ŠIAULIŲ UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

Dainius Figoras

Informatikos magistro studijų II kurso studentas

Programų integravimas Windows operacinėje sistemoje

Magistro darbas

Darbo vadovas:

prof. L. Sakalauskas

Recenzentė:

Lekt. L. Tankelevičienė

Šiauliai, 2007

TURINYS

Įvadas

Tema

Programų integravimas Windows operacinėje sistemoje

Aktualumas

Socialinis. Programinės įrangos kūrimas yra labai sudėtingas procesas susidedantis iš įvairių etapų.

Praktinis. Pats kūrimo procesas trunka pakankamai ilgai (priklausomai nuo kuriamos programinės įrangos gali svyruoti nuo dienų iki metų), todėl jo pagreitinimas turėtų praktinės naudos tiek kūrėjams, kurie galėtu greičiau pateikti programinę įrangą, tiek ir vartotojams, kurie galėtu greičiau pradėti naudotis naujos programinės įrangos teikiamomis galimybėmis.

Konkretaus tyrimo. Šis tyrimas nagrinės įtaką programinės įrangos projektavimui panaudojant operacinės sistemos Windows komponentus.

Tyrimo tikslas

Windows sąsajos komponentų panaudojimas programų gyvavimo ciklo etapuose.

Integruotos programinės įrangos projektavimo pagreitinimas panaudojant Windows operacinės sistemos komponentus.

Problema

Sudėtingėjant aplinkai, kuriai yra kuriama programinė įranga, sudėtingėja ir pati programinė įranga. Kuo programinė įranga sudėtingesnė, tuo sudėtingesnis ir jos sukūrimas, o didžiausią įtaką programinės įrangos kūrimui turi jos suprojektavimas, nes tinkamai suprojektuotą programą lengviau parašyti, testuoti, taisyti klaidas ir platinti. Todėl programinės įrangos projektavimą sunku pervertinti.

Hipotezė

Daroma prielaida, kad integruotų Windows operacinės sistemos komponentų panaudojimas padidins programų projektavimo greitį, bei patikimumą, tai leis greičiau atlikti sistemos testavimą, bei padidins viso programinio produkto patikimumą.

Tikslas, uždaviniai

Apibendrinti programų sistemų kūrimo etapus, išskiriant bendrus etapus būdingus daugeliui projektavimo metodikų.

Sudaryti pagrindinių Windows sąsajų komponentų funkcijų aprašymus

Išsiaiškinti Windows sąsajų komponentų įtaką kiekvienam kūrimo etapui.

Sudaryti bendrą metodą, kuriuo remiantis būtų galima kurti integruotas programas su integruotais komponentais.

Tyrimo metodai

Eksperimentas – bus kuriamas programos projektas pagal paruoštą metodą ir tikrinama, ar gautas rezultatas atitinka iškeltą hipotezę.

Laukiamas rezultatas

Tikimasi, kad bus sukurtas metodas, pagreitinantis tam tikros rūšies programų kūrimą.

1. Programinės įrangos gyvavimo ciklai

1.1. Programinės įrangos gyvavimo ciklų etapai

Pagrindiniai programinės įrangos gyvavimo ciklai ir jų etapai:

Klasikinis (krioklio) programinės įrangos gyvavimo modelis.

Tai pirmas programinės įrangos (PĮ) kūrimo gyvavimo ciklo modelis, sukurtas remiantis kitų inžinerinių procesų pagrindu. Šiame modelyje aprašyti visi reikalingi programinės įrangos kūrimo etapai.

1. Reikalavimų specifikavimas – bendraujant su užsakovu aprašoma funkcinė specifikacija, apribojimai ir programinės įrangos tikslai.
2. Projektavimas – sukuriama PĮ architektūra, aprašomi komponentai ir jų tarpusavio sąryšiai.
3. Kodavimas ir modulių testavimas – suprojektuoti komponentai realizuojami kaip atskiros programos, arba programos moduliai.
4. Surinkimas ir sistemos testavimas – atskiri moduliai integruojami ir testuojami kaip bendra sistema. Tikrinama kaip PĮ atitinka savo funkcinę specifikaciją.
5. Eksploatacija ir palaikymas – dažniausiai tai ilgiausia PĮ gyvavimo dalis, kurioje sistema instaliuojama pas užsakovą, atliekamas klaidų, kurios nebuvo aptiktos ankščiau, taisymas, jos eksploatavimas ir funkcinės specifikacijos ruošimas tolesnei PĮ raidai.

Projektavimą galima išskaidyti į kelias dalis – eskizinis ir detalusis projektavimas [2].

Eskizinį projektavimą sudaro architektūros sukūrimas ir bendras komponentų aprašymas, detaliajame projektavimo etape aprašyti komponentai aprašomi tiksliai.

Evoliucinis programinės įrangos gyvavimo modelis.

Evoliucinio gyvavimo ciklo pagrindas sukūrimas pradinės PĮ versijos, kuri pateikiama vartotojams, vėliau pradinė versija papildoma atsižvelgiant į vartotojų atsiliepimus ir pageidavimus.

Evoliuciniame gyvavimo modelyje visi etapai gali būti kartojami daugelį kartų. Šis modelis neneigia klasikinio (krioklio) modelio, tačiau papildo jį iteratyvumu.

1. Eskizinis aprašymas – bendras sistemos aprašymas bendraujant su užsakovu arba būsimais programinės įrangos naudotojais.
2. Reikalavimų specifikacijos – einamojo ciklo reikalavimų aprašymas.
3. Atlikimas – reikalavimų įgyvendinimas.
4. Atestacija – patobulintos sistemos atestacija.
5. Pradinė versija – pradinė PĮ versija, kuri pateikiama vartotojams ir tolesniuose etapuose tobulinama.
6. Tarpinės versijos – tarpinės PĮ versijos, kurios tobulinamos sekančiuose cikluose.

7. Galutinė versija – galutinė PĮ versija, kuri pateikiama užsakovui.

Programinės įrangos kūrimas ankščiau sukurtų komponentų pagrindų.

Šiame programinės įrangos kūrimo modelyje stengiamasi išnaudoti ankščiau sukurtus komponentus.

1. Reikalavimų specifikacija – bendraujant su užsakovu sudaroma sistemos funkcinė specifikacija.
2. Komponentų analizė – analizuojami komponentai, tinkantys ankščiau aprašytai sistemai.
3. Reikalavimų modifikacija – pagal išrinktų komponentų specifikacijas modifikuojama sistemos specifikacija, kuri suderinama su užsakovu.
4. Sistemos projektavimas – projektuojami komponentų tarpusavio ryšiai.
5. Programavimas ir surinkimas – komponentai surenkami į vientisą sistemą, pagal prieš tai suprojektuotą modelį.
6. Atestacija – sistema atestuojama ir atiduodama užsakovui.

Programinės įrangos kūrimas žingsniais.

Šiame programinės įrangos kūrimo modelyje PĮ kuriama nedideliais etapais – žingsniais. Kiekviename žingsnyje įgyvendinant dalį visos sistemos reikalavimų.

1. Reikalavimų planas – sudaromas reikalavimų planas.
2. Pažingsninis reikalavimų detalizavimas – reikalavimų detalizavimas pagal prieš tai sudarytą planą.
3. Sistemos architektūros sukūrimas – bendras sistemos architektūros sukūrimas, pagal kurią bus kuriama PĮ.
4. Sistemos kūrimo žingsnis – einamo žingsnio realizavimas.
5. Atestacijos žingsnis – einamojo žingsnio atestacija.
6. Surinkimo žingsnis – einamojo žingsnio kuriamos sistemos surinkimas.
7. Sistemos atestacija – visos sistemos atestacija, jei sistema praeina sistemos atestaciją, ji atiduodama užsakovui, kitu atveju vykdomas sekantis sistemos kūrimo žingsnis.

Spiralinis programinės įrangos kūrimo modelis.

Spiraliniame PĮ kūrimo modelyje sistema kuriama pilnai iteratyviai, kiekviena iteracija suskirstoma į keturias dalis, kurios pereinamos visose iteracijose.

1. Tikslų nustatymas – einamojo ciklo reikalavimų analizė, apribojimų aprašymas.
2. Rizikų analizė – alternatyvų įvertinimas, galimų rizikų įvertinimas.
3. Kūrimas ir testavimas – einamojo ciklo sistemos kūrimas.
4. Planavimas – sekančio žingsnio planavimas

Universalus programinės įrangos kūrimo procesas.

Panašus į spiralinį PĮ kūrimo modelį, tačiau turi tiksliai aprašytus atlikimo žingsnius, tai pat išskirtinė šio modelio savybė – UML įrankių naudojimas, visame kūrimo procese.

1. Reikalavimų specifikavimas – dalykinės srities, techninių priemonių ir reikalavimų projektui aprašymas.
2. Analizė – modeliavimas panaudojant prieš tai surinktus reikalavimus.
3. Projektavimas – architektūros sukūrimas pagal objektinį modelį.
4. Realizacija – kūrimas suprojektuotos programinės įrangos (ankstyvuose etapuose gali būti prototipų realizacija).
5. Testavimas – įvykdyto žingsnio testavimas.

MSF (Microsoft Solutions Framework) programinės įrangos kūrimo modelis.

MSF modelis turi tris pagrindines savybes, išskiriančias jį iš kitų PĮ kūrimo modelių: padalinimas į fazes, darbų kontroliavimas, įteratyvumas. MSF modelyje (kaip ir universaliame PĮ kūrimo modelyje) kiekvienoje fazėje atliekami numatyti darbai. MSF apjungia stipriąsias klasikinio gyvavimo ciklo ir spiralinio modelio puses.

1. Analizė – šios fazės tikslas vieningas PĮ koncepcijos sudarymas visiems projekto dalyviams.
2. Planavimas – PĮ architektūros sukūrimas remiantis prieš tai aprašyta koncepcija, loginiai ir fiziniai projekto modeliai.
3. Kūrimas – programinės įrangos kūrimas pagal prieš tai sudarytus projektavimo modelius.
4. Stabilizacija – PĮ testavimas, klaidų taisymas, PĮ pridavimas užsakovui.

