**ŠIAULIŲ UNIVERSITETAS**

INFORMATIKOS FAKULTETAS

INFORMATIKOS KATEDRA

**Andrius Božnis**

# ATVIRO KODO GIS TECHNOLOGIJŲ TYRIMAS

**Magistro darbas**

Darbo vadovas: prof. habil.dr. G. Kulvietis

Šiauliai, 2006 m.

# Įvadas

GIS- viena iš pačių perspektyviausiu šiuo metu informacinių technologijų sričių. Distancinis skenavimas, globalaus pozicionavimo sistemos(GPS) vis labiau plinta ir vis daugiau randa pritaikymo būdų. GIS technologijos savyje sujungia atvaizdų skaitmeninę analizę ir duomenų bazių sistemas. Tai suteikia labai platų gis pritaikymą, GIS naudoja įvairius metodus duomenims tirti, priklausomai nuo jų pobūdžio todėl pritraukia labai įvairių sričių specialistus.

Ateitis priklauso sistemoms, apjungiančiom GIS, dirbtinio intelekto ir ekspertinių sistemų elementus. Ypatingai ekspertinės sistemos ir GIS technologijos turi didelį potencialą, palydovinių nuotraukų tyrimui ir analizei. Ši informacija dar visai neseniai buvo neprieinama plačiajai visuomene, ir tokios informacijos disponavimas bei jos apdorojimas ESM(Elektroninių skaičiavimo mašinų) pagalba buvo tik labai galingų kompanijų ar didelių valstybės institucijų jėgoms. Dabar tokio tipo programinė įranga ir technika telpa ant rašomojo stalo, todėl ji darosi vis prieinamesnė privačiam sektoriui. Tačiau dėl GIS įrangos brangumo natūraliai susiklostė, jog programos skirtos informacijai apdoroti buvo kuriamos stambiems klientams, o atviro kodo programinė įranga vystėsi universitetų ir kitų mokslinių įstaigų priežiūroje, todėl jomis ne visada lengva naudotis, žmogui neturinčiam specialaus išsilavinimo, o komercinės programos tiesiog per brangios. Šios ir kitos sąlygos įtakojo dabartinių atviro kodo programų kūrimą, kurios yra orientuotos į daug kasdieniškesnius sprendimus nereikalauja vartotojo jokių ypatingų žinių ir turi puikia grafinę aplinką.

**Aktualumas.** Akivaizdu jog šiandien į atviro kodo programinę įranga jau žiūrima rimtai. Tačiau kasdienėje veikloje ji dar pakankamai retai naudojama. Dažniau namų vartotojai, ar privačios įstaigos renkasi piratines programas ir rizikuoja savo verslu ir baudomis, nei ieško alternatyvų. Tai įtakoja didelė legalios programinės įrangos kaina, bei informacijos stoka. Dabar knygynuose galima rasti lietuviškų leidinių skirtų atviro kodo biuro programų paketams, o GIS technologijoms knygų lietuvių kalba apskritai nepavyko rasti. Atviro kodo GIS programų begalės, tačiau kurios iš jų tinkamos ir kuo jos skiriasi? Tuo labiau atviro kodo programų dokumentacija daugiausiai parašyta anglų kalba. Atviro kodo idėjas skelbiančiame tinklapyje www.akl.lt taip pat informacija labai kukli šiuo klausimu. Būtent šios aktualios problemos paskatino analizuoti esamą situaciją ir ieškoti alternatyvių problemos sprendimo būdų.

**Mokslinis naujumas.** Šis darbas ko gero pirmasis Šiaulių universitete analizuojantis atviro kodo GIS technologijų programinę įrangą bei panaudojimo galimybes. Nepaisant to, jog pasaulyje GIS technologijos pritraukia dideles mokslo žmonių grupes, Lietuvoje aktyviausiai veikia tik savo programinę įrangą reklamuojančios firmos. Net ne visi Lietuvos universitetai dėsto GIS technologijų pagrindus savo informacinių technologijų studentams.

