas, Autorizavimas, šifravimas. | Autentifikavimas, Autorizavimas, šifravimas. |
| Kaina | Nemokama | Nemokamai SQL Server 2000 Developer Edition vartotojams. | Nemokama. | Nemokama. |

9. Išvados

- SQL CE tinka jei programinės įrangos pagrindinės duomenų bazės yra realizuotos SQL Server 2000 duomenų bazėse. Jei aplinkoje naudojamos Oracle ar kitokios duomenų bazės SQL CE nebeturi prasmės, nes negali su jomis sinchronizuoti duomenų.
- Nepaisant ribotų mobiliųjų įrenginių galimybių, RMS suteikia patogią ir lengvai naudojamą infrastruktūrą ilgalaikiam pastoviam duomenų kaupimui.

- Oracle Database Lite yra gana puikus sprendimas, jei yra poreikis naudoti įvairaus tipo mobiliuose įrenginiuose. Galimybėmis aplenkia Microsoft SQL CE Server. Vienas minusas, kad sinchronizuojasi tik su Oracle duomenų bazėmis.
- IBM DB2 Everyplace panašiai kaip Oracle Database Lite veikia įvairiose platformose teikia puikias sinchronizavimo galimybes. Bet taip pat kaip ir Oracle sinchronizuojasi tik su DB2 duomenų baze.

## Literatūros sąrašas


[1] Access data anywhere with Everyplace. [žiūrėta 2004-12-15], prieiga internete http://www.infoworld.com/DB2_Everyplace_Enterprise_8.1.4/product_46523.html?view=1&curNodeId=0

[2] Java Database Review Places PointBase At The Top. [žiūrėta 2004-11-09], prieiga internete http://wirelessdev.weblogsinc.com/entry/4868643664242743/

[3] Mobile Memories: The MIDP Record Management System. [žiūrėta 2004-11-14], prieiga internete http://today.java.net/pub/a/today/2004/11/16/J2ME-3.html

[4] Oracle Database Lite Overview. [žiūrėta 2004-11-10], prieiga internete http://www.oracle.com/technology/products/lite/lite_datasheet_10g.pdf

[5] Palm OS Database Applications. [žiūrėta 2004-12-13], prieiga internete http://www.the-gadgeteer.com/databases-review.html

[6] SQL Server 2000 Product Overview. [žiūrėta 2004-11-16], prieiga internete http://www.microsoft.com/sql/evaluation/overview/default.asp


## Data Stores Usage in Mobile Systems


Wide usage of mobile devices was quite useless because of the lack of resources in these devices and the lack of data storages for them. Information systems with mobile devices were unable to add additional value to theirs user for a given price. Though, situation has changed with increased resources and processing power on mobile devices. There are a lot of products of data storage systems in market. And developers of information systems encounter problems such as choosing a best Data Base Management System for mobile device.
In this article are described and analyzed data storages for mobile devices (Java Record Management System, DBMSD for Microsoft Pocket PC platform and Palm OS platform), and possibilities of using remote data bases.

# TINKLO RYŠIO PRIEMONĖS MOBILIUOSE ĮRENGINIUOSE

**Andrej Ušaniov, Packevičius Šarūnas, Eduardas Bareiša**

*Kauno Technologijos universitetas, Informatikos fakultetas, Programų Inžinerijos Katedra, Studentų g. 50, LT - 3031 Kaunas*

Rinkoje yra prieinamas platus asortimentas įvairiausių delninių kompiuterių, komunikatorių bei mobiliųjų telefonų. Šių įrenginių vartotojų poreikiai yra skirtingi, tačiau daugelį jų jungia bendras siekis – turėti galimybę keistis duomenimis tarp savo mobilaus įrenginio ir išorės.

Pranešime apžvelgiamos duomenų ryšių technologijos, priemonės bei būdai, kurie leidžia vartotojui atlikti duomenų mainus bei sinchronizavimą: IrDA, Bluetooth, GPRS, EDGE, GSM, HSCSD, Wi-Fi 802.11x.

## 1. Įžanga

Rinkoje yra prieinamas platus asortimentas įvairiausių delninių kompiuterių, komunikatorių bei mobiliųjų telefonų. Šių įrenginių vartotojų poreikiai yra skirtingi, tačiau daugelį jų jungia bendras siekis – turėti galimybę keistis duomenimis tarp savo mobilaus įrenginio ir išorės.

Labai svarbu projektuojant programinė įrangą mobiliems įrenginiams, pasirinkti tinkamą duomenų perdavimo technologiją, atsižvelgiant į pralaidumą, pasiekiamumą, patikimumą, energijos suvartojimą techninės įrangos, galimą darbo tinkle trukmę taip pat reikia įvertinti technologijos teikiamas saugumo galimybes.