1.2. Programinės įrangos gyvavimo ciklų etapų apibendrinimas

Išnagrinėjus visus apžvelgtus programinės įrangos kūrimo modelius, galima išskirti bendrus etapus:

1. Reikalavimų specifikavimas – sudaromas bendras sistemos vaizdas, funkcinė produkto specifikacija.
2. Analizė – analizuojami prieš tai gauti duomenys juos sistematizuojant.
3. Projektavimas – projektuojama sistema.
4. Programavimas (kodavimas) – pagal sudarytą projektinį modelį, kuriamas programinės įrangos kodas.
5. Testavimas ir sistemos atestacija – sukurta sistema testuojama ir priduodama užsakovui.

Dabar dažniausiai naudojami iteracinio pobūdžio programinės įrangos gyvavimo ciklai, kuriuose šie etapai kartojami kiekvienoje iteracijoje, analizuojant, projektuojant, koduojant ir testuojant kiekvieną iteracijos ciklą skirtą kažkuriam programinės įrangos funkcionalumui

(kiekvienas gyvavimo ciklas apibrėžia konkrečią paskirtį iteracijoms ir etapams ar jų eiliškumui). Kuriant programą pagal krioklio gyvavimo ciklo modelį visi šie etapai atliekami nuosekliai.

Vartotojo sąsajos kūrimas nėra išskirtas į atskirą iteraciją ar etapą, tačiau yra vienas svarbiausių elementų, nes būtent pagal vartotojo sąsają vartotojai sprendžia apie programinę įrangą.

Reikalavimų specifikavimo etape suformuojami reikalavimai kuriamai programinei įrangai.

Analizės etape analizuojama dalykinė sritis, vartotojų reikalavimai, funkcionalumas. Analizės etape surenkami duomenys apie kuriamą programinę įrangą, tačiau sprendimai apie jos kūrimą neatliekami. Todėl ši etapas įtakas projektavimui naudojantis Windows operacinės sistemos komponentais tiesiogiai neturi, tačiau leidžia palengvinti šį etapą pateikiant iš anksto žinomą klausimų sąrašą ir taip sumažinant riziką, kad bus surinkti ne visi reikalingi duomenys. Taip pat atliekant analizę gali būti nuspręsta ar galimas Windows komponentų panaudojimas ar ne (jei pavyzdžiui programinė įrangą skirta UNIX operacinėms sistemoms šių komponentų greičiausiai nepavyks panaudoti).

Projektavimo etape projektuojama programa, duomenų bazės vartotojo sąsaja, tinklo architektūra ir kiti reikalingi elementai, todėl tai esminis programinės įrangos kūrimo etapas, sprendimai padaryti šiame etape turės įtakos visam likusiam programinės įrangos gyvavimo ciklui (lygiai tas pats vyksta ir analizės etape, tačiau abstraktesniame lygmenyje ir darant bendresnius sprendimus). Kadangi galimų panaudoti komponentų aibė yra baigtinė ir visi jei turi pakankamai konkrečią paskirtį projektavimas supaprastėja, nes nebereikia ieškoti, ar kurti daugelio svarbių programinės įrangos dalių. Ypač daug pateikiama vartotojo sąsajos komponentų, todėl viena kritiškiausių programinės įrangos dalių supaprastėja iki reikiamų komponentų pasirinkimo ir suderinimo su kitomis dalimis.

Programavimo etape realizuojamas projektas pasinaudojant prieš tai suprojektuotos sistemos modeliu.

Testavimo ir sistemos atestacijos etape kuriami ir vykdomi programos modulių ir jų tarpusavio sąveikos testai. Tikrinama kaip sukurtoji programa atitinka savo specifikaciją. Rastos klaidos fiksuojamos ir taisomos. Kadangi operacinės sistemos komponentai jau būna ištestuoti, papildomo jų testavimo nebereikia, todėl sutaupomas laikas ir resursai.

1.3. Vartotojo sąsaja

Vartotojos sąsajos kūrimui sugaištama apie 50-70% programinės įrangos kūrimo laiko [1] taip pat nuo vartotojo sąsajos priklauso vartotojo požiūris į naudojamą programą, kuo vartotojo sąsaja jam priimtinesnė ir komfortiškesnė tuo priimtinesnis programos naudojimas.

Pagrindiniai vartotojo sąsajos kūrimo principai yra paprastumas, pastovumas [1].

Paprastumo principas leidžia vartotojui greitai susigaudyti programoje negaištant bereikalingo laiko reikalingų funkcijų paieškoms ir pateikiamų duomenų analizei, bei reikalingų duomenų paieškai.

Pastovumo principas leidžia vartotojui greitai rasti reikalingas funkcijas ar perprasti naujos programos veikimą iš savo patirties su kitomis naudotomis programomis. Jei naudojami tokie patys elementai žymintis tą pačią reikšmę, arba atitinkamos operacijos atliekamos pasinaudojant vienoda komandų seka, vartotojas dirbdamas su nepažįstama programa nejaus diskomforto.

Standartinių komponentų panaudojimas leidžia įgyvendinti visus šiuos principus, taip vartotojui pateikiant jam patogią vartotojo sąsają.

2. Windows operacinės sistemos apvalkalo komponentų apžvalga

2.1. Windows operacinės sistemos apvalkalas

Operacinės sistemos apvalkalas yra sistemos pateikiama sąsaja, kad vartotojas galėtų atlikti užduotis pasinaudodama sisteminiais komponentais. Windows operacinėje sistemoje apvalkalo dvejetainis failas vadinimas *explorer.exe*.

Apvalkalo komponentus gali išskirti į dvi dalis:

Apvalko sąsajos išplėtimo komponentus (angl. *shell extension interface*) ir apvalko vardo srities sąsajos išplėtimo komponentus (angl. *shell name space extension interface*).

Apvalkalo išplėtimo komponentai gali pakeisti arba papildyti tam tikro tipo failų elgesį, pavyzdžiui pernešant failą įvykdoma specifinė funkcija, arba kontekstiniame meniu įdedami reikalingi papildomi meniu punktai, savybių dialogo lange pateikiami papildomi duomenys ir kt.

Vardo srities išplėtimo komponentai gali būti dviejų tipų: jei vardo srities išplėtimas susietas su failo tipu, šis išplėtimas funkcionuos kaip kontekstinio meniu išplėtimas. Jei vardo srities išplėtimas susietas su katalogu, tai šiam katalogui bus galima nustatyti rodomą turinį, ikonas, pakatalogius, rūšiavimą, kontekstinius meniu ir kt.

Operacinės sistemos apvalko objektai pasiekiami iš darbalaukio, Windows operacinės sistemos naršyklės (explorer.exe), ar kitų sistemos komponentų.

Detalus operacinės sistemos apvalko komponentų ir jų funkcijų aprašymas pateikiamas pirmame priede.

3. Windows operacinės sistemos apvalkalo komponentų pritaikymas apibendrintam programų gyvavimo ciklo modeliui

Gyvavimo ciklo modelį sudaro etapai:

1. Reikalavimų specifikavimas.
2. Analizė.
3. Projektavimas.
4. Programavimas (kodavimas).
5. Testavimas ir sistemos atestacija.

3.1. Reikalavimų specifikavimas

Pirmiausia reikia atkreipti dėmesį į tai, kad praktiškai visos operacinės sistemos apvalko operacijos yra atliekamos su kokiu nors objektu. Tai palengvina bendravimą tarp programinės įrangos kūrėjų ir užsakovų, nes didėja informacijos perdavimo supratimas tarpusavyje ir programinės įrangos veikimo įvertinimą, bei klaidų taisymą, kadangi abi pusės kalba apie tuos pačius objektus.

3.2. Analizė

Analitikui belieka išskirti visus naudojamus objektus. Šioje vietoje taip pat galima atlikti tam tikrą išankstinį objektų suskirstymą remiantis Windows operacinės sistemos apvalkalo komponentais, kurie jau turi sudarytą hierarchijos sistemą, todėl praktiškai lieka tik surasti reikiamus objektus, arba išskirti jų aibes. Pagrindinę hierarchiją sudaro katalogai ir failai. Failai dažniausiai nusako bet kokį galutinį objektą arba tikslą, katalogai taip pat gali būti objektai (konteineriai), arba klasifikatoriai (pagal ką skirstomi galutiniai failai), arba objektai ir tuo pačiu atlikti klasifikaciją.

Suskirsčius objektus į klases, taip pat žinomos ir galimos jų operacijos, todėl galima patikrinti, ar surinkti visi reikiami duomenys ir papildyti reikalavimus, jei randama klaidų, arba įsitikinti, kad galimų objekto operacijų tikrai nereikia.

Kadangi klaidos reikalavimų etape labai stipriai neigiamai atsiliepia vėlesniuose etapuose papildomas reikalavimų patikrinimas yra labai naudingas.

3.3. Projektavimas

Projektuotojui gavus analizės objektų modelį reikia susieti konkrečiomis savybėmis pasižyminčius analizės objektus su operacinės sistemos apvalkalo komponentų objektais. Kadangi dalies komponentų projektuoti iš naujo nebereikia sutaupoma laiko, taip pat galima sudaryti naujų objektų dalines sąsajas, nes jie turės bendrauti su sisteminiais komponentais, toks išankstinis žinojimas taip pat pagreitina projektavimą (projektuotojas žino, kad nors reikiamo komponento jis ir neturi, tačiau jo sąsaja (angl. interface) jau žinoma.

Windows apvalkalo komponentų naudojimas pakankamai griežtai nusako vartotojo sąsaja, todėl nebereikia ieškoti atskiro sprendimo apie duomenų pateikimą ir sąveiką su vartotoju, taip dar labiau supaprastinant projektavimo etapą.

3.4. Programavimas

Programavimo etapui Windows apvalkalo komponentų naudojimas didelės įtakos neturi, išskyrus, nedidelius programavimo stiliaus pataisymus (apvalko objektų kūrimas ir naudojimas turi savas taisykles, kurių reikia laikytis, o tai gali įtakoti ir bendrą stilių).