**Nauda.** Atviro kodo GIS analizės praktinė nauda pasireiškia tuo jog atlikus atvirų standartų, bei juos įgyvendinančių programų analizę iš bendros atviro kodo GIS programų masės, surenkama pilnavertė GIS sistema turinti galimybę apdoroti, kaupti GIS duomenis bei juos teikti įvairiais duomenų perdavimo būdais. Tokią sistemą galima įdiegti bet kokioje įmonėje ar mokslo įstaigoje, taip išvengiant autorinių teisių pažeidimų, bei suteikiant galimybe visiems suinteresuotiems asmenims pasinaudoti GIS technologijų teikiama nauda.

Šiame darbe atlikęs analizę išskyriau pačius bendriausius atviro kodo GIS programų bruožus, kurie reikalingi tarpusavio programų darbui organizuoti, remiantis atvirais GIS standartais. Atlikau testus, kurių rezultatai leidžia spręsti, apie surinktos sistemos našumą, bei įvertinti kokio dydžio darbams ji tinka atlikti. Taip pat realizavau įskiepi, kurio tikslas pademonstruoti įskiepių sistemos pritaikymo glimybes. Pateikiau šios sistemos trūkumus bei privalumus, kurie leis kiekvienam lengviau įsivaizduoti minimos sistemos pritaikymo glimybes.

# 1 GEOGRAFINIŲ INFORMACINIŲ SISTEMŲ ANALIZĖ

## 1.1 Įvadas

Geografinės informacinės sistemos plėtojamos jau daugiau kaip 30 metų ir per tą laiką virto patikimu teritorinio planavimo įrankiu. GIS programinės įrangos raidai didelę įtaka darė ne tik tobulėjanti kompiuterinė technika, bet ir naujai atsirandantys duomenų šaltiniai, tokie kaip palydovinės nuotraukos, taip vadinama nuotolinio zondavimo technologija. Iš kitos pusės kūrėsi įvairių valstybių kartografijos informaciją kaupiančios organizacijos tokios kaip JAV(USGS) ir Kanados (CGIS), kurios savo darbo pasėkoje sukūrė naujų standartų GIS duomenims saugoti. Kompiuterinei technikai patobulėjus, ir pasidarius prieinamesnei ir smulkioms organizacijoms, bei visuomenei atsirado naujos terpės GIS programoms veikti. Iki tol visos sistemos veikė tik UNIX šeimos operacinėse sistemose. Galiausiai GPS technologijos paplitimas suteikė GIS informacijai naujas panaudojimo galimybes. Tokiu būdu iki mūsų dienų GIS technologijose vyrauja įvairiausi duomenų formatai, kurie įvairiose šalyse yra naudojami su skirtingu populiarumu, dėl to skirtingai jie ir yra "prigiję". Šią painiavą stengiasi išnarplioti standartai, kurie čia vaidina labai svarbų vaidmenį. Ypatingai greitai populiarėja būtent atviri standartai kurie lengviausiai prigyja atviro kodo programinėje įrangoje.

## 1.2 GIS technologijų raida

Gis technologijos pradėjo vystytis kartu su pirmųjų kompiuterių atsiradimu, tačiau oficialiai kalbama, kad GIS technologijos gyvuoja tik apie 30 metų. Viskas priklauso nuo to kaip pažiūrėsi, ar skaičiuosime nuo pirmųjų bandymų laboratorijose, ar nuo panaudojimo praktikoje. Pirmą kartą 1959m. Waldo Tobler [1] bandė panaudoti kompiuterį kartografinių duomenų apdorojimui, taip buvo sukurtas MIMO (map in – map out) duomenų apdorojimo modelis skirtas kartografijos duomenų apdorojimui EMS pagalba. MIMO principai buvo kelrodis tolesniam duomenų apdorojimo, kaupimo ir atvaizdavimo GIS technologijų srityje. MIMO tūrėjo visus būtinus elementus, kurie ir dabar yra realizuoti šiuolaikinėje programinėje įrangoje. Tačiau tais laikas tokia sąvoka kaip "programinė įrangą" buvo neatrasta. Tačiau niekas nestovėjo vietoje. Kompiuteriai darėsi galingesni ir labiau prieinami visuomeninėm organizacijom bei universitetam. Tačiau jie dažniausiai dirbo tekstiniame režime, o grafinės galimybės buvo labai ribotos, todėl duomenų vizualizacijai buvo naudojami ploteriai (spausdintuvai), kas buvo ganėtinai brangu ir nepatogu.