## 2. GPRS

GPRS (General Packet Radio Service) – paketinis duomenų perdavimas GSM (Global System for Mobile Communications) tinklu technologija. Naudojantis GPRS duomenis galima perduoti 9,6 – 171,2kbps sparta, tačiau šiandiena Lietuvos rinkoje teikiamų GPRS paslaugų sparta siekia 30kbps. Dažniausiai įrenginio tipas nusako GSPR paslaugos panaudojimo greitį. Paslaugos kaina priklauso tik nuo parsiustų duomenų kiekio, tai yra naudinga , jei sesijos vyksta dažnai, o vienu metu perduodamas duomenų masyvas yra nedidelis. GPRS paslauga neturi garantuotos duomenų perdavimo spartos. Sesijos metu sparta gali kisti, o kartais gali būti lygi 0kbps, todėl GPRS paslauga nėra tinkama nenutrūkstamiems duomenų srautams.

## 3. EDGE

Enhanced Data Rates for Global Evolution (EDGE) yra skaitmeninių mobilių telefonų technologija, kuri tarnauja kaip išplėtimas GPRS tinklams. Ši technologija yra suderinama su TDMA ir GSM tinklais. EDGE naudoja tą pati spektrą skirtą GSM850, GSM900, GSM1800 ir GSM1900 darbui.

Vietoj to kad naudoti GSMK(Gaussian minimum-shift keying) EDGE naudoja 8PSK (8 Phase Shift Keying), sukurdama 3bit, žodį kiekvienam nešėjo fazės pasikeitimui. Tai efektyviai

patrigubina galima pralaidumą, kurį gali teikti GPRS. EDGE pristato naują technologija, kurios nėra GPRS, Incremental Redundancy, kuri vietoj tok kad persiustų pagadintus paketus, siunčiau daugiau perteklines informacijos. Tuo apdidinant sėkmingo dekodavimo tikimybę.

EDGE gali teiktis spartas iki 384 kbit/s paketiniame rėžime ir tuo pačiu tenkina International Telecommunications Union reikalavimus 3G tinklui, ir buvo IUT priimta kaip dalis IMT-2000 standartų šeimos 3G tinklui. EDGE taip pat pagerina duomenų rėžimą badinamą HSCSD padidindama iš šios paslaugos pralaidumą. EDGE pradedamas diegti į GSM tinklus visame pasaulyje nuo 2003 metų, pradedant JAV.

EDGE yra aktyviai palaikoma GSM operatorių JAV palyginus su visur kitur pasaulyje, nes GSM/GPRS turi stiprų konkurentą CDMA2000. Daugelis kitu GSM operatorių mato UMTS kaip tinkama atnaujinimą ir planuoja pralesit EDGE technologiją. Tačiau, dideli kaštai ir lėti UMTS žingsniai privertė kai kuriuos vakarų Europos GSM operatorius pereiti prie EDGE kaip tarpinio sprendimo.

EDGE teikia Enhanced GPRS (EGPRS), kuris gali būti naudojamas bet kurioms packed switched aplikacijoms, tokioms kaip Internet sujungimai. Didelės spartos reikalaujančios aplikacijos tokios kaip video paslaugos ir kitos multimedia gali pasinaudoti padidėjusia EGPRS' Sparta.

4. WAP

WAP (Wireless Application Protocol) – tai technologija, kurios dėka mobiliuoju telefonu galima pasiekti reikalingą informaciją internete, naudotis elektroniniu paštu. WAP protokolas "išverčia" internete esančią informaciją į mobiliesiems telefonams/įrenginiams suprantamą formatą. WAP naudojimas kartu su GPRS technologija padaro šia paslaugą greitesne, patogesne, pigesne.

5. HSCSD

HSCSD (High Speed Circuit Switched Data) - Didelės spartos duomenų perdavimo technologija. Sesijos metu yra užtikrinta pastovi duomenų perdavimo sparta ir atitinkamai trumpesnis duomenų perdavimo sesijos laikas, todėl ši technologija gerai tinka greitai perduoti didelius duomenų masyvus. Tačiau apmokėjimas už paslaugą taikomas ne už parsiustų duomenų kiekį, bet už paslaugos naudojimo trukmę. Todėl HSCSD technologijos naudojimas yra tikslingas kai reikia pasiekti iš anksto žinomus duomenis.