Taip pat nebereikia programuoti pagrindinių vartotojo sąsajos elementų, užtenka priderinti jų elgesį prie reikiamo kuriamai programinei įrangai.

3.5. Testavimas ir sistemos atestacija

Testavimo metu galima netestuoti atskirų apvalko komponentų, užtenka kad jų sąveika su kitais programinės įrangos komponentais praeitu testus.

4 Programos kūrimo planas

Pagal sudarytą programos gyvavimo ciklo modelį sudaromas programos planas, kuriuo remiantis būtų galima kurti programas integruotas Windows operacinėje sistemoje.

Bandomajam projektui pasirinkta vidutinio dydžio programa skirta prekių ir pardavimų apskaitai hipotetinei įmonei.

Programos plane turi atsispindėti visi būtini gyvavimo ciklo etapai, kad jie atitiktų realios programinės įrangos kūrimo etapus ir problemas.

4.1. Reikalavimų specifikavimas

Poreikių (pagrindinių funkcijų) aprašymas

Klientų sąrašo peržiūra – matomas įrašytų klientų sąrašas, kuriame pateikiami būtiniausi kliento atributai (pavadinimas, kodas, telefonas, adresas), bei kliento piniginė ataskaita (kliento suma už prekes, kliento sumokėta suma, kliento skola = suma už prekes – sumokėta suma).

Prekių sąrašo peržiūra – matomas prekių sąrašas, kuriame pateikiamos būtiniausios prekės detalės (pavadinimas, kodas, kaina, turimas likutis).

Naujos sąskaitos išrašymas – sukuriama sąskaita klientui, kurioje įvedamos parduotos prekės. Parduotų prekių likučiai automatiškai nurašomi, taip pat padidinama kliento suma už prekes.

Naujo kvito/važtaraščio išrašymas – sukuriamas naujas kvitas iš kliento paimtai pinigų sumai. Sukūrus kvitą automatiškai padidinama kliento sumokėta suma.

Patikrinimas ir pagrindimas

Peržiūrėjus pagrindines programos funkcijas pastebima, kad funkcijų esmę sudaro įvairių sąrašų atvaizdavimas. Sąrašų atvaizdavimui galima panaudoti Windows apvalkalo komponentus, todėl šią programą įmanoma sukurti integruotą programinę įrangą.

Pastaba. Reikalavimų etape reikia nuspręsti, ar galima norimą programą sukurti kaip integruotą programinę įrangą ar ne. Jei pagrindiniuose funkciniuose reikalavimuose yra esminių trukdžių neleidžiančių panaudoti Windows apvalkalo komponentų reikia arba išskaidyti programą į keletą dalių, kurioms galima panaudoti integruotus komponentus ir išorines programas, arba kuriamai programinei įrangai pasirinkti kitą kūrimo modelį.

Panaudojimo atvejų sudarymas

Aprašomi panaudojimo atvejų pavadinimai ir jų atlikimo eiga.

Klientų sąrašo peržiūra

Atidaromas langas, kuriame parodomas sąrašas su klientų duomenim.

Klientų sąrašo lange esant neigiamai kliento skolai (skola < 0) skola užrašoma raudonu šriftu.

Kliento sukūrimas

Atidaromas klientų sąrašas.

Paspaudžiamas mygtukas sukuriantis naują kliento įrašą.

Sąraše sukuriamas naujas kliento įrašas su standartiniais duomenim, bet duomenų bazėje dar neišsaugomas.

Pasirenkamas naujai sukurtas kliento įrašas.

Atidaromas dialogo langas, kuriame įvedami kliento duomenys.

Paspaudus sukūrimo mygtuką sukuriamas naujas klientas.

Klientų sąrašas papildomas naujų klientu.

Kliento duomenų pakeitimas

Atidaromas klientų sąrašas.

Pasirenkamas kliento įrašas.

Atidaromas dialogo langas, kuriame matomi sukurto kliento duomenys.

Atliekamas duomenų pakeitimas.

Paspaudus išsaugojimo mygtuką išsaugomi nauji kliento duomenys.

Prekių sąrašo peržiūra

Atidaromas langas, kuriame matomas sąrašas su prekių duomenimis.

Prekės sukūrimas

Atidaromas dialogo langas, kuriame įvedami naujos prekės duomenys.

Paspaudus sukūrimo mygtuką sukuriama nauja prekė.

Prekių sąrašas papildomas nauja preke.

Prekės duomenų išsaugojimas
Atidaromas langas, su pasirinktos prekės duomenis.

Atliekamas prekės duomenų pakeitimas.

Paspaudus išsaugojimo mygtuką išsaugomi nauji prekės duomenys.

Sąskaitos sukūrimas
Atidaromas sąskaitų sąrašo langas.

Atidaromas langas, kuriame įvedami sąskaitos duomenys.

Paspaudus sukūrimo mygtuką sukuriama nauja sąskaita nurodytam klientui.

Sąskaitų sąrašas papildomas nauja sąskaita.

Sukūrus naują sąskaita galima redaguoti sąskaitos eilutes.

Sąskaitos duomenų pakeitimas
Atidaromas pasirinktos sąskaitos duomenų langas.

Atliekamas sąskaitos duomenų pakeitimas.

Paspaudus išsaugojimo mygtuką išsaugomi nauji sąskaitos duomenys.

Sąskaitų sąrašo peržiūra
Pasirenkamas klientas.

Atidaromas langas, kuriame parodomas sąrašas su kliento sąskaitų duomenimis.

Sąskaitos eilutės sukūrimas
Atidaromas sąskaitų sąrašo langas.

Atidaromas sąskaitos langas.

Atidaromas langas, kuriame įvedami sąskaitos eilutės duomenys.

Paspaudus sukūrimo mygtuką nurodytai sąskaitai sukuriama nauja eilutė.

Sąskaitos eilučių sąrašas papildomas nauja sąskaitos eilute.

Sąskaitos eilutės duomenų pakeitimas
Atidaromas sąskaitų sąrašo langas.

Atidaromas sąskaitos langas.

Atidaromas langas su pasirinktos sąskaitos eilutės duomenimis.

Atliekamas duomenų pakeitimas.

Paspaudus išsaugojimo mygtuką, išsaugomi nauji sąskaitos eilutės duomenys.

Sąskaitos eilutės ištrynimas

Atidaromas sąskaitų sąrašo langas.

Atidaromas sąskaitos langas.

Atidaromas sąskaitos eilučių sąrašo langas.

Pasirinkus sąskaitos eilutę spaudžiamas ištrynimo mygtukas.

Paklausiama, ar tikrai norima ištrinti pasirinktą eilutę.

Jei atsakymas teigiamas pasirinkta sąskaitos eilutė pašalinama.

Iš sąskaitos eilučių sąrašo pašalinama sąskaitos eilutė.

Sąskaitos eilučių sąrašo peržiūra

Atidaromas sąskaitų sąrašo langas.

Atidaromas sąskaitos langas.

Atidaromas langas, kuriame matomas sąrašas su sąskaitų eilučių duomenim.

Kvito sukūrimas

Atidaromas kvitų sąrašo langas.

Atidaromas langas, kuriame įvedami naujo kvito duomenys.

Paspaudus sukūrimo mygtuką, sukuriamas naujas kvitas.

Kvitų sąrašas papildomas sukurtu kvitu.

Kvito duomenų pakeitimas

Atidaromas kvitų sąrašo langas.

Atidaromas langas su pasirinkto kvito duomenimis.

Atliekamas duomenų pakeitimas.

Paspaudus išsaugojimo mygtuką, išsaugomi nauji kvito duomenys.

Kvitų sąrašo peržiūra

Pasirenkamas klientas.

Atidaromas langas, kuriame matomas sąrašas su kvitų duomenimis.

Vartotojo prisijungimas

Atidaromas langas, kuriame vartotojas įvedą vardą ir slaptažodį.

Paspaudus prisijungimo mygtuką duomenys išsiunčiami į serverį.

Jei vartotojas prisijungė (duomenys teisingi) vartotojas gauna unikalų identifikatorių, kurį turi išsisaugoti.

Jei vartotojo neprisijungė (vardas arba slaptažodis netinkami) apie informuojamas vartotojas. Ir jam vėl pateikiamas prisijungimo langas.

Vartotojo atsijungimas.

Vartotojas paspaudžia atsijungimo mygtuką.

Į serverį išsiunčiama atsijungimo komanda.

Vartotojas pereina į prisijungimo būseną.

4.2. Analizė

Objektų sąrašo sudarymas ir objektų klasifikavimas

Identifikuojami naudojami objektai, bei kam objektai priklauso, taip pat išskiriami bendriniai objektai apjungiantys kelis naudojamus objektus.

Esant dideliam panaudojimo atvejų skaičiui tikslinga papildomai pasižymėti, kuriuose panaudojimo atvejuose kiekvienas objektas veikia, toks pasižymėjimas palengvina tolesnę objektų analize.