1963m. Harvarde buvo įsteigtas tyrimu centras, tiriantis ir kuriantis vektorinių duomenų apdorojimo algoritmus, bei programinę įrangą. Čia suburta mokslininkų, inžinierių ir programuotojų komanda sukūrė SYMAP. Tai buvo kompiuterinis paketas galintis spausdinti apytikslius žemėlapius ant tuo metu veikusių būgninių spausdintuvų. Savo laiku jis buvo labai populiarus, jį pardavė beveik 500 kopijų tiražu. O tai buvo labai daug, žinant kiek išviso pasaulyje tuo metu buvo kompiuterių, be to čia suskaičiuotos tik legalios kopijos. Vėliau išėjo analogiškas CALFORM paketas, tačiau jis nebuvo toks populiarus, nes gebėjo spausdinti tik ant ploterių, kurie buvo labai brangūs. SYMAP buvo pirmasis plačiai paplitęs kompiuterinis paketas darbui su geografiniais duomenimis. Jis supažindino labai platų vartotojų ratą su GIS darbo ypatumais . Taip pat šiame tyrimu centre buvo išugdytą ne mažai specialistų, žymių projektų ir firmų pradininkų kaip Jack Dangermond (ESRI), Lawrie Jordan and Bruce Rado (ERDAS).

Beveik tuo pačiu metu Kanadoje 1963m. buvo pradėtas didelis projektas CGIS(Canadian Geographic Information System) skirtas Kanados žemėnaudos analizei visos valstybės mastu.

Tai buvo pirmasis tokio masto projektas, kuris atliktas viso kontinento mastu ir labai dideliu masteliu. Projekto vykdymui vadovavo Roger Tomlinson, dažnai vadinamas GIS tėvu, vien už tai, jog jis įtikino tuometinę Kanados vyriausybę, jog vertą ir naudinga buvo imtis tokio projekto. Tai buvo ne vienintelis jo nuopelnas ir užsiėmimas, jis taip pat dirbo dar keliose organizacijose, tarp jų ir ARDA (Agricultural Rehabilitation and Development Administration ). Bendradarbiaujant kartu su IBM kompanija buvo padaryti ženklūs postūmiai kelių technologijų vystymuisi: Būgniniam skaneriui; Duomenų indeksavimo algoritmui (Morton Index); Topologinio kodavimo schemai. CGIS projektas, kaip kartografinių duomenų rinkimo ir kaupimo sistema buvo užbaigtas iki 1971m., tačiau jis gyvuoja ir dirba iki šiol.

1969m. susikūrė ESRI (Environmental Systems Research Institute) kompanija. Pradžioje ji veikė kaip ne pelno siekianti įmonė ir užsiėmė GIS informacijos analize, bei įvairiais smulkiais žemėnaudos projektais. Vėliau įvairių miesto infrastruktūrų projektavimu. Dar vėliau ji sukauptą patirtį panaudojo kuriant įvairius įrankius duomenų analizei. Savo darbe jie naudojo GRID paketą, kuris buvo pradėtas kurti Harvardo Grafikos Laboratorijoje. Juo naudojosi iki pat 1982 kol buvo sukurtas ARC/INFO. Iki šio, galima sakyti legendinio, produkto dar buvo sukurtas trimatis GRID. Vadinamas GRID TOPO, bei darbui su vektoriniais duomenim skirta PIOS(Planning Information Overlay System, Planavimo informacinė sluoksninė sistema). Dabar tai pasaulinio garso įmonė, kurianti visame pasaulyje labai plačiai naudojama GIS programinę įrangą ARC/INFO. Pasižyminčia vartotojui lengvai perprantama aplinka, ko iki tol GIS sistemos nepasižymėjo, bei labai stabiliu darbu ir gausa analizei skirtų įrankių. Kai kurie jos sukurti duomenų saugojimo formatai, darbo principai yra labai paplitę ir priimami kaip standartai, naudojami įvairiose kituose GIS programinės įrangos projektuose.