6. Bluetooth

Bluetooth technologija - tai balso bei duomenų perdavimas radijo bangomis, naudojant 2,4 GHz dažnių diapazoną, dar vadinamą ICM Industrial-Scientific-Medical (mokslo ir medicinos reikmėms skirtas dažnis). Prietaisai, naudojantys bluetooth ryšį, vienas su kitu "susisiekia" iki 10 m atstumu, ši zona vadinama „burbulu“. Bluetooth prietaisai gali būti jungiami po aštuonis į grupes, vadinamas „piconet“. Viename „burbule“ gali būti talpinama iki ą0 „piconetų“. Vienoje „piconet“ grupėje lygiagrečiai gali dirbti 3 balso perdavimo įrenginiai. Didžiausia bluetooth technologijos duomenų perdavimo sparta siekia 1Mb/s, tačiau įprastai darbui naudojami šie greičiai: 721/56Kbps asimetriniam duomenų perdavimui, 432Kbps „full duplex“ duomenų perdavimui. Bluetooth technologijos įmontuotas (build in) saugumas yra užtikrinamas koduojant 128 bitų ilgio raktų. Tačiau galima naudoti ir trumpesnius raktus. Bluetooth technologija yra taikoma šiems įrenginiams: mobiliems telefonams, pranešimų gavikliams, nešiojamiems bei delniniams kompiuteriams, tinklo prieigos įrenginiams, ausinėms ir t.t. Technologijos privalumai: įrenginių sujungimui nereikia išlaikyti nustatytos įrenginių padėties (automobilyje mobilusis telefonas ir laisvųjų rankų įranga išdėstomi nepriklausomai vienas nuo kito), vienu metu bluetooth tinkle gali veikti rinkinys įrenginių. Technologijos trukumas: papildomi veiksmai, reikalingi atlikti susijungimą tarp įrenginių. Didelėje konferencijos salėje, kurioje du partneriai atlieka apsikeitimą elektroninėmis vizitinėmis kortelėmis pasinaudojant Bluetooth technologiją. Susijungimas užtruks tam tikrą laiko tarpą, kol įrenginiai nustatys visus galimus įrenginius. Tačiau, nurodžius norimą įrenginį, susijungimas įvyks tik po to kai bus užtikrintas saugus duomenų perdavimas.

7. IrDA

IrDA (Infrared Data Association) – standartas IrDA duomenų perdavimus infraraudonų spindulių diapazone. IrDA technologija bevielio susijungimo atmaina įrenginiams, kurie įprastai naudoja kabelinius susijungimus. IrDA susijungimas yra taškas–į–tašką (point–to–point) pobūdžio. IrDA technologijos charakteristikos: atstumas tarp įrenginių iki 1 metro, veikimo kampas 30° kampu, duomenų perdavimo sparta nuo 9600bps iki 16Mbps (4Mbps yra dabartinė veikimo sparta, 16Mbps spartos veikimas yra vystomas). Įrenginių ratas, kuriem gali būti pritaikyta IrDA technologija: nešiojami, staliniai ir delniniai kompiuteriai, spausdintuvai, telefonai, modemai, kameros, medicininė bei pramoninė įranga, laikrodžiai. IrDA technologijos

privalumai yra: galimybė greitai sujungti du įrenginius ir atlikti duomenų mainus aukšta perdavimo sparta, sąlyginai pigi/paprasta realizacija. Prie šios technologijos trūkumų galima priskirti negalėjimą vienu metu apjungti daugiau negu du įrenginius. Tačiau privalumai ir trukumai priklauso nuo panaudojimo atvejo. Kaip pavyzdį panaudokime didelę konferencijos salę, kurioje du partneriai atlieka apsikeitimą elektroninėmis vizitinėmis kortelėmis. Naudojant IrDA technologija, mainai atlieka greitai ir paprastai, nepersidengiant su daugeliu kitų įrenginių kaip tai būtų Bluetooth technologijos atveju. IrDA technologijos panaudojimas neužtikrina perduodamų duomenų saugumo fiziniame lygmenyje. Tačiau IrDA technologija reikalauju tiesioginio matomumo zonoje taškas–į–tašką tipo susijungimo, kuomet šnipinėjimo galimybės yra minimalios. Susijungimo autentifikavimas ir duomenų kodavimas atliekami aukštesnio lygio protokoluose ir taikomųjų programų lygmenyje.

8. WIRELESS (Wi-Fi)

Bevieliai tinklai – tai galimybė mobiliajam vartotojui prisijungti prie bevielio tinklo, veikiančio tarp pastatų aibės, neapsiribojant kabelinio prisijungimo trūkumais. Bevielių tinklų technologijos standartas yra IEEE 802,11. Šios technologijos veikimo dažnis bei pralaidumas priklauso nuo technologijos standarto versijos. IEEE 802,11X technologijos standartų charakteristikos pateiktos lentelėje 1. Bevieliai tinklai plačiai taikomi oro uostuose, viešbučiuose bei kavinėse. Mobilusis įrenginys, jungdamasis į bevielį tinklą, užmezga ryšį su AP (Access point – prisijungimo vieta), tai yra su bevielio tinklo tarnybine stotimi (serveriu), kuris funkcionuoja kaip tiltas, jungiant bevielius ir paprastus vartotojus į vieną tinklą.