Kam priklauso objektas	Objekto pavadinimas	Bendrinis objektas
	Vartotojas	
Vartotojas	Prisijungimo langas	Langas
	Serveris	
Vartotojas	Prekė	
Prekė	Prekės langas	Langas

Vartotojas	Prekių sąrašas	Sąrašas
Prekės	Prekės duomenys	Duomenys
Klientas	Kvitas	
Kvitas	Kvito langas	Langas
Klientas	Kvitų sąrašas	Sąrašas
Kvitas	Kvito duomenys	Duomenys
Klientas	Sąskaita	
Sąskaita	Sąskaitos langas	Langas
Klientas	Sąskaitų sąrašas	Sąrašas
Sąskaita	Sąskaitos duomenys	Duomenys
Sąskaita	Sąskaitos eilutė	
Sąskaitos eilutė	Sąskaitos eilutės langas	Langas
Sąskaita	Sąskaitos eilučių sąrašas	Sąrašas
Sąskaitos eilutė	Sąskaitos eilutės duomenys	Duomenys
Vartotojas	Klientas	
Klientas	Kliento langas	Langas
Vartotojas	Klientų sąrašas	Sąrašas
Klientas	Kliento duomenys	Duomenys

Objektų atributų ir metodų sudarymas

Objekto pavadinimas	Atributai	Metodai
Vartotojas	Vardas Slaptažodis Identifikatorius Prisijungimo būsena Prekių sąrašas Klientų sąrašas	Prisijungti Atsijungti
Prisijungimo langas	Vartotojas	Prisijungti
Serveris		Siųsti komandą
Prekė	Prekės duomenys	Sukurti prekę Gauti prekės duomenis Išsaugoti prekę
Prekės langas	Prekė	Išsaugoti Sukurti Parodyti
Prekės duomenys	Prekės kodas Pavadinimas Kaina Likutis Aprašymas	Gauti Sukurti Išsaugoti
Prekių sąrašas		Gauti prekių sąrašą
Kvitas	Kvito duomenys	Gauti kvito duomenis Išsaugoti kvitą Sukurti kvitą
Kvito langas	Kvitas	Išsaugoti Sukurti

		Parodyti
Kvito duomenys	Kvito data Dokumentas Suma	Gauti Sukurti Išsaugoti
Kvitų sąrašas		Gauti kvitų sąrašą
Sąskaita	Sąskaitos duomenys Sąskaitos eilučių sąrašas	Gauti sąskaitos duomenis Išsaugoti sąskaitą Sukurti sąskaitą
Sąskaitos langas	Sąskaita	Išsaugoti Sukurti Parodyti
Sąskaitos duomenys	Sąskaitos numeris Sąskaitos data Suma	Gauti Sukurti Išsaugoti
Sąskaitų sąrašas		Gauti sąskaitų sąrašą
Sąskaitos eilutė	Sąskaitos eilutės duomenys	Gauti sąskaitos eilutės duomenis Išsaugoti sąskaitos eilutę Sukurti sąskaitos eilutę Ištrinti Sąskaitos eilutę
Sąskaitos eilutės langas	Sąskaitos eilutė	Išsaugoti Sukurti Parodyti
Sąskaitos eilutės duomenys	Prekės kodas Pavadinimas Kiekis Vieneto kaina	Gauti Sukurti Išsaugoti Ištrinti
Sąskaitos eilučių sąrašas		Gauti sąskaitos eilučių sąrašą
Klientas	Kliento duomenys Kvitų sąrašas Sąskaitų sąrašas Kvitas Sąskaita	Gauti kliento duomenis Išsaugoti klientą Sukurti klientą
Kliento langas	Klientas	Išsaugoti Sukurti Parodyti
Kliento duomenys	Kliento kodas Pavadinimas Adresas Telefonas Kiti telefonai El. pašto adresas Suma už nupirktas prekes Sumokėta suma Skola	Gauti Sukurti Išsaugoti
Klientų sąrašas		Gauti klientų sąrašą

Potencialių Windows OS apvalkalo objektų suradimas sudarytame objektų sąraše

Naudojant Windows apvalkalo komponentus sąrašus geriausia rodyti pasinaudojant IShellFolderView sąsaja, sąrašų pasirinkimą galima organizuoti sąsajos IShellFolder pagalba. Sąraše pasirinkto objekto savybes peržiūrėti arba keisti galima naudojantis IShellPropSheetExt sąsaja.

Galima suskirstyti objektus pagal jų sąsajų tipus (objektai neturintys Windows sąsajos neminimi):

Vartotojas	IShellFolder
Prekės langas	IShellPropSheetExt
Prekių sąrašas	IShellFolderView
Kvito langas	IShellPropSheetExt
Kvitų sąrašas	IShellFolderView
Sąskaitos langas	IShellPropSheetExt
Sąskaitų sąrašas	IShellFolderView
Sąskaitos eilutės langas	IShellPropSheetExt
Sąskaitos eilučių sąrašas	IShellFolderView
Kliento langas	IShellPropSheetExt
Klientų sąrašas	IShellFolderView
Klientas	IShellFolder

Nesunku pastebėti, kad visi langai yra sąsajos IShellPropSheetExt, o sąrašai – IshellView, todėl jau klasifikuojant objektus galima kartu surašyti ir jų sąsajas, taip pat pastebima, kad praktiškai visi dialogo langai ir visi objekto savybių langai yra IShellPropSheetExt, todėl atliekant klasifikaciją galima atsižvelgti į šią išvadą ir visus objektus, kurie naudojami objektų atributams rodyti ar keisti apjungti į vieną grupę.

Taip pat matosi, kad visi sąrašai naudoja tą pačią IShellFolderView sąsaja.

Reikia nepamiršti, kad sąrašai yra klasifikuojami pagal kažkokius atributus, kaip klasifikatorių reikia naudoti sąsają IShellFolder, kur leidžia sąrašų pasirinkimą atvaizduoti medžio principu.

4.3. Projektavimas

Programos architektūros sudarymas

Programos architektūros vaizdą sudaro aukščiausio lygio programos modelis. Turint programos architektūrą lengviau įvertinti reikiamų komponentų išsidėstymą ir jų tarpusavio sąveiką.

Klasių diagramos sudarymas ir sekų diagramų sudarymas, remiantis panaudojimo atvejais ir trūkstamų metodu suradimas

Pradinis klasių diagramos modelis gaunamas iš objektų lentelės, kur kiekvienas objekto tipas yra klasė, objekto atributas – klasės atributas, o objekto metodas – klasės metodas.

Sekų diagramos sudaromos remiantis panaudojimo atvejų aprašymais. Detalizuojant kiekvieną panaudojimo atvejį, tokiu būdu surandami papildomi klasių metodai ir sudaromas dalinis programos veikimo algoritmas, kuriuo bus galima pasinaudoti vėlesnėse programavimo stadijose.

Windows OS apvalkalo objektų įrašymas klasių diagramoje

Kuriant klasių diagramą reikia atsižvelgti į apvalkalo komponentų specifiką: rodomi objektai turi būti specialaus tipo ir saugomi specialioje struktūroje (ITEMIDLIST), taip pat šiems objektams reikalingos sukūrimo, šalinimo operacijos. Rodomi duomenys struktūrizuojami pasinaudojant IShellFolder sąsaja, o rodomi naudojantis IShelView sąsaja, todėl būtinai reikia sukurti šių sąsajus realizacijas. Taip pat papildomiems veiksmų apdirbimams (kontekstinio meniu apdirbimui, greitų klavišų kombinacijoms ir kt.) naudojami apvalkalo sąsajos, kurios gali būti realizuotos kitose klasėse (priklauso nuo kuriamos programos architektūros).

Klasių diagramos optimizavimas

Sukūrus pradinį klasių diagramos modelį jį galima optimizuoti – išskirti bendras klases, sugrupuoti, išskirti virtualius metodus ir atlikti kitus reikalingus klasių modelio optimizavimo veiksmus.

4.4. Programavimas

Klasių kodo užrašymas

Pasinaudojant sukurta klasių diagrama sukuriamas statinis programos vaizdas – reikalingos klasės, komponentai ir sąsajų realizavimai.

Realizuojami klasių metodai. Programuojant klasių metodus labai naudinga pasinaudoti sukurtomis sekų diagramomis, nes jos aprašo objektų sąveiką ir tada belieka užrašyti šių sąveikų programinį kodą.

4.5. Testavimas

Testavimo plano sudarymas

Sudaromame testavimo plane reikia sudaryti testuojamų klasių ir metodų sąrašą, bei jų atlikimo tvarką.

Sudarant testavimo planą reikia atsižvelgti, kad naudojamų sąsajų pilno testavimo nebūtina atlikti, nes operacinės sistemos sąsajos buvo testuojamos operacinės sistemos kūrėjų ir jų testavimas naudos neatneš. Tačiau jei klasė turi papildomų metodų, kuriuos reikia ištestuoti, jie ignoruojami neturėtų būti.

Testų sudarymas

Testų sudarymo metu sudaromi klasių ir metodų testai, pagal anksčiau sudarytą planą.

Sudarant testus rekomenduotina, jei tik tai įmanoma, naudotis automatiniais testų sudarymo įrankiais.

Testų atlikimas

Testų atlikimo metu atliekami sukurti testai, išsaugomi jų rezultatai (tiek klaidingi tiek ir teisingi).

Klaidų taisymas ir pakartotinis testavimas

Testams, kuriems buvo gauti klaidingi testų rezultatai atliekamas klaidų taisymas ir pakartotinas testavimas.

Ištaisius rastas klaidas ir atlikus jų testavimą reikia atlikti pakartotiną visų testų atlikimą, kad įsitikinti jog programoje klaidų neliko.

5 Pastabos ir komentarai programų kūrimui naudojant Windows apvalkalo komponentus

5.1. Reikalavimų specifikavimas

Dalis panaudojimo atvejų gaunama iš reikalavimų specifikavimo, bet dažniausiai jie nusako tik pagrindines kryptis, kuriomis dirba vartotojas. Poreikių aprašyme dažniausiai nefiksuojami mažiau reikšmingi ir todėl ne taip pastebimi panaudojimo atvejai ir funkcijos.

Jei naudojami duomenų objektai reikia atkreipti dėmesį į tokių objektų sukūrimą, išsaugojimą, ištrynimą ir jų sąrašo gavimą. Panaudojimo atvejų diagramoje nesant kuriam nors iš minėtų panaudojimo atvejų, tai gali būti praleistas panaudojimo atvejis. Jei kažkurio iš šių panaudojimo atvejų duomenų objektui nereikia, tai reikia užfiksuoti, kad vėlesniuose etapuose, tai netaptu bereikalingo darbo šaltinis.

Taip pat dažnai būna praleidžiamas vartotojo panaudojimo atvejis, reikia pasitikslinti kaip vartotojas dirba su programa, ar egzistuoja saugumo lygiai, grupės, vartotojo autentifikacija būtina ar nebūtina. Kaip ir kur saugomi vartotojo prisijungimo duomenys.