Dar šeštame dešimtmetyje prasidėjusios kosminės lenktynės, šaltasis karas, bei smarkiai besivystanti JAV pramonė ortofotografinius duomenų rinkimo metodus padarė nepakankamus. Surinktų duomenų buvo per mažai, tai buvo pakankamai brangu, ir labai sunku įgyvendinti didesnei teritorijai. Todėl buvo nuspręsta sukurti žemę stebėsiantį palydovą. Pirmiausia reikėjo patobulinti naudojamą, taip vadinam nuotolinio zondavimo(remote sensing) technologiją, kurios pagalba būtų galima fiksuoti ne tik paprastą vaizdą bet ir žemės mineralinius turtus. Tais laikais JAV kariniai lėktuvai jau turėjo kameras galinčias fiksuoti ne tik matomą šviesą, bet ir spektrus atitinkančius imfraraudonuosius spindulius(šilumą). Tačiau šiam projektui to buvo maža. Jam buvo skirtos didelės lėšos, į projektą buvo įtraukta įvairių sričių specialistų, ir galų gale 1972m buvo paleistas pirmasis JAV LandSat palydovas. Jame buvo įmontuotos natūralios šviesos, imfraraudonųjų spindulių kameros žemės vaizdams kurti, multispektrinis skeneris, skirtas žemės radiometrinio atvaizdo kūrimui, ir duomenų įrašymo bei perdavimo į žemę įrangą[4]. Jo užduotis buvo kartografiniu duomenų rinkimas apie žemės resursus, žemdirbystės, miškininkystės plotus, vandens išteklius, mineralinius, geologinius išteklius, užterštumą, okeanografijos duomenų rinkimas. Tuo metu neįtikėtina atrodžiusi technologija, galėjo skirti keturis atsispindėjusių bangų spektrus, neskaitant matomos šviesos ir imfraraudonųjų spindulių, bei skiriamoji raiška siekė 80metrų vienam pikseliui. Tai buvo labai sėkmingas projektas, tebeveikiantis iki šiol, dabar jau veikia 7 LandSat palydovas. Palydovo suteikti duomenys buvo kur kas tikslesni, juos buvo galima gauti praktiškai bet kurios žemės vietos, bei per palyginti trumpą laika. Taip pat tai buvo žymiai ekonomiškesnis būdas nei skraidyti su specialiais lėktuvais virš reikiamos teritorijos. Sekantis etapas buvo informacijos panaudojimas. Reikėjo pereiti nuo modelio, kai informacija naudojasi tik "išrinktieji" prie laisvai prieinamo modelio. Tačiau kompiuterių, kurie buvo būtini informacijai apdoroti, buvo nedaug ir labai brangūs. Tuomet buvo sugalvota unikali, tuo metu, schema, kai prie vieno kompiuterio, per modemus ar skirtąsias linijas jungėsi klientai ir per terminalą dirbo. Tačiau pats terminalas, labai kukli aplinka todėl duomenys buvo atvaizduojami ne grafiškai, o tekstiniu formatu. Todėl analizės galimybės ir jos vizualizavimas buvo taip pat ganėtinai sudėtingas. Tokia informaciją galėjo suprasti tik profesionalai, žinantys į ką žiūri, arba ją atspausdinus specialiais spausdintuvais.

Jei kalbėti apie šiuos laikus, tai greičiausiai jei būtų vykdoma apklausa, su kuo asocijuojasi GIS samprata žmonėms, tai tikriausiai 99% būtų GPS. GPS (Global Positioning System, Globalinio pozicionavimo sistema) buvo pradėta kurti 1978 metais, tais pačiais metais buvo paleistas ir pirmasis palydovas iš 24. Pradžioje GPS buvo kuriama kaip galingas ginklas skirtas dideliu tikslumu valdyti įvairių raketų trajektorijas. Ir tik 1984m. buvo leista naudoti GPS civilinėms reikmėms, tačiau su tam tikrais apiribojimais - SA (Selective Availability), specialiai sugeneruota ir į GPS imtuvą perduodama paklaida. Tokių būdu paprastas civilinis GPS imtuvas galėjo pasiekti iki 50m tikslumą, o po 2000m. kai buvo išjungtas SA tikslumas padidėjo nuo 1 iki 3m.

GPS labai greitai atrado savo nišą kasdieniame naudojime. Nuo lėktuvų ir laivų sekimo sistemų iki nešiojamų GPS skirtų rasti kelią mieste, ar net savo automobilį. Žinoma taip pat jis puikiai tiko žemėlapių sudarymui, tuo pačiu ir GIS. Dabar tai viena be kitos neįsivaizduojamos sistemos.