**Lentelė 8 IEEE 802,11X technologijos standartų charakteristikos**

| # | Standartas | Dažnių juosta | Veikimo spindulys | Pralaidumas |
|---|---|---|---|---|
| | 802.11b | 2,4 GHz | 45 – 90 m. | 11Mbps |
| | 802.11a | 5 GHz | 45 – 90 m. | 54Mbps, 48Mbps, 36Mbps, 24Mbps, 12Mbps, 6Mbps |
| | 802.11g | 2,4 GHz | Iki 300 m. | 54Mbps, 48Mbps, 36Mbps, 24Mbps, 12Mbps, 6Mbps |

9. Technologijų palyginimas

Norint įvertinti technologijų tinkamumą programinei įrangai mobiliems įrenginiams. Žemiau lentelėje pateikiamas palyginimas technologijų pagal įvairius kriterijus.

**Lentelė 2 Technologojų palyginimas**

| Kriterijus | IrDA | Bluetooth | Wi-Fi | GPRS | EDGE | HCSD |
|---|---|---|---|---|---|---|
| Atstumas | 10 m. (tiesiogiai) | 10 -100 m. | 50 m. | GSM tinklas | GSM tinklas | GSM tinklas |
| Palaikomos sąsajos | Pasirinktinai | USB, PCI | USB, PCI | Serial, USB | Serial, USB | Serial, USB |
| Perdavimo Sparta | Iki 4 Mbps | Iki 1 Mbps | 1 – 54 MBps | 9,6 – 171,2kbps | Iki 384 kbit/s | Iki 43.2 kbit/s |
| Energijos | Mažas | Vidutinis | Didelis | Mažas | Mažas | Mažas |

| | | | | | | |
|---|---|---|---|---|---|---|
| suvartojimas | | | | | | |
| Privalumai | Didelė sparta, žema kaina. | Saugumas. | Didelė perdavimo sparta. Saugumas. | Didelis pasiekiamumas, gera perdavimo sparta. | Didelis pasiekiamumas. | Didelis pasiekiamumas, . |
| Trūkumai | Vienas su vienu sujungimas, reikalingas tiesioginis matomumas | Mažas atstumas, sudėtinga programinė įranga. Nedidelė sparta. | Mažas atstumas, sudėtinga programinė įranga. Didelis energijos suvartojimas. | Nedidelė realisparta. | Nedaug operatorių teikia šią paslaugą. | Dažnas operatoriaus apmokestinimas už naudojimosi laiką. Vidutinė perdavimo sparta. |
| Tinkamumas | Gr eitam nedidelio duomenų kiekio pasikeitimui. Ribotu atstumu. | Gr eitam duomenų pasikeitimui tarp kelių įrenginių ribotu atstumu, jei reikia saugumo. | Gr eitam duomenų pasiektumui tarp daugelio įrenginių, nuolatinio darbo su tinklu programoms. | Nedideliam duomenų pasikeitimui įvairias laiko momentais. Programoms reikalaujančioms prisijungimo prie tinklo bet kuriu metu. | Dideliam duomenų pasikeitimui įvairias laiko momentais. Programoms reikalaujančioms prisijungimo prie tinklo bet kuriu | Nedideliam duomenų pasikeitimui įvairias laiko momentais. Programoms reikalaujančioms prisijungimo prie tinklo bet kuriu |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | metu. | metu. Jei naudojama sena techninė įranga nepalaikanti GPRS ar EDGE. |

10. Išvados

- Atsižvelgiant į technologijų teikiamas galimybes ir jų savybes jei programinė įranga reikalauja pastovaus darbo su tinklu, yra client-server tipo ir duomenų kiekis nėra didelis galima rinktis GPRS technologiją, jei programinė įranga nereikalauja didelio tinklo pasiekiamumo visą laiką, pavyzdžiui tik vienos organizacijos viduje, pastate, tai Wi-Fi technologija tampa gana patraukli dėl mažų ryšio sąnaudų ir galimybės perduoti didesnius duomenų kiekius.
- Jei programinė įranga reikalauja pastovaus darbo su tinklu, yra client-server tipo ir duomenų kiekis yra didelis galima rinktis EDGE technologiją.
- HCSD technologija galima pasirinkti tuo atveju jei techninė įranga yra senoka ir nepalaiko naujesnių technologijų tokiu kaip GPRS ar EDGE.
- Bluetooth ir IrDA technologijas galima pasirinkti jei reikalingas bendravimas tarp kelių mobilių įrenginių ir sistema yra ne client-server tipo, taip pat atsižvelgus į vienu metu bendraujančių įrenginių kieki galiama pasirinkti arba IrDa arba Bluetooth technologijas.
- Įvertinus technologijas realizuojančios techninės įrangos energijos sąnaudas, programinei įrangai, kuriai reikia ilgos darbo trukmės technologijos tokios kaip Wi-Fi naudojimas yra netinkamas, nes didelis energijos sunaudojimas.

