KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Skirmantas Balčiūnas

Kombinuotosios vartotojo sąsajos projektavimas ir tyrimas

Magistro darbas

Darbo vadovas
prof. dr. Eduardas Bareiša

Kaunas, 2006

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Skirmantas Balčiūnas

Kombinuotosios vartotojo sąsajos projektavimas ir tyrimas

Magistro darbas

Kalbos konsultantė
doc. dr. J. Mikelionienė
2006-05-

Vadovas
prof. dr. Eduardas Bareiša
2006-05-

Recenzentas
doc. dr. K. Motiejūnas
2006-05-

Atliko
IFM-0/2 gr. stud.
Skirmantas Balčiūnas
2006-05-15

Kaunas, 2006

Multimodal user interface design and research

SUMMARY

A number of software that supports multiple interactions such as synergistic use of speech and gesture is increasing. With power and versatility of multimodal interfaces also comes and new issues like increased complexity of software design. These issues should be considered in early design and testing stages of software development process.

In this work we are researching a complex use of users multiple interactions with software. We will represent a method for the specification of multimodal user interfaces offered by other authors. This method is based on CARE properties. The CARE properties provide a formal framework for reasoning design of multimodal systems. Multimodal systems are using fusion mechanisms to combine different modalities.

Later in this work we will take a closer look at testing process of multimodal applications. A new approach for testing modalities fusion mechanism in multimodal applications is offered. This approach will be experimentally demonstrated, evaluated and compared to alternative method.

Turinys

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

1. ĮVADAS

Pastaruoju metu ypač daugėja programinės įrangos gebančios bendrauti su vartotoju per daugiau nei vieną vartotojo sąsają, panaudojant šnekos, gestų atpažinimo arba akies vyzdžio sekimo technologijas. Kombinuotos sąsajos vis dažniau pritaikomos tokiose įvairiose gyvenimo srityse kaip medicinos [25], kariuomenės [6] ar telekomunikacijų [16]. Dauguma mobiliųjų telefonų siūlo naudoti dviejų tipų įvesties modalumus: įprastą klaviatūrą ir balso atpažinimą. Galima tikėtis, kad netolimoje ateityje dauguma vartotojo sąsajų taps kombinuotomis – kelių modalumų sąsajomis.

Kombinuota vartotojo sąsaja suteikia vartotojui pasirinkimo laisvę t. y. leidžia vartotojui pasirinkti jam priimtinesnį, atsižvelgiant į užduotį ar aplinkos pokyčius, bendravimo su programine įranga būdą. Žmonės bendraudami tarpusavyje naudoja įvairius modalumus: šneką, gestus, veido išraiškas ir pan. Paprastai kombinuotos vartotojo sąsajos leidžia derinti įvairius modalumus tarpusavyje, tokiu būdu perkeliant žmogaus bendravimą su techniką į žmogui daug natūralesnį lygi.

Kadangi, ne visada vienas vartotojo sąsajos tipas, kitaip tariant – modalumas, tam tikru atveju yra priimtiniausias ar tinkamiausias vartotojui, tinkamai suprojektuota kombinuota vartotojo sąsaja tampa labai svarbiu visos sistemos panaudojamumo faktoriumi.

Šnekos atpažinimo technologijomis pagrįstas sąsajos valdymas atlaisvina vartotojo rankas, kas yra pakankamai svarbu jei vartotojas vienu metu vykdo keletą darbų. Šnekos atpažinimas dažniausiai naudojamas sąsajos valdymui balsu, t. y. komandoms vykdyti, meniu, mygtukams aktyvuoti [27, 23]. Tuo tarpu kada kaip įvedimo priemonė pasirenkamas pieštukas, jis dažniausiai naudojamas įvairiems simboliams ar grafiniams objektams įvesti [24, 22, 29].

Nors jau ir yra nemažai programinės įrangos su kombinuotom vartotojo sąsajom, tačiau tokios programinės įrangos kūrimas yra iš prigimties vis dar sudėtingas [18]. Nepakankamas supratimas, kaip keli vartotojo sąsajų tipai sąveikauja tarpusavyje, dažnai priveda prie mažo galutinio produkto vartotojo sąsajos patogumo [5].

Šiame darbe tiriama kombinuotos vartotojo sąsajos projektavimo specifika ir problemos. Analizuojami įvairių autorių pasiūlyti metodai kombinuotosioms sąsajoms specifikuoti ir testuoti, kurie padeda programinės įrangos kūrėjams įnešti daugiau aiškumo programinės įrangos kūrimo procese. O tai, savo ruožtu, sąlygoja ir geresnę galutinio produkto kokybę.

Eksperimentiškai pagrindžiamas pasirinktas kombinuotų sąsajų testavimo metodas.

Pagrindinė darbe sprendžiama problema yra programinės įrangos kelių įvedimo būdų kompleksinis panaudojimas. Nagrinėjama kombinuotų vartotojo sąsajų projektavimo metodologija. Kombinuotoms vartotojo sąsajoms specifikuoti siūlomos CARE savybės, kurios leidžia aiškiai apibrėžti vartotojo sąsajų atliekamus veiksmus ir sąsajų tarpusavio ryšius. Skirtingų sąsajų veiksmams apjungti naudojamas modalumų suliejimo mechanizmas. Modalumų suliejimo mechanizmui realizuoti pasiūlomas komponentinis kombinuotų vartotojo sąsajų projektavimo metodas – ICARE.

Tiriamos kombinuotos vartotojo sąsajos automatizuoto testavimo galimybės, palyginami galimi tokio testavimo metodai. Eksperimentiškai pagrindžiamas pasiūlytas testavimo metodas.

2. ANALITINĖ DALIS

2.1. Bandomasis projektas

Norint ištirti programinės įrangos valdymo balsu specifiką ir problemas buvo realizuota informacinė sistema turinti šnekos atpažinimo funkciją. Projekto realizacijos metu buvo analizuojama valdymo balsu funkcijos įtaka reikalavimų specifikavimui, projektavimui, realizavimui, testavimui, diegimui ir palaikymui.

Sukurta informacinė sistema skirta vesti kompensuojamosios technikos apskaitą. Sistemos vartotojai naudodamiesi programinę įranga gali atlikti šias pagrindines funkcijas:

- Vartotojų pateiktų prašymų kompensacinei technikai registravimas, redagavimas, šalinimas, paieška, rūšiavimas.
- Sutarčių sudarymas, redagavimas, šalinimas, paieška, rūšiavimas.
- Kompensacinės technikos pajamavimas, išdavimas.
- Spausdinimui tinkamas ataskaitų apie priemonių judėjimą formavimas, spausdinimas.
- Sistemos vartotojų administravimas.

Sistemą galima pilnai valdyti balsu. Sukurtos valdymo balsu priemonės gali būti panaudojamos ir kituose panašiuose projektuose, kuriuose yra poreikis valdyti programinę įrangą ne vien tik tokiomis įprastomis valdymo priemonėmis, kaip pelė ir klaviatūra, bet ir naudojant valdymą balsu.

Sistemoje menių punktai ir mygtukai gali būti aktyvuojami ištarus jų užrašus. Teksto įvedimo laukai aktyvuojami ištarus šalia jų matomus pavadinimus, o tekstas įvedamas diktuojant žodžius paraidžiui.

2.2. Nagrinėjamos problemos

Šiame darbe pagrindinė sprendžiama problema yra kelių įvedimo būdų kompleksinis panaudojimas. Kombinuotų vartotojo sąsajų kompleksinis panaudojimas realizuojamas per sąsajų suliejimo mechanizmą, kuris dar 1980-aisiais metais buvo paminėtas R. Bolt darbe „Put-that-here“ [3]. Šio mechanizmo realizavimas yra ganėtinai sudėtingas.

Skirtingų sąsajų suliejimo mechanizmas turi būti labai griežtai ir atidžiai suprojektuotas, realizuotas ir ištestuotas.

Deja, bet šiuo metu yra vos keletas specializuotų projektavimo įrankių, skirtų kombinuotom sąsajom projektuoti, tačiau ir pastarųjų efektyvaus panaudojimo galimybės yra labai ribotos, nes jie yra arba skirti labai specifinėm techninėm problemom, tokiom kaip terpių sinchronizavimas [15], arba skirti specializuotiem vartotojų sąsajų tipam. Pavyzdžiui, „Artkit“ projektavimo įrankis yra skirtas tik sąsajoms valdomom pele, klaviatūra ir papildant valdymą gestų atpažinimu [13].

Darbe [11] yra pasiūlytas ganėtinai nesudėtingas kombinuotos sąsajos realizavimo karkasas, skirtas turimos programinės įrangos funkcionalumo praplėtimui įdiegiant kelis įvedimo būdus.

2.3. Modalumas

Modalumas – bendravimo, sąveikos metodas tarp vartotojo ir programinės įrangos [8]. Modalumą galima apibrėžti kaip porą <į, k>, kur „į“ yra naudojama techninė įranga, o „k“ bendravimo, sąveikos kalba. Sąveikos kalba apibrėžiama kaip baigtinė aibė elementų su taisyklėmis. Sąveikos kalbos elementus geba priimti ir išvesti įvesties bei išvesties techninė įranga [20].

Pavyzdžiui, yra galimi tokie modalumai:

- Įvestis balsu: <mikrofonas, pseudo-natūrali kalba> - čia pseudo-natūrali kalba tai apibrėžtas taisyklių ir žodžiu rinkinys (žodynas);
- Grafinė įvestis pele: <kompiuterinė pelė, tiesioginė manipuliacija>;
- Grafinė išvestis: <ekranas, langai ir kiti grafiniai elementai>;

Taigi modalumas apibrėžia duomenų apsikeitimo tarp vartotojo ir programinės įrangos tipą ir formą.

Sistemos požiūriu, daugialypis modalumas yra sistemos gebėjimas bendrauti su vartotoju įvairiais ryšio kanalais [19].