Devinto dešimtmečio pradžioje JAV karinė vadovybė pradėjo ieškoti GIS programinės įrangos skirtos aplinkos būklės monitoringui, analizei bei modeliavimui, tai dalinai įtakojo dar 8 –tame dešimtmetyje JAV priimtas aktas dėl šalies aplinkosaugos politikos. Po komercinių GIS programų analizės buvo prieita išvada jog nei viena iki tol sukurta programinė įranga šiam tikslui netiko. Tuomet buvo priimtas sprendimas susikurti savo unikalią sistemą kuri vėliau įgijo GRASS (Geographic Resource Analysis Support System) pavadinimą. Taigi buvo pasamdyti programuotojai, ir pradėta kurti hibridinė rastrinė-vektorinė sistema VAX UNIX aplinkai. Tai buvo pirma rimta GIS sistema skirta UNIX aplinkai, bei pirma atviro kodo sistema, kuri savo analizės galimybėm gali lygintis su bet kuria rimta GIS komercine uždaro kodo programine įranga.

Devinto dešimtmečio pabaiga ir iki pat praėjusio amžiaus pabaigos, GIS programinės įrangos pasaulyje tvyrojo chaosas. Atsirado daugybė, palyginti pigių, personaliniam kompiuteriams skirtų paketų, kurie nesilaikė jokių standartų, nes jokių oficialių ir privalomų standartų nebuvo. Kiekvienas su savais rastrinių ir vektorinių duomenų saugojimo formatais, savom metodikom. Dauguma paketų buvo ir yra vieni kitų kopijos, t.y. jokių naujovių - analizės, saugumo ar paieškos, ar bet kokioje kitoje srityje. Tiesiog kitas gamintojas ir tiek. Tačiau vienas lyderis išliko tai ARC/INFO. Jis populiarus, patogi, draugiška vartotojo sąsaja, ir pats sau standartas.

## *1.3 Atviro kodo (Open Source) fenomenas*

Atviras kodas tai fenomenas kuris atsirado kartu su interneto išplitimu. Daugumai žmonių atviras kodas asocijuojasi su nemokamom programom deja tai ne visada tiesa. Dažniausiai tokia klaidinga prielaida daroma todėl jog kol kas atviras kodas plačiau išplitęs tarp nemokamų programų, be to dažnai atviro kodo programos vadinamos laisvomis(Free Software an.). Anksčiau buvo manyta jog atviro kodo programos tinkamos tik mažus finansinius išteklius ir mažai besirūpinančiom savo saugumu įstaigoms. Tačiau atviro kodo programų, tokių kaip Apache ir Linux, paplitimas rodo ką kita. Atviro kodo programos vis dar kelia karštus ginčus, saugu ar nesaugu, ir ar tikrai jos tokios nemokamos.

Atviro kodo programa, tai programa kuri laisvai prieinama vartotojui, ne tik kaip paruoštas vykdymui, sukompiliuotas paketas, bet ir kaip programos išeities tekstas. Tačiau tai dar ne pagrindinė savybe. Kad programa galėtų vadintis “laisva” ji turi atitikti specialų OSI(Open Source Initiative) organizacijos nustatyta modelį, o tiksliau 9 būtinus punktus[12].

- **Laisvai platinama**: tai yra programinės įrangos licencinė sutartis negali drausti platinti programinės įrangos kaip kitos programinės įrangos dalies.
- **Atviras kodas**. Programa turi būti platinama kartu su pilnu išeities tekstu(kodu). Arba išeities pilni tekstai turi būti laisvai prieinami internetu ar kitu būdu, užtai nereikalaujant jokio mokesčio.