Literatūros sąrašas

[1] Bluetooth™ and Other Wireless Technologies [žiūrėta 2004-12-18], prieiga internete http://www.informit.com/articles/article.asp?p=24265.

[2] EDGE Introduction of high-speed data in GSM/GPRS networks [žiūrėta 2004-12-10], prieiga internete http://www.ericsson.com/products/white_papers_pdf/edge_wp_technical.pdf.

[3] Finding an RF Solution [žiūrėta 2004-12-11], prieiga internete http://www.maxstream.net/spotlight/find-rf-solution/t4#at4.

[4] Wireless Standards Technologies [žiūrėta 2004-12-15], prieiga internete http://wireless.ittoolbox.com/nav/t.asp?t=426&p=426&h1=426.

Network Access Means in Mobile Devices

There is a large choice of hand held computers, communicators and cell phones in market. Users of these devices needs are quite different, a lot of them are joined by common need – a possibility to interchange data between theirs mobile devices and environment.

This article give a quick review of data exchange technologies and means which enables users to perform data exchange and synchronization, such as IrDA, Bluetooth, GPRS, EDGE, GSM, HSCSD, Wi-Fi 802.11x.

# GRAFINĖS VARTOTOJO SĄSAJOS AUTOMATIZUOTAS TESTAVIMAS MOBILIAI ĮRANGAI SU APRIBOJIMAIS.

**Andrej Ušaniov, Rūta Makūnaitė, Šarūnas Packevičius**

*Kauno Technologijos universitetas, Informatikos fakultetas, Programų Inžinerijos Katedra, Studentų g. 50, LT - 3031 Kaunas*

**Naujuose programiniuose produktuose plačiai taikoma mobilioji įrangą (mobilieji telefonai, delniniai kompiuteriai). Programinės įrangos kūrimo procese svarbią vietą užima testavimas. Dabartinių mobiliųjų įranginių apribojimai (darbo sparta, atminties kiekis, energija, ekrano dydis, platformų įvairumas), sukelia naujas problemas programinės įrangos kūrimo procesui, tame tarpe ir testavimui. Testavimo proceso automatizavimas leidžia sumažinti bandymų trukmę, padidinti testavimo darbų apimtį. Programinę įrangos mobiliems įrenginiams bandymuose svarbią vietą užima grafinė vartotojo sąsaja.**

**Šiame pranešime panagrinėsime grafinės vartotojo sąsajos automatizuotą testavimą mobiliai įrangai, apsibrėšime bandymo tikslus bei jų siekimo būdus.**

## 1. Įžanga

Sparčiai besivystanti mobilioji įranga, tokia kaip mobilūs telefonai bei delniniai kompiuteriai, imama plačiai taikyti naujuose programiniuose produktuose. Tačiau mobilioji įranga turi daug trukumų, tokių kaip santykinai maža procesoriaus darbo sparta, ribotas atminties kiekis, mažas ekranas, sudėtingas duomenų įvedimo procesas. Mobiliosios įrangos apribojimai komplikuoja programinės įrangos kūrimą bei testavimą.

Dažnai kuriama programa mobiliai įrangai yra orientuota į taikymą daugelyje mobiliųjų platformų. Tačiau plati mobiliųjų įrenginių įvairovė kelia tam tikrų problemų. Todėl programos išdirbėjams tenka atsižvelgti į mobiliųjų įrenginių architektūros bei konfigūracijos skirtumus. Mobiliųjų įrenginių skirtumai komplikuoja grafinės vartotojo sąsajos testavimą. Viename įrenginyje puikiai veikianti vartotojo sąsaja gali būti visiškai nepriimtina kitame įrenginyje.

Kuriant bei testuojant programinę įranga naudojami mobiliųjų įrenginių emuliatoriai, kurie neperteikia visų mobilaus įrenginio galimybių, apribojimų bei darbo ypatumų. Todėl yra svarbu išbandyti grafinę vartotojo sąsają tiesiogiai mobiliajame įrenginyje ar skirtingų mobiliųjų įrenginių grupėje.