2.4. Kombinuota vartotojo sąsaja

Prieš pradedant nagrinėti kombinuotos sąsajos savybes yra svarbu suprasti, kuo skiriasi kombinuota (daugiamodalė) vartotojo sąsaja nuo įprastos grafinės vartotojo sąsajos:

- Grafinės vartotojo sąsajos paprastai vienu metu geba priima veiksmus tik iš vieno veiksmų šaltinio. Pavyzdžiui, dauguma grafinių vartotojo sąsajų ignoruos duomenų įvedimą klaviatūra tol, kol yra paspaustas pelės klavišas. Tuo tarpu kombinuotoje vartotojo sąsajoje veiksmai gali būti priimami lygiagrečiai iš skirtingų modalumų.
- Grafinėse vartotojo sąsajose visi atliekami veiksmai yra atominiai ir nedviprasmiški, tuo tarpų kombinuotose vartotojo sąsajose veiksmai gali būti kombinuojami tarp skirtingų sąsajų tipų. Šias kombinacijas reikia atskirai įvertinti ir nustatyti, kokį būtent veiksmą vartotojas nori atlikti.
- Grafinės vartotojo sąsaja dažniausiai yra atskirta į atskirą vaizdavimo lygį ir yra nepriklausoma nuo sistemos atliekamų funkcijų verslo logikos lygyje. Tačiau vartotojo sąsajos paprastai veikia tame pačiame kompiuteryje kartu su visa sistema. Kombinuotų vartojo sąsajų modalumai reikalauja daug didesnių sistemos resursų (pavyzdžiui, naudojant šnekos atpažinimą). Resursai yra ypač riboti mobiliuosiuose įtaisuose, tad pastaruoju metu vis dažniau projektuojant kombinuotas vartotojo sąsajas yra svarstoma galimybė parskirstyti sistemą tinkle perkeliant įvesties duomenis apdorojančius komponentus į atskiras tarnybines stotis.

Programinėje įrangoje su kombinuota vartotojo sąsaja kiekviena sąsaja gali būti naudojama nepriklausomai nuo kitos, tačiau galimybė vienu metu pasinaudoti keliomis sąsajomis gali sukelti nemažai problemų. Kombinuotos sąsajos panaudojimas naudojant daugiau nei vieną sąsajos tipą (pavyzdžiui, naudojant klaviatūrą ir šnekos atpažinimą) labai stipriai padidina galimų veiksmų sekų aibę.

2.4.1. Kombinuotos sąsajos aspektų charakterizavimas CARE savybėmis

Darbe [8] pateikiamas būdas kaip kombinuotą vartotojo sąsaja galima apibrėžti CARE savybėmis. CARE savybės suteikia formalias priemones sistemoms su kombinuotomis sąsajomis projektuoti. CARE tai angliškų žodžių *Complementary* (papildymas), *Assignment* (paskyrimas), *Redundancy* (pertekliškumas) ir *Equivalence* (lygiavertiškumas) akronimas.

Lygiavertiškumas

Aibės *M* modalumai yra laikomi lygiaverčiais tada, kai iš vienos būsenos norint pasiekti kitą būseną yra galimybė pasirinkti vieną iš modalumų. Aibė *M* turi būti sudaryta

bent iš dviejų modalumų. Lygiavertiškumas suteikia pasirinkimo laisvę tarp modalumų tam pačiam tikslui pasiekti.

Vienas iš modalumų lygiavertiškumo pavyzdžių KTAS projekte yra tada, kai vartotojas turi pasirinkimo laisvę norėdamas sukurti naują sutartį ir gali paspausti mygtuką su užrašu „Sukurti“ arba ištarti sakinį „Mygtukas sukurti“.

Taigi lygiavertiškumo savybė suteikia sistemai daugiau lankstumo: įvedimui vartotojas gali rinktis jam patogesnę sąsaja. Lygiavertiškumas dažnai padeda ir tais atvejais, kai dėl tam tikru laikinų trukdžiu vartotojas negali naudotis viena iš sąsajų. Pavyzdžiui, aplinka tampa pernelyg triukšminga, kad būtų galima naudotis šnekos atpažinimu arba vartotojo rankos tampa užimtomis, jei tuo metu reikia laikyti kokias nors pagalbines darbo priemones. Tuo metu praverčia valdymas balsu.

Paskyrimas

Modalumas laikomas paskirtu tuomet, kai norint iš vienos būsenos pasiekti kitą nėra galimybės pasirinkti jokio alternatyvaus modalumo ir yra būtina pasinaudoti tuo vieninteliu galimu modalumu.

Kaip modalumų paskyrimo pavyzdys KTAS projekte gali būti laikomas langų padėties valdymas. Langų padėti galima valdyti naudojant vienintelį įvesties modalumą – kompiuterinę pelę.

Pažymėtina, kad modalumų lygiavertiškumo ir paskyrimo savybės apibrėžia tik modalumo pasirinkimo galimybę tam tikroje stadijoje, būsenoje. Tuo tarpu pertekliškumo ir papildomumo savybės apibrėžia kombinuotą modalumų panaudojimą laikinaisiais ryšiais.

Pertekliškumas

Aibės *M* modalumai laikomi pertekliškais tada, kai modalumai yra lygiaverčiai ir yra panaudoti viename laikinajame lange (nustatytame laiko intervale). Pertekliškumas reikalauja lygiavertiškumo egzistavimo. Tarkim, kad vartotojas vieno laikinojo lango metu aktyvuoja lygiaverčius veiksmus skirtingomis sąsajomis (modalumais). Jei sistema nepalaiko pertekliškumo apdorojimo, tada tas pats veiksmas bus atliktas du kartus. Pertekliškumo apdorojimas yra svarbus programinės įrangos projektavimo etape ir turi būti aiškiai apibrėžta, kaip sistema turi elgtis aptikusi modalumų perteklišką panaudojimą.

Pertekliškumas pirmą kartą buvo aprašytas atliekant WOZ eksperimentą darbe [28]. Darbe [9] teksto įvedimo ir šnekos atpažinimo pertekliškumas yra išskiriamas į dvi kategorijas:

- Pažodinis sutapimas. Pavyzdžiui, klaviatūra įvedamas tekstas tiksliai sutampa su diktuojamuoju balsu.
- Santrumpinis sutapimas. Pavyzdžiui, klaviatūra įvedamas tekstas yra santrumpa teksto diktuojamo balsu.

Papildymas

Aibės *M* modalumai laikomi papildantys vienas kitą tuomet, kai norint iš vienos būsenos pasiekti kitą vieno laikinojo lango metu yra būtinai panaudojami visi aibės *M* modalumai. Atskiras individualus aibės M modalumų panaudojimas nesuteikia galimybės pasiekti kitą būseną. Sistema aptikusi vienas kitą papildančius veiksmus iš skirtingų modalumų dažniausiai paleidžia sąsajų suliejimo mechanizmą (žiūrėti skyrių 2.4.2 kur yra detaliau aprašomas sąsajų suliejimo metodas). Vienas kito modalumų papildymo pavyzdys KTAS projekte gali būti tada, kai vartotojas ištaria frazę „redaguoti šią sutartį“ ir tuo metu pele pasirenka vieną sutartį iš sutarčių sąrašo.

Projektuojant kombinuotas sąsajas svarbu atkreipti dėmesį ir į tai, kad skirtingi vartotojo veiksmai atlikti iš skirtingų sąsajų tame pačiame laikinajame lange nebūtinai yra papildantys vienas kitą. Tokiu atveju neturėtu būti paleistas ir sąsajų suliejimo mechanizmas. Darbuose [19, 17] yra detaliau aprašoma, kaip spręsti šias sąsajų suliejimo problemas.

CARE savybės gali būti naudojamos įvairiuose kombinuotų sąsajų projektavimo etapuose. Pavyzdžiui, anksčiausiuose etapuose CARE savybėmis galima specifikuot pirminius bendriausius kombinuotos vartotojo sąsajos bruožus. Vėliau, suprojektavus sistemą gali prireikt labiau detalizuotos CARE specifikacijos. Baigus projektavimo darbus, naudojant aprašytas CARE savybes gali atlikti programinės įrangos panaudojamumo tyrimą arba atlikti automatizuotą kombinuotos sąsajos testavimą.

CARE savybės gali būti naudojamos sistemai apibrėžti skirtinguose sistemos abstrakcijos lygiuose [21]:

- Sistemos funkcijų lygyje.
- Vartotojo poreikių lygyje.

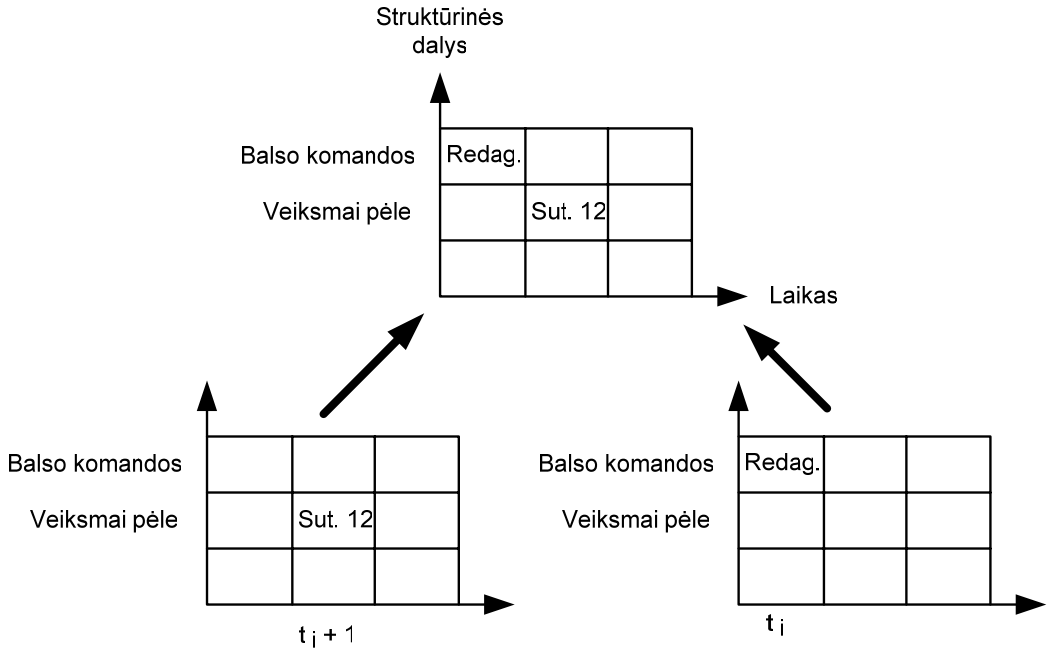
Darbe [8] rašoma, kaip vartotojo poreikiai sistemoms su kombinuotomis sąsajomis siejasi su sistemos funkcijų savybėmis. Papildomas savybių apibrėžimas ir vartotojo poreikių

lygyje leidžia dar sistemos projektavimo etape numatyti koks bus sistemos panaudojamumas. Pabrėžiama, kad papildomai specifikuojant kombinuotų sąsajų savybes vartotojo poreikių lygyje dažnai išsprendžiamos tokios problemos kuomet vartotojas patenka į aklavietę negalėdamas pasirinkti norimo veiksmo su vienu iš vartotojo sąsajos tipu (modalumu) arba kuomet vartotojas tiesiog nežino apie galimybę, kartais galbūt net efektyvesnę, atlikti tą patį veiksmą alternatyviu modalumu.