- **Autoriaus kūrinio originalumo išsaugojimas**: Kodo pataisymai patobulinimai ar kiti pakeitimai turi būti platinami kartu su pradiniu autoriaus sukurtu išeities kodu kaip pataisos(patches).
- **Negali būti diskriminuojamos jokie pavieniai asmenys ar jų grupės**.
- **Negali būti apribota programos panaudojimo sritis**.
- **Licencijos platinimas:** Iš atviro kodo, padarytoms naujoms programoms negali būti taikoma kita sutartis, kaip ta su kuria kodas buvo gautas, bei negali būti jokių papildomų sąlygų.
- **Licencinė sutartis negali būtis specifinė**: Tai yra skirta kažkokiai pavieniai programai ar jų rūšiai.
- **Sutartis negali drausti kitų programų, ar jų kodo panaudojimo**.

Pagrindinė šių apribojimų esmė yra ta, kad licencinė programinės įrangos sutartis negali apriboti vartotojo naudoti programinės įrangos taip kaip jam atrodo tinkamiausia. Yra nemažai šiuos punktus atitinkančių sutarčių, kaip pavyzdžiai galėtų būti "Free Software fundation"(FSF) ar "Gnu public license" (GPL) sukurta Ričardo Stolmano(Richard Stallman).

Atviro kodo pradžia galima laikyti 70-ų metų pradžia. Kada tiksliai ir kas pradėjo šią idėją nėra užfiksuota, tačiau manoma jog atviro kodo pradininkas buvo Ričardas Stolmanas (Richard Stallman). Jis sukūrė pradinį atviro kodo modelį kokį jį turime dabar.

## Atviro kodo trūkumai

Pagrindiniai argumentai prieš atviro kodo programinę įranga yra šie :

- Atviro kodo programinė įranga sudaro galimybę autorinių teisių pažeidimui
- Atviras kodas yra nesaugus, nes potencialūs kenkėjai gali rasti „skilių" ir jomis pasinaudoti.
- Atviro kodo programas sunku įdiegti ir sunku gauti pagalba iškilus problemoms.

Na pats keisčiausias argumentas yra būtent pirmas, nes pati atviro kodo sutartis nurodo autorių teisių apsaugą, ir kaip ji turi būti įgyvendinta. Kitas dalykas yra atviro kodo ir komercinių programų naudojimas kartu. Čia gali kilti konfliktas, jei komercinio produkto sutartyje yra numatyta, jog programos darbo rezultatas, taip pat priklauso programai. Na tai būtų pakankamai keista. Pavyzdžiui jei "Microsoft" pareikštų savo autorines teises į mano parašyta referatą, nes jis buvo parašytas Microsoft Office pagalba. Nors šis pavyzdys juokingas, tačiau tokia situacija gali kilti, pavyzdžiui naudojant web servisus, ar komponentus todėl reikia būti atsargiam, ir gerai išsiaiškinti programinio produkto licencinės sutarties sąlygas.

Antrasis argumentas, su laiku vis silpnėjo, ypač po rastų didelių spragų komerciniuose produktuose, bei šnipinėjimo skandalų ir šiai dienai galima beveik šimtu procentų teigti jog atviro kodo programos yra žymiai saugesnės. Esmė yra ta, jog atvira kodą kasdien peržiūri daugybė programuotojų. Todėl jis nuolat tikrinamas ir nėra jokių kliūčių radus klaidą ją tuoj pat pataisyti. Komercinės programos saugumas remiasi informacijos ribojimo principu. Tai ką su programa galima daryti yra parašyta dokumentacijoje, o tai ko ten nėra tarsi ir neegzistuoja. Deja realybėje yra visai kitaip, klaidas daro visi, todėl svarbu jas laiku ištaisyti. Kadangi kiekvienos kompanijos ištekliai riboti, ji stengsis juos panaudoti spręsti rimčiausioms saugumo ar kitoms problemoms, todėl greičiausiai bus sprendžiamos tos problemos su kuriomis susiduria daugiausiai vartotojų, todėl mažuma liks nuskriausta. Net jei ji ir turi sugebėjimus surasti ir ištaisyti klaidą, dažniausiai komercinių produktų licencijos tai draudžia, todėl teks laukti, o laukimas tai pinigai, o smulkios kompanijos dažniausiai jų neturi.