Straipsnyje nagrinėjami grafinės aplinkos mobiliajai įrangai testavimo ypatumai, užbrėžiami automatizuoto grafinės sąsajos testavimo tikslai bei jų siekimo būdai.

## 2. Vartotojo sąsaja mobilioje aplinkoje

Vienas iš svarbiausių mobilios įrangos trukumų yra mažas ekrano dydis. Todėl viename programos lange yra sudėtinga atvaizduoti ar įvesti daug informacijos. Dėl šitos priežasties, kuriant vartotojo sąsaja, stengiamasi viename programos lange patalpinti ribotą laukų kiekį. Dažniausiai viename lange vartotojas gali įvesti tik vieno parametro reikšmę. Tokiu būdu paprastas parametrų įvedimo langas stacionariame kompiuteryje pavirsta į langų aibę mobiliajame įrenginyje. Projektuojant mobiliųjų įrenginių sąsaja, iš programos langų sudaromas navigacijos grafas, kuriame mazgas atitinka programos langą, o briauna atitinką perėjimą tarp langų [1]. Langų seka, per kurią vartotojas pereina, atlikdamas operaciją, vadinamas scenarijumi.

Navigacija mobilioje aplinkoje turi savo ypatumu: programos vartotojas turi visada sugebėti išeiti ir/arba pareiti iš programos lango į kitą, todėl lange privalo būti išėjimo (grįžimo atgal) mygtukas, taip pat papildomai mygtukas, leidžiantis atlikti perėjimą į sekantį langą [2].

Programos bandymas realių įrenginių aibėje gali užtrukti ilgą laiko tarpą. Be to, kiekvienas įrenginys gali turėti savo ypatumų, tokių kaip atskirų langų naudojimas lauko reikšmės įvedimui bei atvaizdavimui. Todėl grafinė sąsaja skirtinguose įrenginiuose bus atvaizduojama skirtingai – tai dar labiau apsunkina testavimo procesą.

Grafinės vartotojo sąsajos testavimas reikalauja iš žmogaus daug pastangų: reikia stebėti ne tik programos reakciją į vartotojo veiksmus, bet ir tikrinti programos langų atitikimą reikalavimams. Testavimas rankiniu būdu trunka ilgą laiko tarpą. Regresinis testavimas yra praktiškai neįmanomas arba labai brangus.

## 3. Testavimo uždaviniai

Mobilios programinės įrangos grafinės vartotojo sąsajos automatizuotam testavimui nustatysime šiuos uždavinius:

- Išbandyti atskiro programos lango funkcionalumą [3]
    - Nustatyti leidžiamų įvesties simbolių aibę

    - Navigacijos galimybės: išėjimas/grįžimas atgal, perėjimas į sekantį langą
- Programos langų grafo išbandymas
    - Automatizuotas scenarijų vykdymas
    - Patikrinti programos langų pasiekiamumą
    - Realaus programos navigacijos grafo atitikimas suprojektuotam
    - Nustatyti projektavimo metu nenumatytus programos langus
- Navigacijos dokumentacijos generavimas pagal realų navigacijos grafą išgautą iš realaus įrenginio
- Išmatuoti vėlinimą – laiko trukmė nuo vartotojo veiksmo pradžios iki atvaizduoto rezultato
- Susiejimas su kitais testavimo būdais: vienetų testavimas, sistemos testavimas, regresinis testavimas

## 4. Problemos

Ribotas mobiliųjų įrenginių atminties kiekis, verčia kaupti testavimo atvejų duomenys išorėje. Tai gali būti stalinis kompiuteris, turintis pakankamai resursų testavimo atvejų saugojimui.

Kodo perkodavimas (obfuscation). Siekiant sumažinti išeities kodo apimti, atliekamas kodo perkodavimas, kai visu klasių, metodų bei kintamųjų vardai yra pervadinami į sutrumpintus. Toks kodas užima ženkliai mažiau atminties, tačiau yra neskaitomas žmogui. Perkodavimo metu nesudaroma vardų atitikimo lentelė, kurioje būtų nurodyti atitikimai tarp senų bei naujų vardų. Taip pat toks kodas neleidžia pasinaudoti atspindžio technologija (reflection).

Šiuo metu programavimo platformos mobiliajai įrangai (J2ME, Mobile.NET) dėl įrenginių apribojimų nepalaiko atspindžio technologijos (reflection) [4], kurios pagalba galima kurti objektus žinant jų klasės pavadinimą. Todėl, norint pasiekti programos langus bei jų turinį, reikalinga taikomosios programos sąsaja (API), kuri turės atspindžio technologijos funkcionalumą ir leis apeiti kodo perkodavimo problemas.