2.4.2. Sąsajų suliejimo metodas

Sąsajų suliejimo metodas yra pagrįstas bendru suliejimo objektu. Kaip pavaizduota 2.1 pav., suliejimo objektas yra dvimatės matricos struktūros, kur eilutės arba vertikali ašis skirta žymėti struktūrines dalis, kurias sistema gali apdoroti, o stulpeliai arba horizontali ašis – tai laiko žymės, skirtos sužymėti, kuriuo laiko momentu buvo iškviesta vartotojo veiksmų tam tikra struktūrinė dalis.

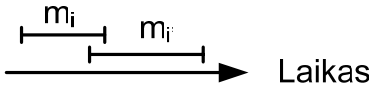
Panagrinėkime pavyzdį, kaip veiktų sąsajų suliejimo mechanizmas KTAS projekte: laiko momentu t_1, kai vartotojas ištaria sakinį „redaguoti šia sutarti“, o laiko momentu t_1+1 su pele iš sutarčių sąrašo pasirenka sutartį „Sutartis nr. 123“. Paveikslėlio 2.1 kairėje, apačioje vaizduojamas šnekos atpažinimo varikliuko iškviestas ir sugeneruotas suliejimo objektas, o dešinėje pelės veiksmo iškviestas sugeneruotas objektas. Suliejimo mechanizmas sulieja šiuos du objektus į vieną ir gaunamas naujas objektas su sulietais veiksmais, kuriuos turi įvykdyti sistema.



2.1 pav. Dviejų objektų suliejimo mechanizmas

Iškvietus suliejimo mechanizmą yra bandoma pritaikyti vienas iš trijų suliejimo tipų šia tvarka: mikrosuliejimas, makrosuliejmas, kontekstinis suliejimas.

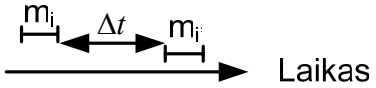
- Mikrosuliejimas yra aktyvuojamas tada, kai duomenys yra gauti lygiagrečiai t. y. tuo pačiu arba labai artimu laiko momentu. Suliejimas vykdomas tada kai aptinkami vienas kitą papildančių veiksmai ir jų laiko intervalai persidengia.



2.2 pav. Du objektai galintys patekti į mikrosuliejimą

- Makrosuliejimas įvykdomas esant tokiom pačiom sąlygom kaip ir mikrosuliejimo atveju, tačiau sujungia duomenis gautus viename laikinajame lange. Makrosuliejimas dažnai aktyvuojamas kuomet sistema dėl resursų nepakankamumo ar kitų priežasčių trumpai užlaiko duomenų perdavimą programinei įrangai.

  Jei pažymėtumėm, kad laikinojo lango trukmė yra Δt, duomenų d_1 laiko žymė yra t_1, o duomenų d_2 laiko žymė t_2, tai mikrosuliejimas įvyks tuomet kai $t_2 \in [t_1 - \Delta t, t_1 + \Delta t]$. Δt (laikinojo lango trukmė) paprastai yra nustatoma eksperimentiniu būdu – pagal sistemos našumą arba vartotojo įpročius.



2.3 pav. Du objektai galintys patekti į makrosuliejimą

- Kontekstinis suliejimas vykdomas neatsižvelgiant į laiko žymes, o suliejant duomenis tik pagal veiksmų kontekstą ir eiliškumo seką. Pavyzdžiui, KTAS projekte vartotojas pele pirmiausia gali pažymėti tam tikrą sutartį sutarčių sąraše, o vėliau grįžęs po kurio laiko prie kompiuterio ištarti „redaguoti šią sutartį“.

Suliejimo mechanizmas ne tik išpildo kombinuotos sąsajos modalumų vienas kito papildymo savybes, bet ir atlieka pertekliškumo savybių patikrinimą. Šis patikrinimas atliekamas prieš bandymą pritaikyti mikrosuliejimą. Pertekliškumas aptinkamas tada, kai

randami skirtingų suliejimo objektų (iš skirtingų modalumų) tie patys duomenis gauti tuo pačiu arba labai artimu laiko momentu. Tokiu atveju parenkamas tik vienas iš jų pagal aukštesnį prioritetą turintį modalumą. Jei modalumų prioritetai nenaudojami arba yra vienodi parenkamas pirmasis gautas.

Suliejimo mechanizmas turi būti universalus ir gebėti apdoroti įvairaus tipo struktūrines dalis.

Plačiau pats suliejimo mechanizmas ir sąlygos kuriomis yra įvykdomas modalumų suliejimas yra nagrinėjamas darbe [20].

Priklausomai nuo pasirinktos strategijos, suliejimo mechanizmas gali būti aktyvuotas iškart kuomet tik gaunama pakankamai duomenų (ankstyvasis suliejimas) arba aktyvavimas gali būti užlaikytas iki laikinojo lango pabaigos su tikslu surinkti visus galimus duomenis (tingus suliejimas).

2.5. Komponentinis kombinuotų vartotojo sąsajų projektavimas

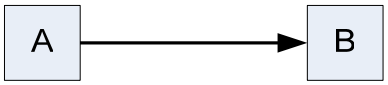
Darbe [4] buvo pasiūlytas komponentinis kombinuotų vartotojo sąsajų projektavimo metodas, dar vadinamas kaip ICARE (Interaction-CARE). Jame naudojamas modalumų suliejimo mechanizmas, kuris nepriklauso nuo konkretaus naudojamo modalumo. Šis projektavo metodas pagrįstas dviejų tipų programinės įrangos komponentais:

- Elementarieji komponentai. Šiam tipui priklauso įrangos komponentai ir sąveikos kalbos komponentai.
- Sudėtiniai komponentai. Šie komponentai realizuoja kombinuotą modalumų panaudojimą (suliejimo mechanizmą) CARE savybėmis (žiūrėti skyrių 2.4.1). Šie komponentai nėra priklausomi nuo konkretaus modalumo ir gali apjungti duomenis gautus iš dviejų ar daugiau modalumų.

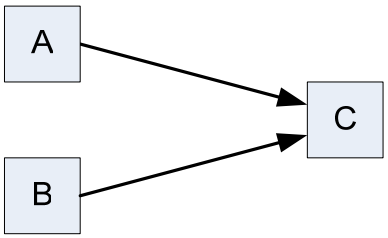
Elementarieji (įrangos ir sąveikos kalbos) komponentai įneša papildoma abstrakcijos lygį tarp techninės įrangos ir sistemos. Jie papildo įvesties techninės įrangos generuojamus veiksmus tokiais atributais kaip atlikto veiksmo laiko žymė, pasikliaujamumo faktorius ir pan.

Konceptualiajame ICARE modelyje autoriai sudėtinius komponentus skirsto į tris skirtingomis CARE savybėmis pasižyminčius komponentus: papildomumo, pertekliškumo ir pertekliškumo/lygiavertiškumo. Kiekvienas iš šių komponentų turi savo suliejimo mechanizmą.

Lygiavertiškumo ir priskyrimo CARE savybių komponentai nėra atskirai išskirti nes šios savybės yra išreiškiamos ryšiais tarp kitų komponentų. Pavyzdžiui, paveikslėlyje 2.4 yra pavaizduoti du komponentai susieti ryšiu kuris ir simbolizuoja priskyrimo savybę t.y. alternatyvaus pasirinkimo nebuvimą. O paveikslėlyje 2.5 pavaizduota lygiavertiškumo savybė – kuomet bent du komponentai yra susieti su vienu ir tuo pačiu komponentu. O tai reiškia, kad šie komponentai yra lygiaverčiai.



2.4 pav. ICARE komponentas A priskirtas komponentui B



2.5 pav. Du lygiaverčiai A ir B ICARE komponentai priskirti komponentui C

Sudėtiniai komponentai vykdo suliejimo mechanizmą aprašytą skyriuje 2.4.2. Suliejimas aktyvuojamas aptikus duomenų pertekliškumą ar papildomumą.

Susiejus elementariuosius komponentus su sudėtiniais komponentais pagal pastarųjų CARE savybes galima efektyviai valdyti modalumų suliejimą t.y. aptikti veiksmų papildymus tarp tam tikrų komponentų (atskirų modalumų), bei eliminuoti aptiktus pertekliškumus tarp modalumų.