Kitas aspektas yra šnipinėjimas. Turbūt visi pamena skandalą kilusį kai buvo pastebėta, jog Microsoft Windows operacinė sistema, be jokio vartotojo sutikimo siunčia konfidencialius duomenis. Atrodo pirmasis atvejis buvo pastebėtas 1998m vienos iš IT saugumu užsiimančių firmų darbuotojo, kai jis pastebėjo, jog tinklu keliauja jo kompiuterio identifikaciniai duomenys, nors pats jis to nepageidavo ir jokiose dokumentacijose apie tai nebuvo užsiminta. Vėliau

Microsoft teisinosi, jog tokiais būdais kovoja su piratavimu. Kad ir kaip ten būtų akivaizdu, jog paslėpti programuotojui uždarame programos kode slaptas "duris" (back door), su ar be kompanijos valdžios žinia, yra labai paprasta. Buvo atvejis kai tokios durys "MS FrontPage" buvo nepastebėtos net 4 metus[10], bet žinoma, tik ne piktadarių. Atviro kodo programose tokie dalykai neįmanomi.

Trečiasis argumentas yra teisingas su tam tikrom išlygom. Paimkime tokias populiarios programas kaip "Linux OS" ir "Apache" www serverį. "Linux" jau truputi skiriasi nuo daugumos atviro kodo programų tuo, kad dabar aplink ją buriasi nemažai naujų kompanijų užsiimančių pakankamai nauju verslu, "Linux OS" palaikymu. Palaikymas – tai techninė pagalba, techninė konsultacija, klaidų paieška ir taisymas, naujų funkcijų diegimas. Kitaip tariant, priklausomai nuo distribucijos jūs už tam tikrą mokestį galite gauti vadinamąjį palaikymą 24h per parą, todėl čia jokių ypatingų problemų netūrėtų kilti. Programoms, kurios neturi apmokamo palaikymo galima rasti atsakymus į dominančius klausimus forumuose, žinių kanaluose, el. pašto kanaluose (mailing lists). Žinoma, naivoka tikėtis rasti atsakymus lietuvių kalba. Tačiau tokie projektai kaip "OpenOffice" jau turi tokia galimybę. Be to dažniausiai jei jūs susidūrėte su kokia problema, greičiausiai jūs nesate pirmas todėl pasirausę forumuose atsakymą tikrai rasite. Jei galų gale jūsų naudojamos programos vartotojų ratas menkas, mažai informacijos laisvai prieinamuose šaltiniuose. Jūs visada galite pasisamdyti specialistą, kuris išspręs problemą ir ras reikiamą sprendimą. Nors tai atrodo ir ne kokia alternatyva. Tačiau su atviru kodu jūs niekada neatsirasite aklavietėje, nes visada kodas yra atviras.

## Atviro kodo privalumai

Atviras kodas turi labai daug privalumų:

- Atviras kodas saugesnis.
- Atviras kodas stabilesnis.
- Atviras kodas palankus standartams.
- Atvirą kodą galima taisyti bet kada.
- Atviras kodas turi labai mažai arba visai neturi apribojimų.

Kodėl atviras kodas yra saugesnis nei komercinių programų jau aptariau anksčiau. Tačiau norėčiau priminti, kad tai nėra tik tušti žodžiai ar idėja, kuri gali būt kada nors įgyvendinta. Dar 2000 - siais metais Europos sąjungos vadovybė kurdama prioritetines ES vystymosi kryptis nusprendė, kad reikia įsteigti elektroninę vyriausybę [12]. Ir kaip pagrindą ji pasirinko būtent atviro kodo programinę įrangą. ES raporte apie atviro kodo programinę įrangą sakoma jog, ji tūrėtų būti saugesnė nei bet kuri kita komercinė įranga, nes būtent viešumas nesuteikia progos programos kode paslėpti jokių "šnipų". O klaidos pasitaiko bet kokioje programinėje įrangoje, svarbu laiku jas ištaisyti. Tuo labiau atviro kodo programose, jis nuolat tobulinamas, o komercinėse, po vieno projekto užbaigimo, dažniausiai programuotojai užsiima kitu, todėl jis užsimiršta ir kai tenka taisyti klaidas, reikia jį vėl prisiminti, įsisavinti ir pataisyti. Kaip rodo Sasser viruso, nusiaubusio daugybe kompiuterių, paplitimo istorija, sugaištas laikas klaidos taisymui gali labai brangiai kainuoti.