Sudarius išsamią panaudojimo atvejų diagramą nebėra reikalo rašyti atskirą funkcinę specifikaciją (nebent užsakovas to pageidauja), nes visos reikalingos funkcijos užfiksuotos panaudojimo atvejuose ir pagal jų realizavimą galima sekti programos kūrimo eiga. Funkcinės specifikacijos sudarymas būtų panaudojimo atvejų perrašymas ir papildymas nereikšmingomis funkcijomis, apie kurių egzistavimą galima pasakyti iš karto (pavyzdžiui: panaudojimo atvejuose nefiksuojamas atskiras prisijungimas prie duomenų bazės, kas būtų įrašyta į funkcinę specifikaciją, bet apie prisijungimo funkcijos reikalingumą galima pasakyti iš karto).

Išsami panaudojimo atvejų diagrama suteikia keletą privalumų: suteikia pilnesnį kuriamos programos vaizdą, iš anksto numatoma, kokių resursų reikės, panaudojimų atvejų diagramą lengviau analizuoti, nei funkcinės specifikacijos lentelę. Išsami panaudojimo atvejų diagrama leidžia nerašyti funkcinės specifikacijos, taip sutaupant laiko.

Atliekant panaudojimo atvejų aprašymą geriausia pradžioje juos aprašyti abstraktesne forma, apibendrintai. Sudarius bendrus aprašymus patikrinama dalykinės srities procesai ir panaudojimo atveju atitikimas, taip siekiant išsiaiškinti, ar visi būtini panaudojimo atvejai aprašyti. Kai tikrinimas atliktas kiekvienas panaudojimo atvejis peržiūrimas detaliau, aprašant tikslesnę jo atlikimo eigą ir naudojamų objektų dalykinės srities atributus. Kad dalykinės srities objektų atributų paieška būtu greitesnė ir patogesnė, visus atributus geriausia iškelti į atskira žodyną.

5.2. Analizė

Pagrindinių objektų sąrašą galima gauti iš panaudojimo atvejų. Likusius objektus galima gauti iš panaudojimo atvejų aprašymų.

Duomenys dažniausiai atvaizduojami dvejopai: pasirinkto objekto – matomi visi to objekto duomenys ir to paties tipo objektų sąrašas – matomas sąrašas su dalimi objektų atributų. Tai galioja praktiškai visiems atvaizduojamiems objektams, todėl jei darant analizę pastebima, kad kažkurio iš šių objektų nėra sudarytame objektų sąraše reikia patikrinti panaudojimo atvejus, nes tokių objektų nebuvimas gali signalizuoti klaidą reikalavimų specifikavime.

Sudarant objektų atributų sąrašą dalį atributų galima gauti iš **dalykinės srities atributų aprašymų**. Rekomenduotina pagrindinių objektų naudojamus duomenis perkelti duomenų objektui, kartu perkeliant ir duomenų gavimo, išsaugojimo ir šalinimo operacijas (jei tai įmanoma), tokiu atveju atsiradus naujiems pagrindinio objekto atributams operacijos su duomenim bus lokalizuotos ir nereikalaus papildomo pagrindinio objekto perprogramavimo (jis apsiribos reikiamų duomenų apdirbimo operacijomis).

Išankstinis objektų suskaidymas į apvalko komponentus ir savitus objektus leidžia pamatyti tikslesnį programos modelio vaizdą ir jos šios programos veikimo modelį.

5.3. Projektavimas

Sudarius bazinę programos architektūrą ją galima suskaidyti į dalis – dalis naudojanti Windows apvalko sąsajas (realizuojančią šias sąsajas) ir likusią dalį, architektūroje taip pat tikslinga nurodyti per kokias dalis ar komponentus šios dvi dalys jungiasi. Padėti sudaryti bazinį programos architektūros modelį gali objektų sąrašas ir objektų klasifikacija – kuri duoda apibendrintą sistemos vaizdą.

Kuriant pradinę klasių diagramą reikia atsižvelgti į programos architektūrą, taip pat labai naudinga kuo stipriau atskirti Windows sąsajų realizavimo ir kitas klases, taip bus stipriau modulizuojama programa ir palengvinamas testavimas, nes sąsajų realizavimo klases kartais būna pakankamai sudėtinga ištestuoti. Taip pat reikia nepamiršti, kad pilnam objekto funkcionavimui (duomenų struktūrizavimui, rodymui ir interaktyvumui su vartotoju) dažniausiai reikia bent dviejų skirtingų sąsajų realizavimo – IShellFolder ir IShellView, o taip pat duomenų struktūros ITEMIDLIST, kuria operuoja Windows apvalkas, realizavimas.

5.4. Programavimas ir testavimas

Kuriant programos kodą ir testuojant rekomenduojama kuo dažniau naudoti automatinius įrankius (klasių kodo generavimą iš klasių diagramos, automatinius testų sudarymus), taip bus sutaupomas laikas ir padaryta mažiau klaidų.

6 Problemų analizė

Didžiausia problema kuriant programas naudojančias Windows apvalko komponentus yra sudėtingas jų testavimas, nes ribojamas priėjimas prie šių komponentų derinimo, geriausias šios problemos sprendimas programos logiką realizuoti atskirose klasėse ir sąsajų komponentams palikti tik vaizduojamąja dalį.

Taip pat naudojant Windows operacinės sąsajos komponentus kartais gali iškilti nesklandumų su pačiu operacinės sistemos apvalku ir reikėti jį perkrauti (dėl klaidingo veikimo apvalkas pats kartais gali persikrauti, kitais atvejais gali neatlaisvinti naudojamų resursų, todėl negalima susieti kuriamos binarinės bibliotekos).

Išvados

Išanalizavus naudojamus Windows sistemos apvalko komponentus ir jų panaudojimo galimybes nustatyta kada tikslinga kurti programas naudojančias šiuos apvalko komponentus – programos, kurių esmę sudaro struktūrizuotų duomenų atvaizdavimas, peržiūra ir redagavimas.

Sudarytas programinės įrangos kūrimo modelis ir programos planas geriausiai tinka mažoms ir vidutinio dydžio programos, nes jis remiasi aktyviu bendravimu su būsimo programos naudotoju ar naudotojais, tiek reikalavimų sudarymo etape, tiek ir vėlesniuose (sudaromas detalus panaudojimo atvejų sąrašas, keletą kartų peržiūrimi ir tikslinami panaudojimo atvejų aprašymai). Kuriant didelę sistema gautis labai didelis kiekis informacijos, kurią reikėtų apibendrinti ir išanalizuoti, o tai būtų labai sunku ir galima palikti klaidų, tokiu atveju visą projektą galima skaidyti į mažesnes dalis ir kiekvienai atskirai pritaikyti programų kūrimo metodą/planą.

Literatūra

1. R. J. Torres “User Interface design and development”, Prentice Hall RTR, 2002
2. Albertas Čaplinskas. Programų sistemų inžinerijos pagrindai. I, II dalys, Matematikos ir informatikos institutas, 1996
3. Ian Sommerville. Software engineering (sixth edition), Person Education, 2001
4. MCSD training kit, Microsoft Press, 2000
5. Dino Esposito. Visual C++ Windows Shell Programming, Wrox Press, 1998
6. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/programmersguide/shell_int/shell_int_extending/extensionhandlers/shell_ext.asp (2006-03-12)
7. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/programmersguide/shell_adv/namespaceextension/nse_works.asp (2006-03-12)
8. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ishellview2/ishellview2.asp (2006-03-12)
9. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ipersistfolder2/ipersistfolder2.asp (2006-03-12)
10. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ipersistfolder3/ipersistfolder3.asp (2006-03-12)
11. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/iqueryinfo/iqueryinfo.asp (2006-03-12)
12. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ishellbrowser/ishellbrowser.asp (2006-03-12)
13. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ienumidlist/ienumidlist.asp (2006-03-12)
14. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ishellfolder/ishellfolder.asp (2006-03-12)
15. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/iinputobject/iinputobject.asp (2006-03-29)
16. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/iinputobjectsite/iinputobjectsite.asp (2006-03-29)
17. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ifileviewer/ifileviewer.asp (2006-03-29)
18. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/icontextmenu/icontextmenu.asp (2006-03-29)
19. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/icontextmenu2/icontextmenu2.asp (2006-03-29)
20. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/icontextmenu3/icontextmenu3.asp (2006-03-29)

21. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ishellextinit/ishellextinit.asp (2006-03-29)
22. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ishellchangenotify/ishellchangenotify.asp (2006-03-29)
23. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/iextracticon/iextracticon.asp (2006-03-29)
24. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ishellicon/ishellicon.asp (2006-03-29)
25. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ishelllink/ishelllink.asp (2006-03-29)
26. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/ishellpropsheetext/ishellpropsheetext.asp (2006-03-29)
27. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/icopyhook/icopyhook.asp (2006-03-29)
28. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/idragsourcehelper/idragsourcehelper.asp (2006-03-29)
29. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/shellcc/platform/shell/reference/ifaces/idroptargethelper/idroptargethelper.asp (2006-03-29)
30. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/com/html/13fbe834-1ef8-4944-b2e4-9f5c413c65c8.asp (2006-03-29)
31. http://www.codeproject.com/shell/NamespaceExtImpl.asp (2007-02-15)
32. http://www.codeproject.com/shell/shlext.asp (2007-02-15)

Priedai

Priedas 1. Windows apvalko komponentų sąsajų ir jų funkcijų aprašymai.

IShellView

Apvalko vaizdas yra paprastas langas, su savo stiliais ir lango procedūra. Vaizdo objektas yra dešinėje Windows Explorer pusėje matomas vaizdas, kuriame atvaizduojamas pasirinkto katalogo turinys. Vaizdo objektas naudojasi IShellView sąsaja darbui su aplanko vaizdu ir katalogo vaizdo nusakymui. IShellView sąsaja išvesta iš IOleWindow.