3. PROJEKTINĖ DALIS

3.1. Sistemos paskirtis

Informacinė sistema skirta vesti kompensuojamosios technikos apskaitą. Sistemos vartotojai naudodamiesi programine įranga gali atlikti šias pagrindines funkcijas:

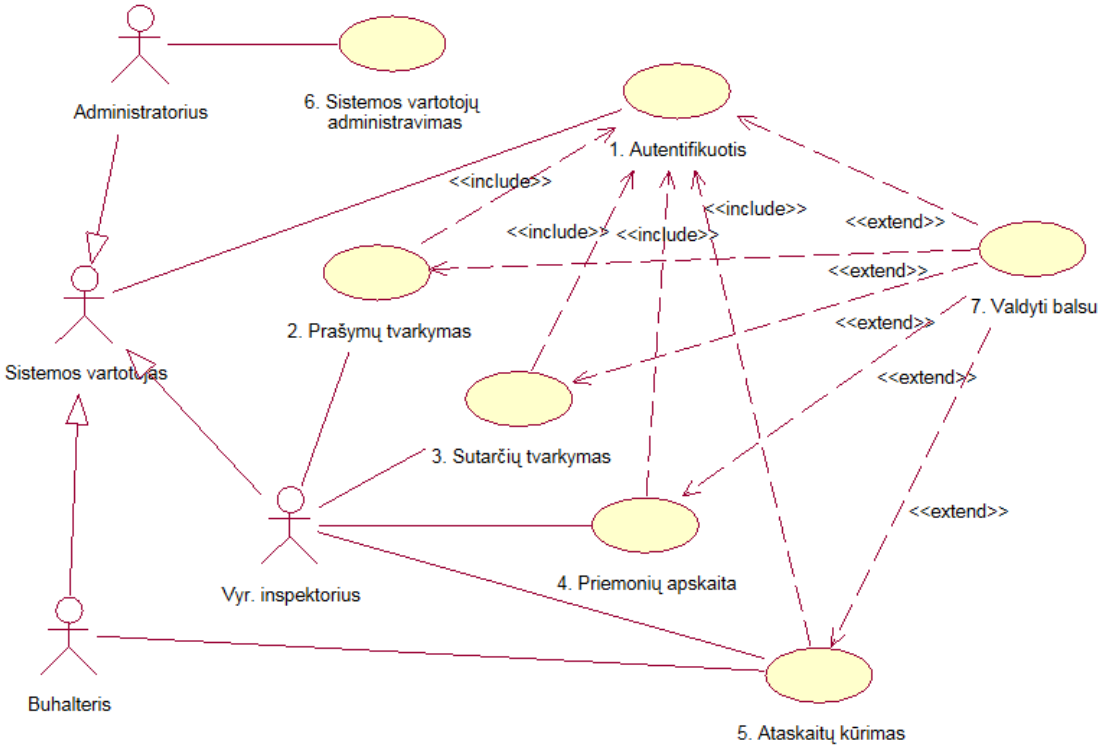
- Įvairūs veiksmai su vartotojų prašymais kompensacinei technikai.
- Įvairūs veiksmai su sutartimis.
- Įvairūs veiksmai su kompensacine technika.
- Formuoti spausdinimui tinkamas ataskaitas apie priemonių judėjimą.
- Kurti kitus sistemos vartotojus.

Šiai sistemai sukurtos valdymo balsu priemonės gali būti panaudojamos ir kituose panašiuose projektuose kur yra poreikis valdyti programinę įrangą ne vien tik įprastomis valdymo priemonėmis, tokiomis kaip pelė ir klaviatūra, bet ir naudojant valdymą balsu. Panaudojus sukurtas valdymo balsu priemones, sistemos funkcionalumas praplėstas įdiegus alternatyvų valdymą balsu. Tai suteikė vartotojams pasirinkimo laisvę, bei galimybę naudotis sistema efektyviau kombinuojant skirtingų sąsajų tipus tarpusavyje.

Sistemoje menių punktai ir mygtukai gali būti aktyvuojami ištarus jų užrašus. Teksto įvedimo laukai aktyvuojami ištarus šalia jų matomus pavadinimus, o tekstas įvedamas diktuojant žodžius paraidžiui.

3.2. Esminiai reikalavimai

Reikalavimai sistemai yra specifikuoti naudojant Volere šabloną. Vartotojo atliekamos funkcijos pateikiamos panaudos atveju diagrama, kurioje aiškiai matomos bendros sistemos atliekamos funkcijos vartotojų atžvilgiu (3.1 pav.).



3.1 pav. Sistemos panaudos atvejų diagrama

Šio darbo kontekste aktualiausias yra 7-tas panaudos atvejis, kuris praplečia kitus pagrindinius panaudos atvejus. Panaudos atvejai yra detalizuoti į funkcinius ir nefunkcinius reikalavimus.

Pagrindiniai funkciniai reikalavimai iškelti panaudos atvejui „Valdyti balsu“:

- Sistema turi leisti įvesti duomenis diktuojant juos balsu.
- Sistema turi leisti aktyvuoti meniu punktus ir mygtukus ištarus jų užrašus.

Vienas iš svarbiausių nefunkcinių reikalavimų iškeltų valdymui balsu buvo, kad šnekos atpažinimo funkcija turi veikti esant nedideliam pašaliniam triukšmui, nes darbo aplinkoje galimas nedidelis pašalinis triukšmas, kuris neturi įtakoti sistemos veikimo. Todėl šnekos atpažinimo tikslumas turi siekti nemažiau kaip 90% esant nedidesniam nei 25 db triukšmo lygiui.

Kitas svarbus nefukcinis reikalavimas buvo, kad turi būti nesudėtingas pakartotinis šnekos atpažinimo komponento panaudojimas kituose projektuose. Į tai buvo atsižvelgiama jau pradiniuose sistemos kūrimo etapuose.

3.3. Architektūra

Sistemos architektūra pateikiama keliais vaizdais:

- panaudojimo atvejų,
- statinis,
- dinaminis
- išdėstymo.

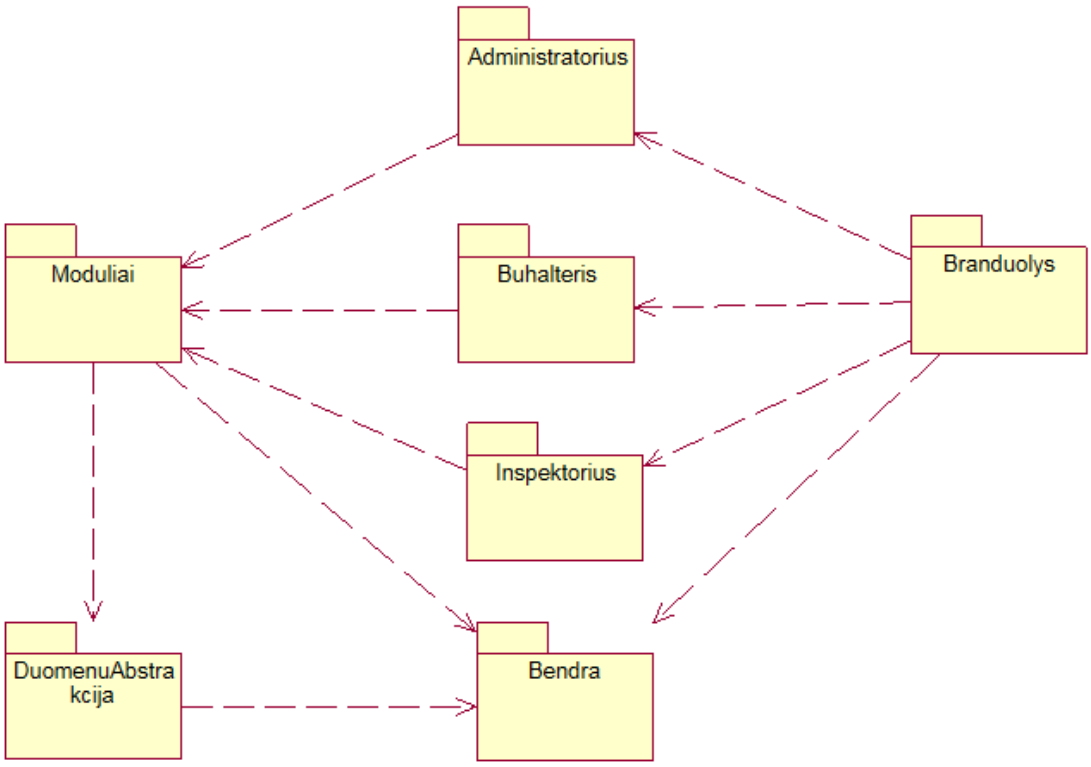
Šie vaizdai yra pateikiami kaip *Rational Rose* modeliai naudojant unifikuota modeliavimo kalba (UML). Sistemos architektūra pateikta remiantis RUP (*Rational Unified Process*) rekomendacijomis. Sistemos specifikacija pateikta šiais vaizdais kuriems įgyvendinti reikia UML diagramų:

- Panaudojimo atvejų vaizdas (panaudojimo atvejų diagrama)
- Sistemos statinis vaizdas (paketai ir klasių diagramos)
- Sistemos dinaminis vaizdas (būsenų, veiklos, sekų, bendradarbiavimo diagramos)
- Išdėstymo vaizdas (išdėstymo diagrama)

Architektūrinius sprendimus įtakoję reikalavimai:

- Sistemas turi būti suprojektuota taip, kad ją galima būtų lengva išplėsti ar prijungti naujus modulius.
- Kuriama sistema bus pateikta kaip atviro kodo, nekomercinė programinė įranga.
- Sistema neturi leisti neautorizuotiems vartotojams prie jos prisijungti.
- Sistema šnekos atpažinimui naudos Speech API 5.1
- Sudarant sistemos architektūrą, turi būti atsižvelgta į būtinas programos vykdymo charakteristikas, apibrėžtas reikalavimų specifikacijoje.

Statiniame vaizde sistema yra išskaidyta į paketus aukščiausiame lygyje. Sistemos suskaidymas į paketus pateiktas 3.2 paveikslėlyje.



3.2 pav. Sistemos išskaidymas į paketus aukščiausiame lygyje

Pakete *Branduolys* pateikiamos klasės kurios sudaro sistemos branduolį t.y. per jas yra iškviečiami kiti sistemos objektai. Į paketą įeina ir šnekos atpažinimą visoje sistemoje realizuojanti klasė.

Pakete *Inspektorius* pateiktos klasės realizuojančios vyr. inspektoriaus panaudos atvejus.

Pakete *Buhalteris* pateikiamos buhalterio panaudos atvejus realizuojančios klasės.