Automatizuotam testu atlikimui turi būti naudojama speciali aplinka (framework) [5]. Testavimo aplinka atsakinga už testuojamos programos metodų kvietimą. Pagal pateiktą navigacijos grafą bei scenarijus testavimo aplinka atlikinės perėjimus tarp langų, perduodamą testinius duomenys programai bei fiksuodama rezultatus.

## 5. Sprendimo būdai

Išorinėje duomenų bazėje yra saugomi programos navigacijos grafas, programos langų aprašymas, testavimo atvejai bei testavimo rezultatų istorija. Išorinės duomenų bazės panaudojimas leidžia išspręsti mobilaus įrenginio riboto atminties kiekio problemą. Iš duomenų saugyklos į testavimo aplinka perduodamas navigacijos grafas bei scenarijai. Kiekvienam navigacijos mazgui perduodami testavimo atvejų duomenys.

Atspindžio problema sprendžiama taikant programinius interfeisus (API). Šių interfeisų pagalba testavimo aplinkoje bus pasiekiami visi programos lango duomenų įvedimo/išvedimo laukai, jų reikšmės bei komandiniai mygtukai. Per programinę sąsaja gauti lango parametrai lyginami su suprojektuotais. Tokiu būdu bus nustatoma ar langas atitinka grafinės vartotojo sąsajos reikalavimus [3]. Taip pat API bus taikomas ir metodams skirtiems atlikti logines operacijas.

Automatizuoto testavimo aplinka – tai savarankiška programa, veikianti mobiliajame įrenginyje. Testavimo aplinka naudoja išorinę duomenų saugyklą testavimo duomenims gauti. Pagal paduotus duomenis: navigacijos grafą, scenarijus ir testinius duomenis, API pagalba kviečiami testuojamosios programinės įrangos metodai. Testavimo aplinka valdys duomenų perdavimą bei fiksuos rezultatus.

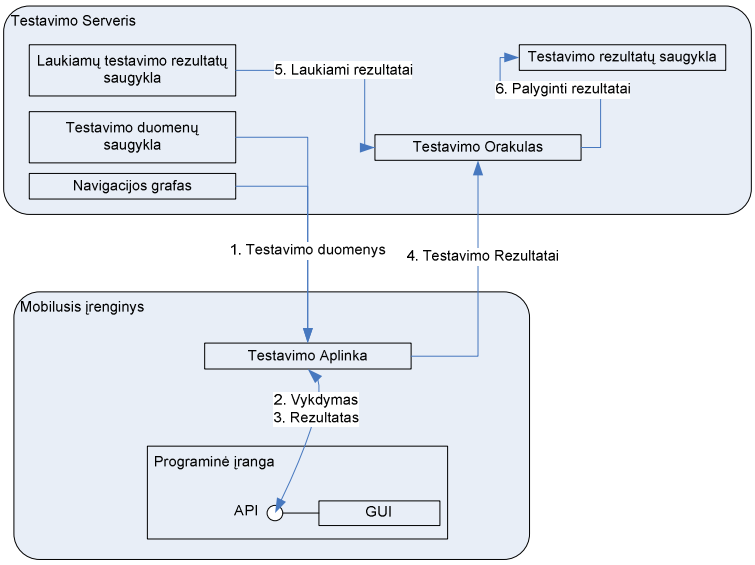
Atliekamas kiekvieno vaizduojamo lango bandymas. Tikrinami duomenų įvedimo laukai, leistinos įvesties simbolių aibės. Nustatoma ar visi laukai yra vaizduojami ekrane.

Pagal scenarijų bandoma atlikti perėjimą iš vieno lango į kitą, testavimo aplinka fiksuoja perduotus į metodą duomenys bei gautus rezultatus: loginės operacijos reikšmė bei atvaizduotas langas. Taip pat testavimo aplinka fiksuoja vėlinimą: laiko trukmė per kuria programa reaguoja į vartotojo veiksmus. Perėjimas tarp langų turėtų užtrukti ne ilgiau nei dvi sekundės. Gauti rezultatai perduodami į testavimo orakului. Testavimo orakulas atlieka gautų bei laukiamų rezultatų palyginimą.

Paveikslėlyje 5.1. yra pateikta grafinės vartotojo sąsajos automatizuoto testavimo aplinkos struktūrinė schema. Schemoje pavaizduoti duomenų srautai tarp modulių:

1. Testavimo duomenys: navigacijos grafas bei testavimo atvejai. Kiekvienam navigacijos mazgui yra sudaryti testavimo atvejai. Remiantis suprojektuotu navigacijos grafu testavimo aplinka valdys programinę įrangą.
2. Vykdymas. Per API į įvedimo laukus paduodami testavimo duomenys.