Metodai, kuriuos reikia įgyvendinti norint naudotis IShellView komponentu:

AddPropertySheetPage()	Leidžia įdėti papildomus puslapius į Folder Options... dialogo langą.
CreateViewWindow()	Sukuria ir gražina langą įterptą į dešinę Windows Explorer lango pusę.
DestroyViewWindow()	Sunaikina langą, kuris sukuriamas CreateViewWindow().
EnableModeless()	Šiuo metu Window Explorer nenaudoja. Gražinama E_NOTIMPL.
EnableModelessSV()	Šiuo metu Window Explorer nenaudoja. Gražinama E_NOTIMPL.
GetCurrentInfo()	Gražina dabartinio katalogo nustatymus naudojantis FOLDERSETTINGS struktūrą.
GetItemObject()	Gražina rodykle į kontekstinio meniu arba tarpinės atmintinės (clipboard) sąsają, nurodytiems objektams. Dažniausiai iškviečiama iš bendrų dialogo langų.
Refresh()	Priverčia perpaišyti katalogo turinį.
SaveViewSate()	Išsaugo vaizdo būseną.
SelectItem()	Pakeičia vieno ar daugiau elementų pažymėjimo būseną.
TranslateAccelerator()	Apdoroja kiekvieno klavišo paspaudimus, kai fokusas suteikiamas išplėtimui. Gražina S_OK jei Windows Explorer neturi toliau apdoroti klavišo paspaudimo.

UIActivate()	Iškviečiama kai aktyvacijos būsena pasikeičia. Pavyzdžiui kai katalogas aktyvuojamas arba deaktyvuojamas.
GetWindow()	Gražina vaizdo lango identifikatorių (handle). Šis metodas paveldimas iš IOleWindow.
ContextSensitiveHelp()	Katalogas turi įeiti arba išeiti į nuspėjamos pagalbos režimą ir apdoroti gaunamas žinutes skirtingai. Paprastai šis metodas išplėtimuose neperrašomas. Šis metodas paveldėtas iš IOleWindow.

IShellView2

IShellView2 išplečia IShellView sąsają.

CreateViewWindow2()	Naudojamas sukurti naują apvalkalo vaizdo langą. Naujas langas sukuriamas dešinėje Windows Explorer lango pusėje.
GetView()	Gražina dabartinį arba standartinį apvalkalo vaizdą.
HandleRename()	Naudojamas pakeisti elemento identifikatorių.
SelectAndPositionItem()	Naudojamas pasirinkti ir pozicionuoti elementą apvalkalo vaizde.

IShellBrowser

Sąsaja naudojama išplėtimų objektų peržiūrai, naudojamas Explorer programoje ir Open File dialogo lange.

BrowseObject()	Perduoda pranešimą Windows Explorer programai peržiūrėti kitą katalogo objektą.
EnableModelessSB()	Perduoda pranešimą Windows Explorer programai įgalinti (enable) arba deaktyvuoti (disable) modalius dialogo langus.
GetControlWindow()	Gražina lango identifikatorių (handle) nurodantį į peržiūros valdiklį.
GetViewStateStream()	Gražina IStream sąsają, kuri gali būti naudojama vaizdo specifinės informacijos saugojimui.

InsertMenuSB()	Leidžia konteineriui įterpti meniu grupes į meniu, kuris rodomas kai išplėtimas yra naudojamas arba peržiūrimas.
OnViewWindowActive()	Iškviečiamas apvalkalo vaizdo, kai vaizdo langas arba vienas jo vaikų gauna fokusą arba aktyvuojami.
QueryActiveShellView()	Gražina dabartinį aktyvų (rodomą) apvalkalo vaizdo objektą.
RemoveMenusSB()	Leidžia konteineriui pašalinti bet kurį meniu elementą iš meniu ir išlaisvinti visus jo naudojamus resursus.
SendControlMsg()	Išsiunčia valdiklio žinutę į įrankių juostą (tool bar) arba būsenos juostą (status bar) esančius Windows Explorer lange.
SetMenuSB()	Instaliuoja meniu į vaizdo lange.
SetStatusTextSB()	Nustato ir parodo būsenos tekstą objekte, kuris yra lango rėmelio konteineryje.
SetToolBarItems()	Įdeda įrankių juostos elementą į Window Explorer įrankių juostą.
TranslateAcceleratorSB()	Apdirba greitų klavišų kombinacijas peržiūros langui, kol vaizdas yra aktyvus.

IEnumIDList

Pateikia metodus leidžiančius apvalkalui išvardinti katalogo turinį.

Clone()	Sukuria naują elemento objektą su tuo pačiu turiniu ir būsena, kaip dabartinis objektas.
Next()	Gražina nurodytą elementą iš elementų sekos ir atsižvelgiant į dabartinę poziciją.
Reset()	Grįžta į elementų sekos pradžią.
Skip()	Praleidžia nurodytą elementų kiekį numeruotoje sekoje.

IShellFolder

Pateikia metodus leidžiančius apvalkalui prieiti prie papildomų katalogų standartiniu keliu. IShellFolder sąsaja paslepia papildomą kodą nuo Explorer.

BindToObject()	Gražina IShellFolder objektą pakatalogiui.
BindToStorage()	Užprašomas rodyklė į objekto išsaugojimo sąsają.
CompareIDs()	Nustato dviejų failų arba katalogų reliatyvią tvarką.
CreateViewObject()	Užprašomas objektas, kuris gali būti naudojamas gauti informaciją arba sąveikauti su juo.
EnumObjects()	Leidžia klientui nustatyti katalogo turinį sukuriant elementų seką ir gražinant jo IEnumIDList sąsają.
GetAttributesOf()	Gražina vieno arba daugiau failų ar katalogų atributus, esančius objekte atstovaujamame IShellFolder sąsaja.
GetDisplayNameOf()	Gražina vardą nurodytam failo objektui arba pakatalogiui.
GetUIObjectOf()	Gražina OLE sąsają, kuri gali būti naudojama metodų iškvietimui nurodytam failo objektui arba katalogui.
ParseDisplayName()	Išverčia failo objekto arba katalogo rodomą vardą į elementų identifikatorių sąrašą.
SetNameOf()	Nustatomas failo objekto arba pakatalogio rodomas vardas, pakeičiant elemento identifikatorių proceso metu dinamiškai.

IPersistFolder

Leidžia inicializuoti kai kuriuose apvalkalo išplėtimus ir bet kuriuos vardo srities išplėtimus.

Initialize()	Perduoda apvalkalo katalogo objektui komandą, kad šis pats inicializuotusi pagal jam perduota informaciją.

IPersistFolder2

Išplečia IPersisFolder sąsają. Naudojamas gauti informaciją iš apvalkalo katalogo objektų.

GetCurFolder()	Gražina ITEMIDLIST įrašą katalogo objektui.

IPersistFolder3

Išplečia IPersisFolder2 sąsają. Leidžia katalogų objektams realizuoti nestandartinį elgimąsi arba katalogų nuorodas.

GetFolderTargetInfo()	Pateikia katalogo nuorodos vietą ir atributus.
InitializeEx()	Inicializuoja katalogą ir nurodo jo vietą vardo srityje. Jei katalogas yra nuoroda, metodas taip pat nurodo katalogo vietą į kurį rodo katalogo objektas.

IQueryInfo

Gražina atributus (flags) ir informacijos tekstą elementams esantiems kataloge.

GetInfoFlags()	Gražina informacinius atributus (flags) elementui. Šiuo metu metodas nenaudojamas.
GetInfoTip()	Gražina informacinį tekstą elementui.

IShellIconOverlay

Naudojamas nustatyti objekto ikonos padengimui.

GetOverlayIconIndex()	Gražina ikonos padengimo, esančio sisteminiame paveiksliukų sąraše, indeksą.
GetOverlayIndex()	Gražina padengimo, esančio sisteminiame paveiksliukų sąraše, indeksą.

5.2. Apvalko sąsajos:

IInputObject

Naudojamas pakeisti vartotojo sąsajos aktyvacijos ir proceso greitintuvus įvedimo objektui priklausiančiam apvalkalui.

IInputObject sąsaja realizuojama jei kuriamas apvalkalo objektas, kuriame įvedami duomenys. Šio objekto tiesiogiai iškviesti nereikia, jis iškviečiamas iš apvalkalo ir informuoja apie įvedimo elementų aktyvaciją arba leidžia apdoroti klaviatūros klavišų greitintuvus (accelerators).

HasFocusIO()	Nustato ar bent vienas iš lango objektų turi klaviatūros fokusą.
TranslateAcceleratorIO()	Leidžia objektui apdoroti klaviatūros greitintuvus.
UIActivateIO()	Vartotojo sąsaja aktyvuoja arba deaktyvuoja objektą.

IInputObjectSite

Naudojamas įvedimo elementų klaviatūros fokuso pasikeitimo komunikacijai.

OnFocusChangeIS()	Informuoja apie pasikeitusį fokusą.

IContextMenu

Sąsaja iškviečiama iš apvalkalo, kad sukurti arba sulieti meniu su apvalkalo objektu.

Naudojamas kai reikia sukurti dinaminį meniu, priklausantį nuo naudojamo apvalko objekto. Arba kuriamas vardo srities išplėtimas, kuriam reikia sukurti meniu nuorodas.

GetCommandString()	Gauna informaciją apie meniu komandą, įskaitant pagalbos eilutę ir nuo kalbos nepriklausančią arba kanoninį komandos vardą.
InvokeCommand()	Įvykdo komandą susietą su meniu elementu.
QueryContextMenu()	Papildo meniu nurodyta komanda.

IContextMenu2

Naudojamas sukurti arba sulieti meniu elementus, susietiems su apvalkalo objektu, kai meniu saugoma savininko paišomi meniu elementai.

Šią sąsają reikia realizuoti jei apvalkalo išplėtimui arba vardo srities išplėtimui reikia apdoroti įvykiu: WM_INITMENUPOPUP, WM_DRAWITEM, WM_MEASUREITEM.

HandleMenuMsg()	Leidžia kliento IContextMenu objektui apdoroti parnešimus susietus su savininko paišomais meniu elementais.