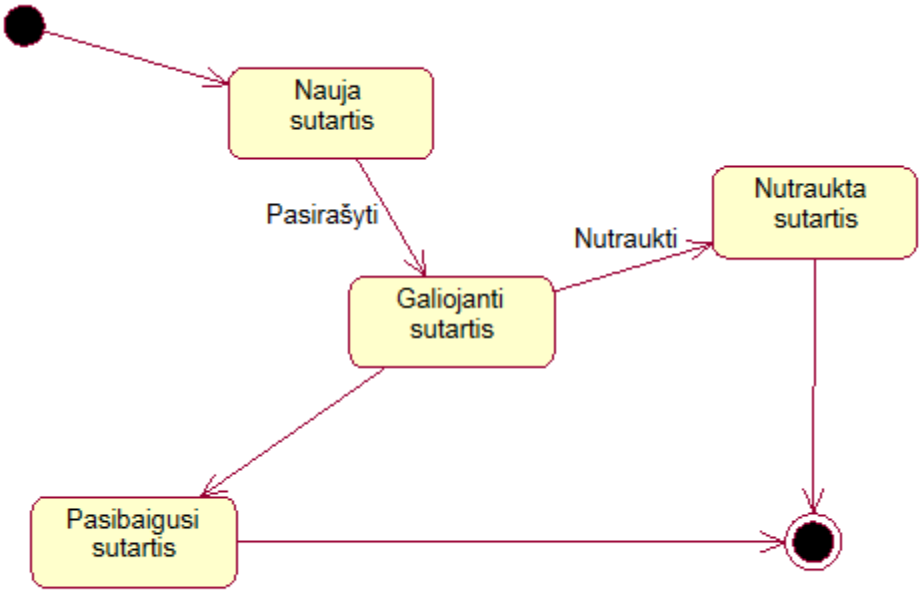
Pakete *Administratorius* pateikiamos klasės skirtos administratoriaus sąsajos realizacijai.

Pakete *Bendra* pateikiamos bendros paskirties klasės kurios yra naudojamos kituose paketuose arba abstrakčios klasės kurios negali būti priskirtos kažkuriam kitam paketui.

Paketas *DuomenuAbstrakcija* skirtas duomenų bazės abstrakcijos klasėms. Tai klasės skirtos darbui su duomenų baze. Jos naudojamos daugumoje kitų paketų.

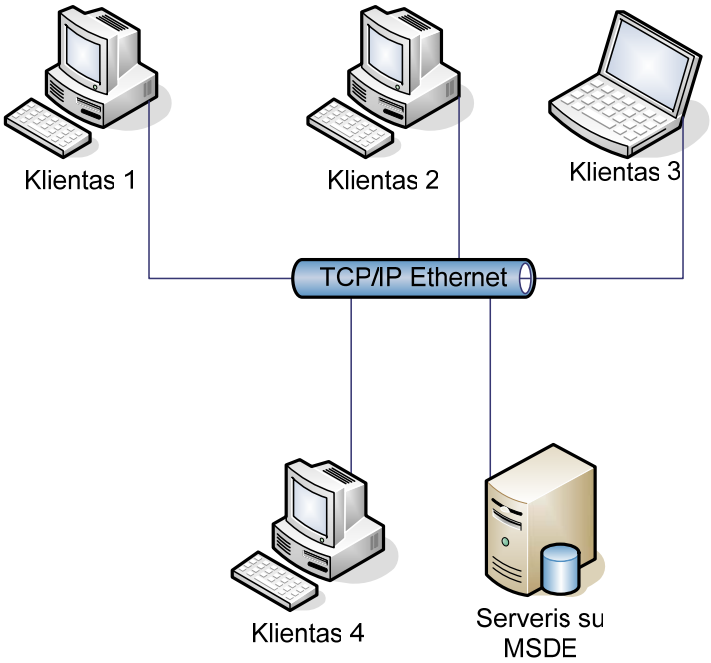
Pakete *Moduliai* saugomos klasės realizuojančios tokias sistemos teikiama funkcijas kaip prašymų, sutarčių tvarkymas, priemonių apskaitos vedimas, ataskaitų generavimas, spausdinimas ir t.t. Sistemą nesunkiai galima praplėsti naujais moduliais. Šis paketas, pagal modulius yra suskirstytas į žemesnio lygio paketus.

Sistemos procesų vaizde pateikiamos sistemos objektų būsenų, veiklos, sistemos elementų bendradarbiavimo bei sekų diagramos. Viena, pavyzdinė, sistemos esybės *Sutartis* būsenų diagrama pateikiama sekančiame paveikslėlyje.



3.3 pav. Esybės „Sutartis“ būsenos diagrama

Paveikslėlyje 3.4 pateikta sistemos išdėstymo diagrama. DBVS gali veikti ir kliento kompiuteryje.



3.4 pav. Sistemos išdėstymas

Duomenų bazė diegiama tarnybinėje stotyje su Microsoft Windows XP/2000/2003 operacine sistema. Duomenų bazės valdymo sistema – Microsoft SQL Server Desktop Engine.

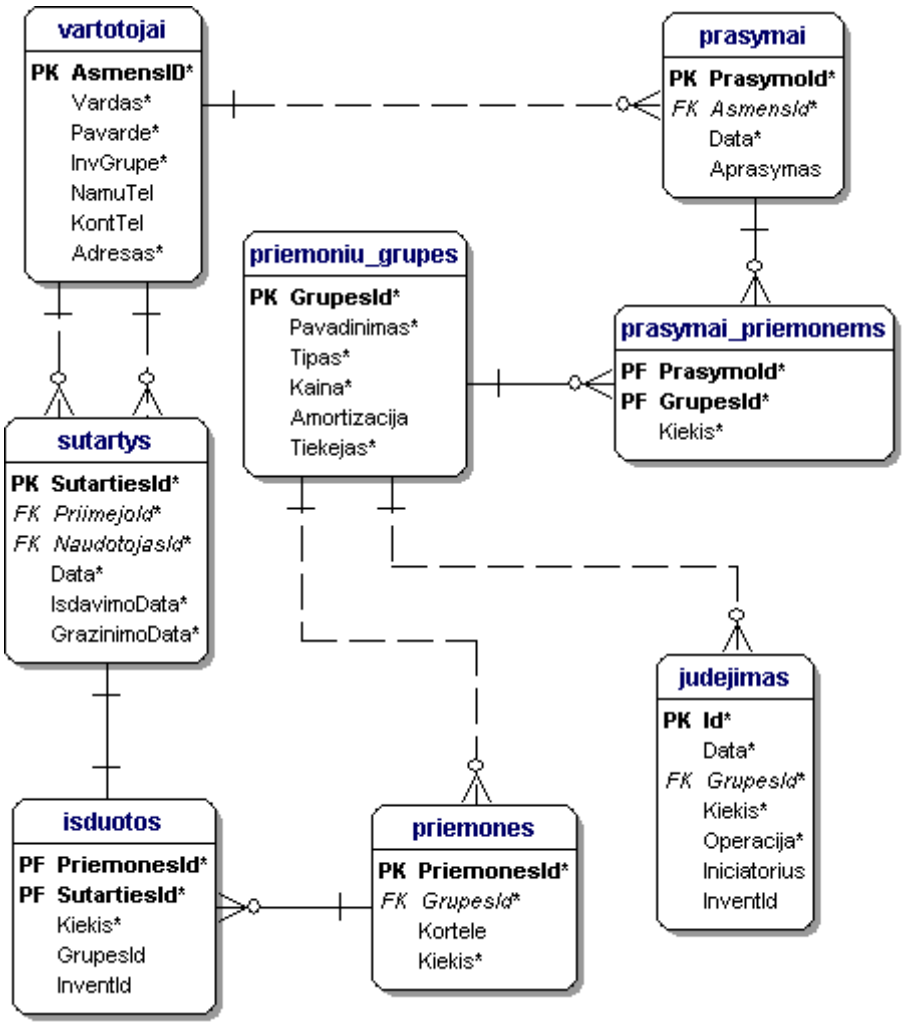
Tarnybinėje stotyje yra reikalingas tinklo palaikymas. Microsoft SQL Server 2000 Desktop Engine teikia paslaugas programiniame lygyje naudojant TDS (Tabular Data Stream) protokolą (transporto ir tinklo lygyje – TCP/IP).

Informacinė sistema realizuota .NET platformoje ir naudoja Microsft .NET Framework.

Kliento kompiuteryje turi būti Microsoft Windows XP/2000/2003 operacinė sistema su įdiegta Microsoft .NET Framework 1.1. virtualia mašina, bei SAPI 5.1

3.4. Duomenų bazė

Duomenų bazės valdymo sistemai yra pasirinkta MSDE (Microsoft SQL Server 2000 Desktop Engine) duomenų bazės valdymo sistema. Duomenų bazės modelis pateiktas paveikslėlyje 3.5.



3.5 pav. Duomenų bazės modelis

4. TYRIMO DALIS

4.1. Kombinuotų sąsajų testavimas

Sistemų su kombinuotomis vartotojo sąsajomis kūrime jaučiamas žinių, metodologijų ir įrankių trūkumas ne tik projektavimo etape, bet ir testavime. Šiame skyriuje patyrinėsim sistemų su kombinuotomis vartotojo sąsajomis testavimo galimybes ir metodus.

Testuojant kombinuotas vartotojo sąsajas pagrindinis dėmesys yra skiriamas sąsajų kompleksinio panaudojimo t.y. sąsajų suliejimo mechanizmo testavimui. Testavimo tikslas – aptikti klaidas apjungiant duomenis iš skirtingų vartotojo sąsajų.

Kombinuotų vartotojo sąsajų testavimas, kaip ir projektavimas, remiasi CARE savybėmis.

Kombinuotos vartotojo sąsajos didina programinės įrangos efektyvumą ir naudojimo lankstumą. Tačiau tokios sąsajos didina ir programinės įrangos sudėtingumą, nes reikia apdoroti didelius kiekius įvairių tipų vartotojo įvesčių. Sąsajos testavimas, savo ruožtu, taip pat tampa sudėtingesnis, nes reik įvertinti dar didesnius kiekius galimų vartotojo veiksmų ir žmogiškųjų išteklių panaudojimas šios sąsajos testavimui tampa per ilgas ir per brangus. Būtent dėl šios priežasties planuojant kombinuotų sąsajų testavimą reikia įvertinti automatizuoto testavimo galimybes.

Sekančiuose poskyriuose paanalizuosim porą automatizuotų kombinuotos vartotojo sąsajos testavimo metodų.