3. Rezultatas. Fiksuojama programinės įrangos reakcija į įvestus duomenys. Nustatomas loginės operacijos rezultatas bei sekantis atvaizduotas langas.
4. Testavimo rezultatai grąžinami į testavimo serverį. Testavimo rezultatai nusakys lango duomenų laukus, leidžiamas įvedamų simbolių aibes, galimus perėjimus iš programos lango.
5. Laukiami rezultatai – pagal programos projektą parengti duomenys, nusakantys kiekvieno lango duomenų laukus, leidžiamų įvedimo simbolių aibes, galimus parėjimus iš programos lango.
6. Testavimo rezultatai palyginami su laukiamais rezultatai. Nustatomas realus programos navigacijos medis. Testavimo rezultatai saugojami saugykloje.




**Paveikslėlis 5.1 Grafinės vartotojo sąsajos automatizuoto testavimo aplinkos struktūrinė schema**

Testavimo aplinkos galimybė fiksuoti programos reakciją į paduodamus duomenys bus pritaikyta atliekant testavimą rankiniu būdu: vartotojas dirbs su programa, įvedinės duomenys, o testavimo aplinka fiksuos programos reakciją į vartotojo veiksmus.

## 6. Automatizuotos grafinės sąsajos testavimo aplinkos realizacija

Automatizuota testavimo aplinka buvo sudaryta ir taikyta kuriant programinę įrangą „Mobilaus vadybininko darbo vieta“. Programinė įranga buvo sukurta naudojant Java 2 Micro Edition (J2ME) technologija. Kadangi J2ME technologija nėra pririšta prie platformos, programinė įranga buvo išbandyta keliose platformose: Pocket PC tipo delniniame kompiuteryje bei mobiliuose telefonuose Nokia 3510i, Nokia6100 ir Sony-Erricson 630T.

Testavimo aplinka buvo skirta atlikti vienetų testavimą (unit testing). Duomenys tarp testavimo serverio bei mobilaus įrenginio buvo perduodami GPRS tinklo ryšiu taikant XML formatą. Testavimo serveris realizuotas Jacarta Tomcat taikomųjų programų serveryje taikant Java 2 Enterprise Edition (J2EE) technologiją.

Dabartinis realizacijos etapas reikalauja testavimo aplinkos išplėtimo, kuris leistų valdyti testavimo procesą remiantis navigacijos grafu bei scenarijais. Navigacijos grafas bei scenarijai bus aprašomi XML kalba.

## 7. Perspektyvos

Ateityje planuojama plėsti testavimo aplinkos funkcionalumą. Numatomi plėtimai:

- Iš projektavimo metu sudarytų klasių bei sekų diagramų generuoti testavimo scenarijus.
- Iš realaus navigacijos grafo, t.y. to, kuri atvaizduoja įrenginys, generuoti vartotojo dokumentacija. Dokumentacijoje bus pavaizduoti langai, atitinkantys konkretų mobilaus įrenginio modelį.

## 8. Išvados

- Automatizuotas vartotojo sąsajos testavimas padės testuotojams išvengti rutininio darbo ir susikoncentruoti ties testavimo atvejų sudarymu
- Automatizuotas testavimas leidžia atlikti regresinį testavimą

## Literatūros sąrašas


[1] **Jackwind Li Guojie** , Build your stock with J2ME,. [žiūrėta 2004-09-11], prieiga intenete: http://www.ibm.com/developerWorks

[2] **V. Piromian**, Wireless J2ME Platform Programming, *Sun Microsystems Press*, 2002, ISBN-0-13-044914-8

[3] Guidelines for Testing J2ME Applications, [žiūrėta 2004-012-11], prieiga intenete: http://www.nokia.com

[4] **M. Ancona, W. Cazzola**, Implementing the essence of reflection: a reflective run-time environment, *Symposium on Applied Computing archive. Proceedings of the 2004 ACM symposium on Applied computing*, 2004, 1503-1507 p.

[5] **I. Satoh**. A Testing Framework for Mobile Computing Software. *IEEE Transactions on Software Engineering*, 2003.


## Automated GUI testing for mobile devices with constraints


Mobile devices such as cell phones and personal digital assistance are widely used in new software products. Testing takes important place in software development process. Constraints of mobile devices (speed, amount of memory, energy, small screen, wide range of platforms) raise new problems for software development process including testing phase. Automated approach of software testing reduces testing time and increases testing range. It is important to distinguish graphic user interface as an important part of testing. The aim of this article is analyze automated testing of GUI for mobile devices, define testing tasks and ways to complete them.