IContextMenu3

Naudojamas sukurti arba sulieti meniu elementus, susietus su apvalkalo objektu, kai meniu realizacijai reikia apdoroti WM_MENUCHAR pranešimus.

HandleMenuMsg2()	Leidžia kliento objektams (IContextMenu3 klasės) apdoroti susietas su savininko paišomais meniu elementais.

IShellExtInit

Naudojamas apvalkalo objektų išplėtimui. Ši sąsaja naudojama IContextMenu ir IshellPropSheetExt objektų inicializacijai (kito tipo objektai šia sąsaja nesinaudoja).Sąsajos tiesiogiai iškviesti nereikia, nes ji iškviečiama iš apvalkalo.

Initialize()	Inicializuoja savybių lango, meniu ir pertempk ir paleisk išplėtimų objektus.

IShellChangeNotify

Naudojamas informuoti vardo srities išplėtimą kai elemento identifikatorius (ID) pasikeičia.

OnChange()	Iškviečiamas informuoti vardo srities išplėtimą, kad įvyko įvykis paveikiantis objektą.

IExtractIcon

Sąsaja leidžia gauti ikoną susietą su objektu esančiu kataloge.

Extract()	Ištraukia ikonos paveikslėlį iš nurodytos vietos.
GetIconLocation()	Gražina ikonos vietą ir indeksą.

IShellIcon

Sąsaja naudojama susieti ikonos indeksą su IShellFolder objektu.

GetIconOf()	Gražina ikoną objektui esančiam viduje nurodyto katalogo.

IShellLink

Sąsaja skirta apvalkalo nuorodų kūrimui, keitimui ir sprendimui.

GetArguments()	Gražina komandinės eilutės argumentus susietus su apvalkalo nuorodos objektu.

GetDescription()	Gražina apvalkalo nuorodos objekto aprašymo eilutę.
GetHotkey()	Gražina apvalkalo nuorodos objekto greito klavišo kombinaciją.
GetIconLocation()	Gražina apvalkalo nuorodos objekto ikonos vietos (kelią ir indeksą).
GetIDList()	Gražina apvalkalo nuorodos objekto elementų identifikatorių sąrašą.
GetPath()	Gražina apvalkalo nuorodos objekto kelią ir failo vardą.
GetShowCmd()	Gražina apvalkalo nuorodos objekto rodomą komandą.
GetWorkingDirectory()	Gražina apvalkalo nuorodos objekto darbinį katalogą.
Resolve()	Bando surasti apvalkalo nuorodos objektą, net jei jis buvo perkeltas ar pervardintas.
SetArguments()	Nustato apvalkalo nuorodos objekto komandinės eilutės argumentus.
SetDescription()	Nustato apvalkalo nuorodos objekto aprašymo eilutę.
SetHotkey()	Nustato apvalkalo nuorodos objekto greitų klavišų kombinaciją.
SetIconLocation()	Nustato apvalkalo nuorodos objekto ikonos vietą (kelią ir indeksą).
SetIDList()	Nustato apvalkalo nuorodos objekto rodyklę į elementų identifikacijos sąrašą (PIDL).
SetPath()	Nustato apvalkalo nuorodos objekto kelią ir failo vardą.
SetRelativePath()	Nustato apvalkalo nuorodos objekto reliatyvų kelią.
SetShowCmd()	Nustato apvalkalo nuorodos objekto rodomą komandą.
SetWorkingDirectory()	Nustato apvalkalo nuorodos objekto darbinio katalogo pavadinimą.

IShellPropSheetExt

Sąsaja leidžia savybių lange (rodomame failų objektui), įdėti naują arba pakeisti egzistuojantį puslapį.

AddPages()	Įdeda vieną arba daugiau savybių puslapių rodomų failų objektui. Kai reikia pardoti failo savybių langą, apvalkalas iškviečia šį metodą kiekvienam savybių apdorojimo objektui užregistruotam rodomo failo tipui.
ReplacePage()	Pakeičia savybių lango puslapį Control Panel objektams.

ICopyHook

ICopyHook yra COM tipo sąsaja skirta sukurti kopijavimo valdikliui (en. *copy hook handler*). Kopijavimo valdiklis yra apvalkalo išplėtimas, kuris nusprendžia ar gali apvalkalo katalogas arba spausdintuvas būti perkeliamas, kopijuojamas, pervadinamas arba pašalinamas. Apvalkalas iškviečia CopyCallback metodą prieš atliekant vieną iš šių operacijų.

CopyCallback()	Nusprendžia ar apvalkalui galima perkelti, kopijuoti, pašalinti arba pervadinti katalogą arba spausdintuvo objektą.

IDragSourceHelper

Sąsajos pagalba galima nustatyti paveiksliuką, kuris bus rodomas kol tęsis pernešimo ir palaikomo (drag-and-drop) operacija.

InitializeFromBitmap()	Inicializuoja pernešamų paveiksliukų menedžerį langų valdikliui.
InitializeFromWindow()	Inicializuoja pernešamų paveiksliukų menedžerį valdikliui su langu.

IDropTargetHelper

Sąsaja leidžia numetamam objektui atvaizduoti paveiksliuką, kol šis yra virš numetamo objekto.

DragEnter()	Informuoja pernešamų paveiksliukų menedžerį, kad buvo iškviestas pernešamo objekto IDropTarget::DragEnter metodas.

DragLeave()	Informuoja pernešamų paveiksliukų menedžerį, kad buvo iškviestas pernešamo objekto IDropTarget::DragLeave metodas.
DragOver()	Informuoja pernešamų paveiksliukų menedžerį, kad buvo iškviestas pernešamo objekto IDropTarget::DragOver metodas.
Drop()	Informuoja pernešamų paveiksliukų menedžerį, kad buvo iškviestas pernešamo objekto IDropTarget::Drop metodas.
Show()	Informuoja pernešamų paveiksliukų menedžerį parodyti arba paslėpti pernešamą paveikslėlį.

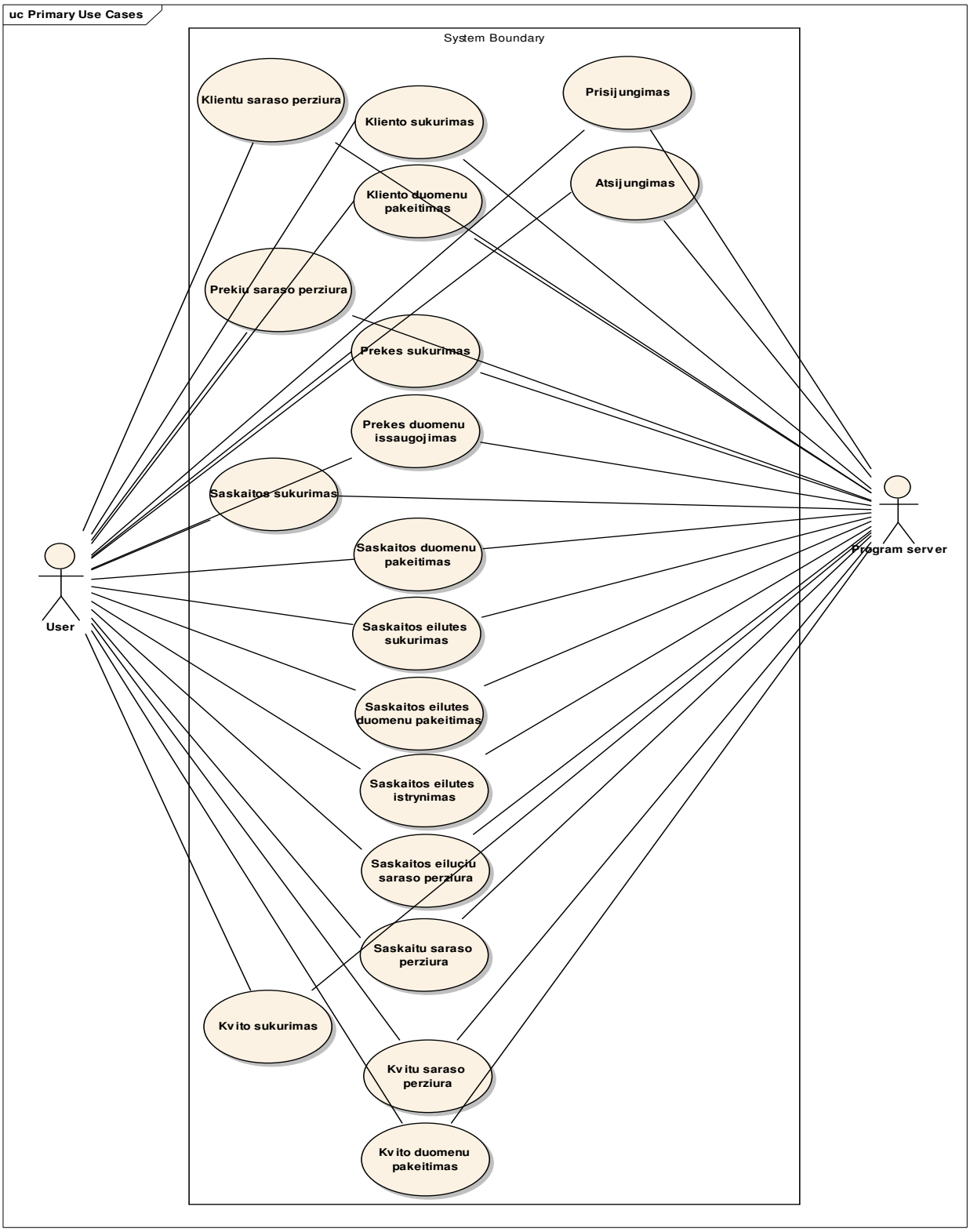
IDropTarget

Sąsaja atsakinga už pernešimo ir paleidimo (drag-and-drop) operacijas programoje. Paleidimo programa (drop-target application) atsakinga už: objekto paleidimo ant taikinio efekto nustatymą, validžių duomenų pateikimą kai paleidimas įvyksta, atgalinio ryšio su šaltiniu užtikrinimas, kad šaltinis galėtų atlikti vizualias manipuliacijas (tokias kaip kursoriaus nustatymo), pertempimo peržiūros (scrolling) realizavimą, programų registravimą ir atlaisvinimą kaip paleidimo taikinius.