4.1.1. ICARE komponentų testavimo karkasas

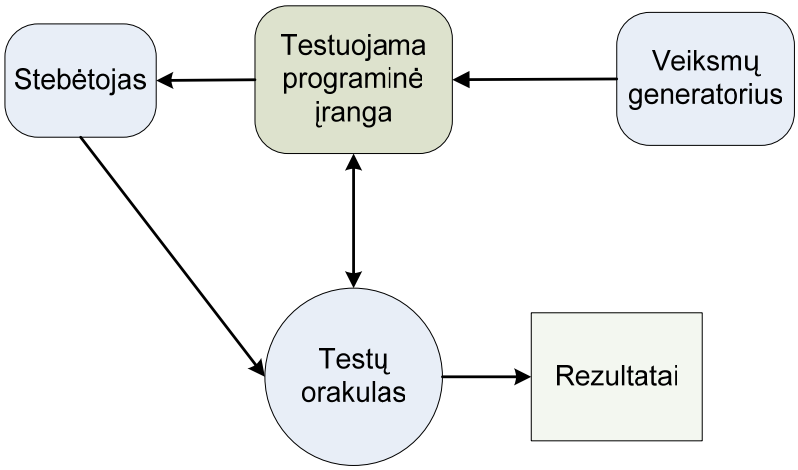
Programinės įrangos testavimo procese vienas iš svarbiausių etapų yra testo rezultatų apdorojimas ir nustatymas ar testas pateikė laukiamus rezultatus. Už šiuos etapus testavimo procese yra atsakingas testų orakulas. Testų orakulo vaidmenį gali atlikti ir žmogus (testų inžinierius), tačiau kuomet testinių atvejų yra daug tai pareikalauja pernelyg daug laiko ir testavimas jau negali būti laikomas automatizuotu. Tad rezultatų vertinimą dažniausiai atlieka specializuota programa arba komponentas, kuris lygina gautus testo rezultatus su laukiamaisiais ir išsaugo rezultatus kartu su savo priimtu verdiktu.

Paveikslėlyje 4.1 pateikta ICARE komponentų testavimo karkaso principinė schema. Karkasas sudarytas iš šių elementų:

- Veiksmų generatorius.

Komponentas imituojantis įvairių tipų sąsajas. Siunčia imituojamų sąsajų veiksmus testuojamajai programinei įrangai.

- Stebėtojas.
  Komponentas gaunantis testo rezultatus.
- Orakulas.
  Palygina gautus rezultatus su laukiamaisiais ir juos išsaugo kartu su savo priimtu verdiktu.



4.1 pav. Testavimo karkaso principinė schema

Ruošiant testus pirmiausia yra svarbu nuspręsti kurios sistemos savybės bus tikrinamos ir paruošti veiksmų generatorių atitinkamų įvedimo duomenų generavimui. Pavyzdžiui, vienas iš testų galėtų tikrinti ar korektiškai veikia sąsajų suliejimo mechanizmas vykdant vienas kitą papildančius veiksmus. Tokiu atveju turėtų būti patikrinta ar tenkinamos šios sąlygos:

- Jei vienas kitą papildančių veiksmų laiko žymės yra pernelyg dideliame atstume vienas nuo kito (netelpa į laikinąjį langą), tada veiksmas neturėtų būti įvykdytas;
- Jei vienas kitą papildančių veiksmų žymės telpa į laikinąjį langą, tada veiksmas turėtų būti įvykdytas;

Testinių atvejų generavimui reikėtų sukurt specializuotą įrankį arba pasirinkti vieną iš jau egzistuojančių. Vienas iš tokių yra Tobias [14], kuris pagal apibrėžtą scenarijų gali sugeneruot didelius kiekius testinių atvejų.

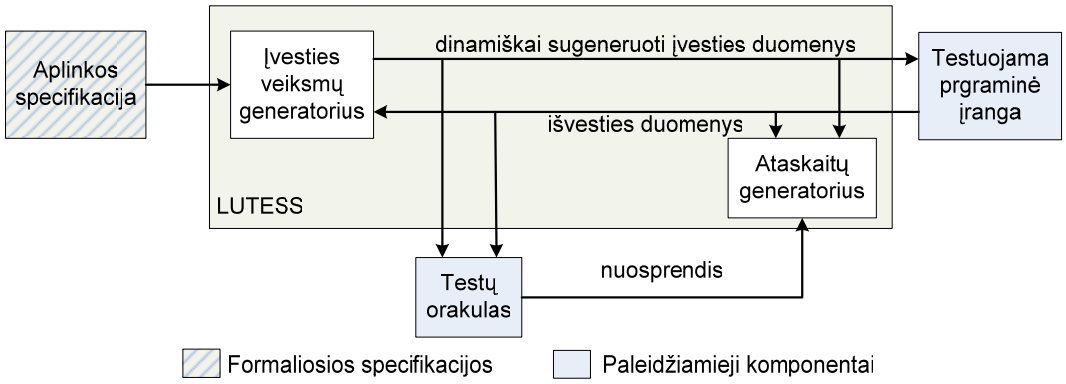
Kadangi testiniai atvejai yra automatiškai generuojami dideliais kiekiais, viena iš sudėtingiausių užduočių testuojant kombinuotas vartotojo sąsajas yra specifikuoti kriterijus

pagal kuriuos testų orakulas galėtų nuspręsti apie testo sėkmę arba nesėkmę. Tam kode gali būti panaudojami įvairūs objektų savybių aprašai. Pavyzdžiui, C# kalboje parašytoms sistemoms gali būti panaudotas spec# [2, 1] praplėtimas.

4.1.2. Lutess testavimo aplinka

Lutess [28, 10] yra testavimo įrankis skirtas sinchroninės programinės įrangos funkcijoms testuoti. Šio įrankio pagalba yra generuojamos testuojamosios programinės įrangos įvesties sekos atsižvelgiant į iš anksto apibrėžtas formaliąsias specifikacijas. Formaliomis specifikacijomis yra aprašoma galima testuojamos programos aplinkos elgsena. Lutess generuoja testavimo atvejus, valdo testų orakulo bei testuojamojo komponento vykdymą, išsaugo įvedimo bei išvedimo sekų reikšmes kartu su testų orakulo nuosprendžiu.

Šis testavimo įrankis tinka tik sinchroninei programinei įrangai testuoti, o formalios specifikacijos yra aprašytos Lustre sinchronine programavimo kalba. Lustre yra programavimo kalba skirta realaus laiko sistemoms specifikuoti [12].



4.2 pav. Programinės įrangos testavimas su Lutess

Lutess testavimo įrankiui reikia aplinkos aprašo ir testuojamo komponento orakulo kuris nusprendžia ar testas pavyko (4.2 pav.). Testas yra įvykdomas per vieną veiksmo ir atoveiksmio ciklą. Įvesties duomenų generatorius atsitiktiniu būdu pagal apibrėžtas aplinkos specifikacijas parenka įvesties reikšmes ir perduoda jas testuojamajam komponentui, kuris atsako išvesties reikšmėmis, o pastarosios perduodamos atgal generatoriui. Tada generatorius vėl generuoja naujas įvesties reikšmes ir ciklas kartojasi. Orakulas apdoroja kiekvieno ciklo įvesties bei išvesties reikšmes ir nustato ar nebuvo nukrypta nuo programinės įrangos formaliomis specifikacijomis apibrėžtų savybių. Įvesties duomenų generatorių sukuria Lutess pagal apibrėžtas formaliąsias specifikacijas, o orakulas ir testuojamas komponentas yra atskirai vykdomos programos.

Nors Lutess yra testavimo aplinka skirta sinchroninei programinei įrangai testuoti, tačiau ją galima panaudoti ir dialoginių sistemų testavimui, nes kombinuotą dialoginę sistemą taip pat galima laikyti sinchronine programine įranga, jei yra fiksuojami visi vartotojo veiksmai bei kiti išoriniai veiksniai. Tai reiškia, kad testo metu įvesties duomenys bus išduoti tik tada, kai testuojamas komponentas yra pasiruošęs juos priimti. Šios sąlygos įvykdymu pasirūpina pati Lutess testavimo aplinka. Taigi tuo atveju kai testuojama kombinuota vartotojo sąsaja, testo tikslas yra patikrinti ar vartotojo veiksmų seka (pateikiama kaip loginiai vienetai) yra įvykdoma ir yra gaunama atitinkama rezultatų seka.

Dažnai kombinuotose sąsajose veiksmai yra atliekami papildomuoju arba pertekliniu būdu (žiūrėti skyrių 2.4.1), kuomet vartotojo veiksmai yra kombinuojami bent iš dviejų tipų sąsajų laikinajame lange [20]. Lutess suteikia galimybę testuoti ir įvertinti tokius kombinuotus veiksmus.

Vartotojo elgsena su sąsaja dažniausiai nėra visiškai atsitiktinė ir yra nukreipta į tam tikros užduoties atlikimą. Formaliomis specifikacijomis galima apibrėžti ne tik visiškai atsitiktinę vartotojo elgseną, bet daug realistiškesnę elgseną apibrėžiant įvairias elgsenos tikimybes arba nurodant konkrečius scenarijus kuriuos reikia įvykdyti.

4.1.2.1. Pagrindinės problemos

Lutess įrankyje nėra galimybės imituoti kombinuotos vartotojo sąsajos vykdomus veiksmus. Lutess sugeba tik generuoti ir priimti logines įvesties reikšmių sekas. Todėl norint susieti Lutess su testuojamu komponentu yra būtina tarp jų įvesti papildomą abstrakcijos lygį, kuriame Lutess loginių reikšmių sekos būtų verčiamos į tikrąsias vartotojo veiksmų sekas ir atvirkščiai.

Abstrakcijos lygį įvesties reikšmių sekos konvertavimui galima realizuoti vienu iš dviejų būdų:

- Sekos verčiamos į techninės įrangos siunčiamus signalus (pavyzdžiui, pelės ar klaviatūros klavišų paspaudimus).
    - Privalumas – nereikalingas testuojamo komponento išeities kodas.
    - Trūkumas – techninės įrangos siunčiamų signalų provokavimas gali būti sudėtingas ir reikalaujantis techninės įrangos žinių.
- Sekos verčiamos į testuojamos programinės įrangos veiksmus (angl. k. „*events*“).
    - Privalumas – gana nesudėtingas realizavimas.

    - Trūkumas – abstrakcijos lygio realizavimas gali pareikalauti testuojamo komponento perkompiliavimo.

Abstrakcijos lygyje loginės išvesties reikšmių (programos būsenos) sekas galima gauti:

- Analizuojant testuojamo komponento objektų savybių ar kintamųjų reikšmes.
    - Privalumas – nesudėtinga realizacija.
    - Trūkumas – norint gauti testuojamo komponento objektų savybių ir kintamųjų reikšmes, tam gali tekti specialiai modifikuoti testuojamą komponentą.
- Naudojant specializuotus dialogų analizatorius.
    - Privalumas – nebūtina žinoti testuojamo komponento išeities kodo struktūros.
    - Trūkumas – specializuotas analizatorius gali papildomai kainuoti ir gali pareikalauti papildomų žinių kaip juo naudotis.

Taigi kombinuotos vartotojo sąsajos testavimui su Lutess reikia:

- Testuojamo komponento su paruošta sąsaja per kurią būtu galima pateikti logines įvesties reikšmių sekas ir kuri grąžintų logines išvesties (rezultatų) reikšmių sekas.
- Testų orakulo apibrėžiančio testuojamo komponento savybes, kurias komponentas privalo atitikti.
- Lustre aplinkos elgsenos formaliųjų specifikacijų, kurios apibrėžia galimą vartotojo elgseną su testuojamu komponentu t.y. fizikai įmanomus atlikti veiksmus ir/arba veiksmų tikimybes įvairiomis sąlygomis.
- Įrankis sukurtas tik Linux Fedora operacinei sistemai, tad ir testuojamasis komponentas privalo veikti šioje operacinėje sistemoje. Šį apribojimą galima apeiti realizavus nuotolinio testavimo karkasą, kuomet Lutess ir testuojamasis komponentas veikia skirtingose operacinėse sistemose.

4.2. Metodų palyginimo kriterijai

Tinkamiausio automatizuoto testavimo metodo pasirinkimui naudojami šie metodų palyginimo kriterijai:

- Testų vykdymo sparta
  Kadangi vykdomi labai dideli kiekiai testinių atvejų, bendras sistemos testavimo laikas gali tiesiogiai priklausyti nuo testinio atvejo įvykdymo laiko.
- Realizacijos sudėtingumas
  Realizacijos sudėtingumas leidžia įvertinti kaip greitai galima paruošti sistemą testavimui.
- Rezultatų pateikimas
  Įvertinama rezultatų pateikimo formos patogumas.
- Nepriklausomumas nuo sistemos realizavimo kalbos
  Kriterijus leidžia įvertinti ar metodas gali būti panaudotas testuojant bet kokia kalba realizuotą sistemą.

Lentelėje 4.1 pateikiamos metodo įvertinimo metrikos ir jų galimos reikšmės bei skaitiniai kiekvienos reikšmės įverčiai.

4.1 lentelė. Metodų vertinimo metrikos, reikšmės ir įverčiai

Metrika	Galimi įvertinimai	Įvertinimas balais
Testų vykdymo sparta	Lėta	0
	Vidutinė	1
	Greita	2
Realizacijos sudėtingumas	Sudėtingas	0
	Vidutinis	1
	Nesudėtinga	2
Rezultatų pateikimas	Nepatogus	0
	Patogus	1
Nepriklausomumas nuo sistemos realizavimo kalbos	Ne	0
	Taip	1

4.3. Metodų palyginimas

Pagal pateiktus kriterijus įvertinami tiriamieji metodai. Įvertinimai pateikiami lentelėje 4.2.

4.2 lentelė. Metodų palyginimas

Metodas / Kriterijus	Lutess testavimo aplinka		Naudojant spec# ir Tobias	
	Įvertinimas	Balai	Įvertinimas	Balai
Testų vykdymo sparta	Vidutinė	1	Vidutinė	1
Realizacijos sudėtingumas	Sudėtingas	0	Nesudėtingas	2
Rezultatų pateikimas	Nepatogus	0	Nėra	0
Nepriklausomumas nuo sistemos realizavimo kalbos	Taip	1	Ne	0
Bendras įvertinimas balais	-	2	-	3

Taigi įvertinus metodus pagal kriterijus matome, kad kombinuotų sąsajų testavimui verta pasirinkti testavimo metodą naudojant spec# ir Tobias technologijas, nes naudojant šį metodą galima gana greitai sukurti testavimo karkasą ir paruošti sistemą testavimui. Tiesa, šis metodas tinkamas tik C# kalba parašytoms sistemoms. Kita kalba parašytoms sistemoms reikia ieškoti alternatyvos spec# specifikacijoms.

Lutess testavimo aplinką verta pasirinkti tada, kai sistema yra realizuota kita nei C# kalba. Naudojant šį metodą, prireiks žinių Lustre specifikavimo kalboje, bei paruošti papildomus reikalingus testavimo komponentus kurie apjungtų testuojamą programinę įrangą su Lutess įrankiu.

Kadangi KTAS projektas realizuotas C# kalboje, o Lutess įrankis veikia tik Linux Fedora operacinėje sistemoje, eksperimentinei daliai buvo pasirinktas patogesnis automatizuotas kombinuotų vartotojo sąsajų testavimo metodas, kuriame naudojamos spec# ir Tobias technologijos.

5. EKSPERIMENTINĖ DALIS

5.1. Tikslas

Norėdami eksperimentiškai pagrįsti tiriamojoje dalyje teoriškai aprašytą komponentiškai realizuotos kombinuotos sąsajos testavimo metodą, realizavome bandomąjį modalumų suliejimo mechanizmą ICARE architektūroje, testavimo karkasą ir atlikom eksperimentinį modalumų suliejimo mechanizmo automatizuotą testavimą. Eksperimento tikslas – nustatyti pasirinko testavimo metodo efektyvumą testuojant ICARE architektūroje realizuotus modalumų suliejimo mechanizmus bei nustatyti šio testavimo metodo privalumus ir trūkumus.

5.2. Realizacija

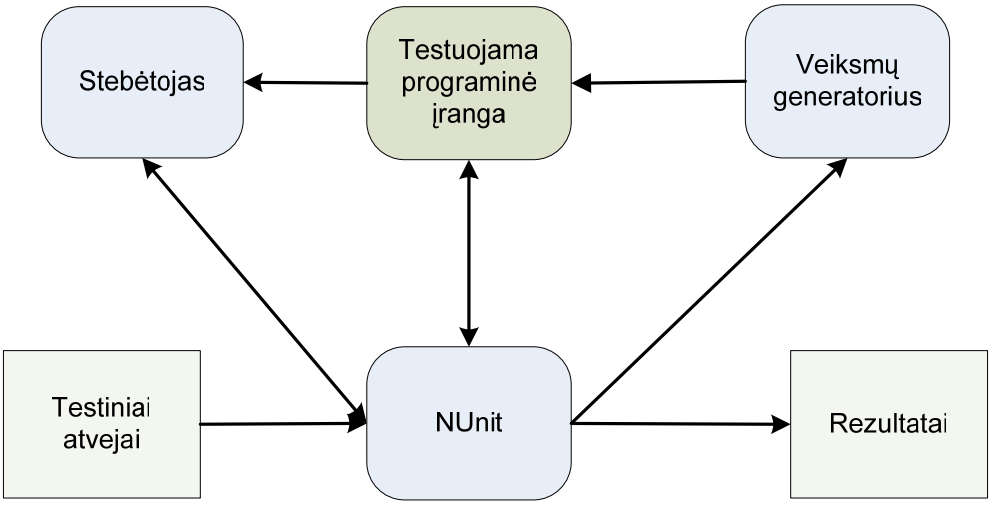
Testavimo procese naudojami šie komponentai:

- Testinių atvejų generatorius Tobias.
- NUnit komponentas, kuris valdys testų paleidimą, vykdymą ir rezultatų apdorojimą.
- Spec# kalbos specifikacijos, kurias naudojant bus nustatomas testo rezultatas.

Testiniams atvejams generuoti naudojamas patogus kombinatorinis duomenų generavimo įrankis – Tobias, kuris pagal apibrėžtą scenarijų gali sugeneruoti didelius kiekius testinių atvejų. Sugeneruoti testiniai atvejai yra tekstinio pavidalo. Jie yra išsaugomi rinkmenoje, iš kurios testavimo metu yra nuskaitomi.

Spec# kalba yra C# kalbos praplėtimas. Spec# specifikacijomis yra apibrėžiamos testuojamų objektų savybių reikšmės, metoduose nustatomos savybių išankstinės sąlygos. Jeigu testo vykdymo metu bus pažeistos aprašytos spec# kalbos specifikacijos bus iššauktas prieštaravimas ir tai reikš, kad testas nepavyko. Taigi šių specifikacijų dėka gaunamas automatizuotas testų orakulas.

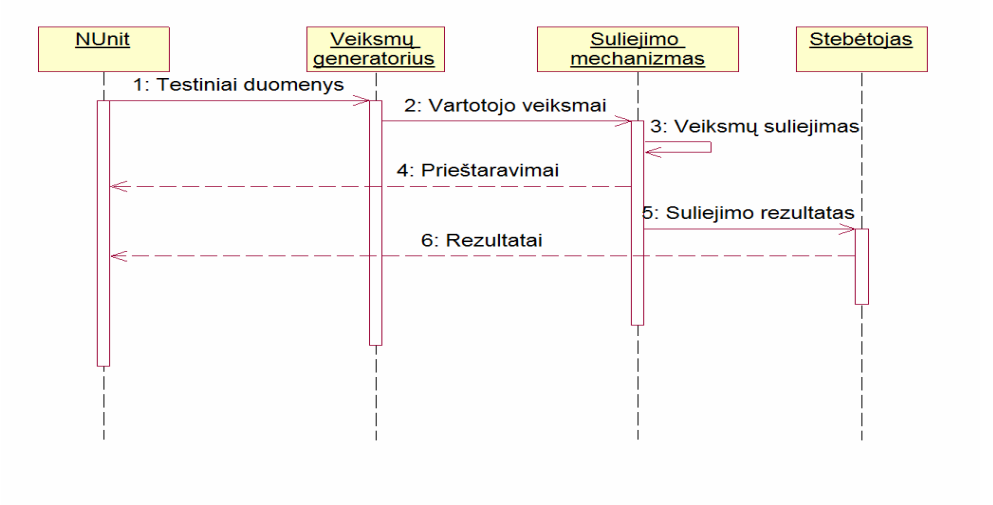
Realizuoto testavimo karkaso schema pateikta paveikslėlyje 5.1.