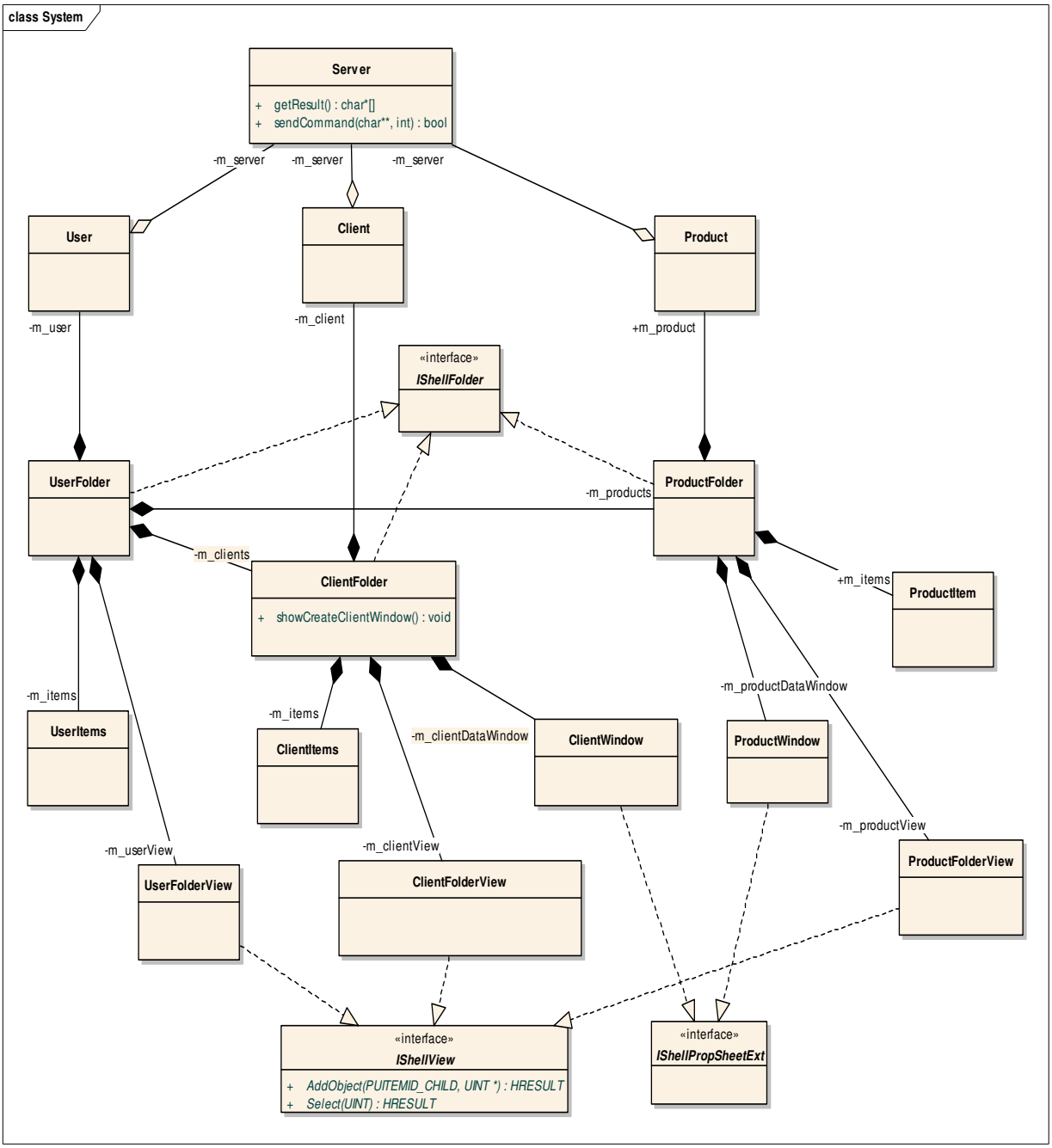
Šią sąsają reikia realizuoti jei kuriama programa gali būti pertempimo ir paleidimo operacijos taikiniu. IDropTarget sąsaja susieta su programos langu ir realizuojama programos lango objekto. Norint užregistruoti lango objektą kaip paleidimo taikinį (drop target) reikia iškviesti RegisterDragDrop() metodą.

DragEnter()	Nustato ar paleidimas (drop) gali būti priimtas ir jo paleidimą įvykdo, jei jis galimas.
DragOver()	Vartotojas informuojamas pasinaudojant DoDragDrop() funkcija.
DragLeave()	Priverčia paleidimo taikinį sustabdyti jo įvykius.
Drop()	Numeta (drop) duomenis taikinio lange.

Priedas 2. Panaudojimo atvejų diagrama.

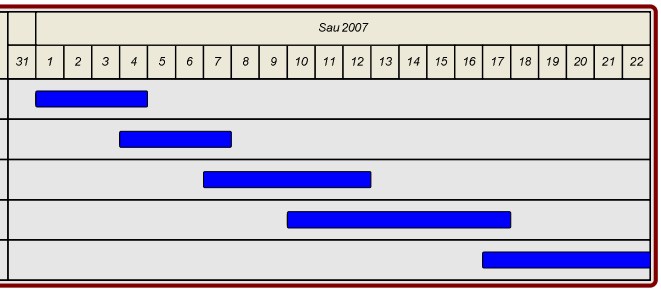


Priedas 3. Pradinė klasių diagrama



Priedas 4. Pavyzdinio programos projekto vieno ciklo Ganto diagrama

ID	Task Name	Start	Finish	Duration
1	Reikalavimų specifikavimas	2007-01-01	2007-01-04	4d
2	Analizė	2007-01-04	2007-01-07	4d
3	Projektavimas	2007-01-07	2007-01-12	6d
4	Programavimas	2007-01-10	2007-01-17	8d
5	Testavimas ir taisymas	2007-01-17	2007-01-24	8d



Priedas 5. Dalykinės srities duomenų aprašymas

Klientų sąrašo eilutės duomenys – kliento kodas, kliento pavadinimas, adresas, pagrindinis telefonas, elektroninio pašto adresas, pinigų suma už nupirktas prekes, sumokėtų pinigų suma, skola.

Kliento duomenys – kliento kodas, kliento pavadinimas, adresas, pagrindinis telefonas, kiti telefonai, elektroninio pašto adresas, suma už nupirktas prekes, sumokėta suma, skola.

Prekių sąrašo eilutės duomenys – prekės kodas, prekės pavadinimas, prekės kaina, prekės likutis.

Prekės duomenys – prekės kodas, prekės pavadinimas, prekės kaina, prekės likutis, prekės aprašymas.

Sąskaitos sąrašo eilutės duomenys – sąskaitos numeris, sąskaitos data, sąskaitos suma.

Sąskaitos duomenys - sąskaitos numeris, sąskaitos data, sąskaitos suma, kliento pavadinimas.

Sąskaitos eilutės sąrašo duomenys – prekės kodas, eilutės tekstas, kiekis, vieneto kaina, eilutės suma.

Sąskaitos eilutės duomenys - prekės kodas, eilutės tekstas, kiekis, vieneto kaina, eilutės suma.

Kvitų sąrašo eilutės duomenys – kvito data, kvito dokumentas, kvito suma.

Kvito duomenys - kvito data, kvito dokumentas, kvito suma.

Klientų sąrašo eilutės duomenys – kliento kodas, pavadinimas, adresas, pagrindinis telefonas, kiti telefonai, el. pašto adresas, suma už nupirktas prekes, sumokėta suma, skola.

Kliento duomenys - kliento kodas, pavadinimas, adresas, pagrindinis telefonas, kiti telefonai, el. pašto adresas, suma už nupirktas prekes, sumokėta suma, skola.

Priedas 6. Kompaktinis diskas

Santrauka

Figoras, Dainius. Informatikos magistro baigiamasis darbas: programų integravimas Windows operacinėje sistemoje. Darbo vadovas prof. L. Sakalauskas; Šiaulių Universitetas. – Šiauliai, 2007. – 46 lapai.

Darbo tikslas – išanalizuoti esamus programų kūrimo modelius, Windows operacinės sistemos sąsajos komponentus ir sudaryti programos kūrimo modelį pritaikytą kurti programoms integruotoms į Windows operacinę sistemą panaudojant sistemos apvalko sąsajas.

Darbe išanalizuoti dažniausiai naudojami programų kūrimo modeliai - krioklio, evoliucinis, komponentinis, žingsninis, spiralinis, universalus, MSF (Microsoft Solution Framework). Jų pagrindu sukurtas programų kūrimo modelis skirtas mažiems ir vidutinio dydžio programiniams projektams kurti. Sudarytas reikalingų Windows apvalko sąsajų sąrašas ir jų aprašymai.

Pagal sudaryta modelį sukurtas pavyzdinis programos projektas ir surašytos pastabos ir komentarai palengvinantys tokio pobūdžio programų kūrimą. Taip pat aprašytos pagrindinės problemos ir jų sprendimo būdai.

Software integration into Windows Operative System

By Dainius Figoras

Supervisor Prof. L. Sakalauskas

Summery

Aim of the work: To analyze existing software developing models, connecting components of Windows Operative System and create a model of software development that further could be used to develop programs integrated into Windows Operative System by using the connecting shell of the system.

Commonly used software development models were analyzed in the thesis, including cascading, evolutional, componential, stepwise, spiral, universal and MSF (Microsoft Solution Framework) models. A novel model of software development was established on bases of investigated models that could be used for development of minor and middle software projects. Furthermore, the list with a detail elucidation of Windows shell connections was added. According to the established model, the software project was developed. In addition, common problems, possible solutions as well as general comments on software development were discussed. This could facilitate the mentioned software development.