5.1 pav. Testavimo karkasas

Veiksmų generatorius yra klasė, kuri pagal Tobias sugeneruotus testinių atvejų duomenis, imituoja įvairių modalumų atliekamus veiksmus ir perduoda juos testuojamiem ICARE komponentam.

Stebėtojas yra klasė, kuri priima suliejimo mechanizmo sugeneruotą galutinį veiksmą kurį turi įvykdyti sistema.



5.2 pav. Testo vykdymas

5.3. Testavimo strategija

Komponentams testuoti buvo sukurti 4 testavimo duomenų tipai, kuriuose buvo beveik 1000 testinių atvejų.

Taikant „juodos dėžės“ testavimo metodą testinių atvejų generavimas buvo atliekamas laikantis šių strategijų:

- Testiniai atvejai tikrina sistemos elgseną įprastomis veikimo sąlygomis. Testinius atvejus sudarė veiksmai iš įvairių sąsajų tipų su įvairiomis laiko žymėmis. Buvo parenkami įvairūs užlaikymo laikai tarp veiksmų.
- Ribiniai testiniai atvejai testuojantys komponentų elgseną neįprastomis veikimo sąlygomis – veiksmai iš sąsajų siunčiami vienu metu dideliais kiekiais.

Taikant „baltos dėžės“ metodą testiniai atvejai parenkami atsižvelgiant į testuojamų komponentų realizacijos specifiką.

5.4. Rezultatai

Testų rezultatai pateikti 5.1 lentelėje. Testavimo metu aptiktas klaidas galima suskirstyti į du pagrindinius tipus:

- programavimo klaidos,
- specifikacijos klaidos (klaidingas spec# kodas).

Daugiausia klaidų buvo aptikta *tingaus suliejimo* strategijos testuose. Pavyzdžiui, buvo netinkamai suliejami vartotojo veiksmai įvykę viename laikinajame lange, todėl nebuvo atliktas ir laukiamas veiksmas. Taip pat aptikta nemažai klaidų apskaičiuojant laikinojo lango ribas: kai kuriais atvejais laikinojo lango pabaigoje veiksmai nebuvo įvykdomi.

5.1 lentelė. Komponentų testų rezultatai

Testas	Testinių atvejų kiekis	Aptiktų klaidų kiekis
Siunčiami 5 veiksmai naudojant ankstyvojo suliejimo strategiją	346	1
Siunčiami 5 veiksmai naudojant tingaus suliejimo strategiją	346	4
Siunčiami 5 veiksmai be pauzių	251	2
Testuojamos įvairios komponentų elgsenos atsižvelgiant į programinį kodą	12	0

Taigi eksperimento metu buvo nustatyta, kad naudojant pasirinktą kombinuotų sąsajų testavimo metodą galima automatizuotu būdu aptikti klaidas kombinuotose sąsajose ir tokiu būdu užtikrinti bent minimalų sistemos pasikliaujamumo lygį.

Eksperimento metu nustatyti metodo privalumai:

- Pilnas automatizuoto testavimo procesas: testinių atvejų generavimas su automatizuotu testų orakulu.
- Nesudėtingas bei greitas testavimo karkaso realizavimas.

Eksperimento metu nustatytas pagrindinis metodo trūkumas:

- Pakankamai sudėtingas programinio kodo specifikavimas spec# kalbos aprašais. Norint gauti pakankamai gerą automatizuotą testų orakulą reikia tiksliai ir išsamiai aprašyti visas objektų savybes ir elgseną.

6. IŠVADOS

1. Kombinuotosios vartotojo sąsajos projektavimas yra pakankamai naujas ir sudėtingas procesas, reikalaujantis nemažai šios srities praktinių ir teorinių žinių.
2. Kombinuotų sąsajų kūrime vis dar jaučiamas metodologijų ir įrankių trūkumas ne tik projektavimo etape, bet ir testavime.
3. Išanalizuotos kombinuotų vartojo sąsajų projektavimo ir testavimo metodologijos, jų privalumai ir trūkumai.
4. Pasiūlytas ir eksperimentiškai pagrįstas automatizuotas kombinuotų sąsajų testavimo metodas, leidžiantis pakankamai greitai ir efektyviai aptikti klaidas kelių įvedimo būdų suliejimo mechanizme.
5. Projekto metu sukurtos šnekos atpažinimo priemonės palengvina valdymo balsu įdiegimą programinėje įrangoje.

7. LITERATŪRA

1. Barnett, M., DeLine, R., Fähndrich, M., Rustan, K., Leino, M., Schulte, W. Verification of object-oriented programs with invariants. JOT 3(6), 2004.
2. Barnett, M., Rustan, K., Leino, M., Schulte, W. The Spec# programming system: An overview. Springer, 2004.
3. Bolt, R. A. "Put-that-there": Voice and gesture at the graphics interface. 1980, p. 262–270.
4. Bouchet, J., Nigay, L., Ganille, T. ICARE Software Components for Rapidly Developing Multimodal Interfaces. France, 2004.
5. Bourguet, M. L., A Toolkit for Creating and Testing Multimodal Interface Designs. London.
6. Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. QuickSet: Multimodal interaction for distributed applications. *Proceedings of the Fifth ACM International Multmedia Conference: tarptautinės konferencijos pranešimų medžiaga*. New York, 1997, p. 1325–1328.
7. Cohen, P. R., Oviatt, S. L. The role of voice input for human-machine communication.
8. Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R. M. Four easy pieces for assessing the usbility of multimodal interaction: the CARE properties. Lillehammer, 1995.
9. Dowell, J., Shmueli, Y., Salter, I. Applying a cognitive model of the user to the design of a multimodal speech interface. 1996.
10. du Bousquet, L., Ouabdesselam, F., Richier, J.-L., Zuanon, N. Lutess: a Specification Driven Testing Environment for Synchronous Software. *ICSE'99*. ACM Press, 1999, p. 267-276.
11. Flippo, F., Krebs, A., Marsic, I. A Framework for Rapid Development of Multimodal Interfaces. 2003, p. 109–116.
12. Halbwachs, N. Synchronous programming of reactive systems, a tutorial and commented bibliography. *CAV'98, LNCS 1427*. Springer Verlag, 1998.
13. Henry, T.R., Hudson, S.E., Newell, G.L. Integrating Gesture and Snapping into a User Interface Toolkit, in Proc. *Symposium on User Interface Software and Technology*. ACM Press, 1990, p. 112-121.

14. Ledru, Y., du Bousquet, L., Maury, O., Bontron, P. Filtering TOBIAS combinatorial test suites. *Fundamental Approaches to Software Engineering (FASE'04)*, Barcelona, Spain, 2004.
15. Little, T.D.C, Ghafoor, A., Chen, C.Y.R., Chang, C.S., Berra, P.B. Multimedia Synchronization. *IEEE Data Engineering Bulletin*. 1991, p. 26-35.
16. Nardelli, L., Orlandi, M., and Falavigna, D. A Multi-Modal Architecture for Cellular Phones. State College, PA, USA, 2004, p. 323–324.
17. Nigay, L. 1994 Conception et modélisation logicielles des systemes interactifs : application aux interfaces multimodales. *Daktaro disertacija*. Grenoble universitetas, 1994.
18. Nigay, L. A Case Study of Software Architecture for Multimodal Interactive System: a voice-enabled graphic notebook, 1991.
19. Nigay, L., Coutaz, J. A Design Space For Multimodal Systems: Concurrent Processing and Data Fusion. *INTERCHI'93*. New York, 1993, p. 172-178. 1993.
20. Nigay, L., Coutaz, J. A Generic Platform for Addressing the Multimodal Challenge. 1995, p. 98–105.
21. Nigay, L., Coutaz, J. Multifeature Systems: The CARE Properties and Their Impact on Software Design. France, 1997.
22. Oviatt, S. L. Multimodal interactive maps: Designing for human performance. Human-Computer Interaction (special issue on Multimodal Interfaces), 1997.
23. Oviatt, S. L., Cohen, P. R. Discourse structure and performance efficiency in interactive and noninteractive spoken modalities. Computer Speech and Language, 1991.
24. Oviatt, S. L., Olsen, E. Integration themes in multimodal human-computer interaction. *International Conference on Spoken Language Processing: tarptautinės konferencijos pranešimų medžiaga*, 1994, p. 551-554.
25. Oviatt, S., Cohen, P., Wu, L., Vergo, J., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J., Ferro, D. Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions, 2000.
26. Parissis, I., Ouabdesselam, F. Specification-based Testing of Synchronous Software. *ACM SIGSOFT Fourth Symposium on the Foundations of Software Engineering*. ACM Press, 1996.

27. Cohen, P. R., Oviatt, S. L. The role of voice input for human-machine communication. *Proceedings of the National Academy of Sciences*. Washington, D. C., 1995.
28. Siroux, J., Guyomard, M., Multon, F., Remondeau C. Oral and gestural activities of the users in the GEORAL system. 1996.
29. Suhm, B. Multimodal interactive error recovery for non-conversational speech user interfaces. Fredericiana University, Germany, 1998.

8. TERMINŲ IR SANTRUMPŲ ŽODYNAS

CARE – (*Complementary, Assignment, Redundancy, Equivalence*) papildymas, paskyrimas, pertekliškumas ir lygiavertiškumas – savybės, kurios naudojamos kombinuotosioms vartotojo sąsajoms specifikuoti.

DBVS – duomenų bazės valdymo sistema.

ICARE – (*Interaction CARE*) komponentinis kombinuotų vartotojo sąsajų projektavimo metodas.

Lustre – programavimo kalba, skirta tikralaikėms sistemoms specifikuoti.

Lutess – testavimo aplinka, skirta sinchroninės programinės įrangos funkcijoms testuoti.

MSDE – (*Microsoft SQL Server Desktop Engine*) Microsft SQL duomenų bazės valdymo sistemos supaprastinta, nemokama versija.

RUP – (*Rational unified process*) kompanijos Rational sukurtas unifikuotas procesas.

SAPI (*Speech Application Programming Interface*) – kalbos aplikacijų programavimo sąsaja.

TCP/IP – (*Transmission Control Protocol/Internet Protocol*) susiekimo protokolų rinkinys.

TDS – (*Tabular Data Stream*) lentelės duomenų srautas.

UML – (*Unified Modeling language*) unifikuota modeliavimo kalba.