Atviras kodas labai palankus ir imlus standartams ypač atviriems. Priešingai nei komercinėms kompanijoms, siekiančioms padidinti savo pelną, atviro kodo kūrėjams nereikia pririšti savo vartotojų. Dažnai programinės įrangos gamintojos sukuria savo duomenų saugojimo standartus, kurie yra užpatentuoti, tokių būdu jūs išsaugojate duomenis formate kuriuo negalite atverti ar tik redaguoti jokioje kitoje programinėje įrangoje, nes tai draudžia licencija, arba tiesiog jokios kitos programos to formato nesupranta. Blogiausia jog dažnai su naujesnės versijos analogiška programa išsaugoti duomenys negali būti atverta su senesne versija. Todėl jei jūs kompanijoje turite kelias programas, privalote atnaujinti jas visas, nes kitaip jos tampa tarpusavyje nebesuderinamomis. Dėl šių priežasčių, labai nenoriai, yra komerciniuose produktuose įgyvendinami atviri standartai. Kaip pavyzdį galiu paminėti MS Office ir

OpenOffice produktus. Vos tik Doc dokumento formatas tapo viešas, MS Office paketas ėmė prarasti savo rinką, kaip pagrindinis ir nepakeičiamas teksto apdorojimo paketas.

**Apibendrinimas**

Reikia paminėti, kad atviro kodo naudingumo/žalingumo diskusija yra ne karas prieš komercinius produktus, o dviejų programinės įrangos paradigmų kova – atviro ir uždaro kodo programinės įrangos[10]. Kas laimės? Sunku pasakyti. Panašu, jog abi paradigmos turi savų privalumų ir trūkumų, bei lengvai ras sau panaudojimo nišas pasaulyje. Panašu, jog tai ne karas, o savotiškas savireguliacijos principas, kuris jau pradeda akivaizdžiai veikti šiandieniniame pasaulyje.

Mano nuomone, niekada niekas nebuvo, nebus ir neturi būti už dyka. Kas investuos savo laiką ir gal net pinigus į išradimus iš kurių neturės jokios finansinės ar praktinės naudos?! Žinoma niekas, nebent keletas entuziastų. Tačiau toks principas mus greičiau prives prie stagnacijos nei klestėjimo. Todėl nauji atradimai, algoritmai ir programos visada bus mokamos, tačiau labai svarbu, jog tai kas jau seniai žinoma, atrasta ir priimama kaip savaime suprantami dalykai nebūtų paslėpta nuo visuomenės, o viešai prieinama. Todėl čia atviras kodas atlieka labai svarbų vaidmenį, nes jis neleis atsirasti monopoliams, kurie amžiais valdys programinės įrangos progresą ir raidą, bei suteiks sveiką konkurenciją komerciniams produktams.

## *1.4 Atviro kodo GIS analizė*

Atviro kodo GIS programų nestinga. Jų yra pilna įvairioms platformoms, ir labai įvairiom paskirtim. Daugumą jų galima rasti tinklapyje www.freegis.org, tačiau ne visus. Šiame tinklapyje nuolat atnaujinama informacija apie esamus laisvai platinamus atviro kodo GIS projektus, bei naujai atsirandančius. Kiekvienas kūrėjas, sukūręs savo programą stengiasi informaciją apie ją patalpinti šiame tinklapyje, bei nuolat ją atnaujinti, todėl labai lengva pastebėti kurie projektai yra pačiame"jėgų žydėjime", o kurie jau senai nebeatnaujinami ir baigia savo dieneles. Projektų yra įvairių sričių, tai ir serverinės programos arba pavienės Gis duomenų peržiūros ir redagavimo programos, arba tik peržiūros programos. Taip pat galima rasti komponentų skirtų GIS peržiūros programų ar apletų kūrimui. Daug įvairių bibliotekų GIS duomenims apdoroti, atvaizduoti, tinklui skirtų GIS projektų. Taip pat įrankių įvairioms duomenų transformacijoms atlikti, bei komandinės eilutės įrankių. Taigi įvairovė atviro kodo GIS tikrai didelė ir kažką tinkančio sau tikrai galima rasti, tačiau yra ir trūkumų

Pagrindinė problema yra pats projektų stovis, jei projektas seniai nebeatnaujinamas, tai greičiausiai jis tikrai nėra tobulas, ar labai naudingas. Nes atviro kodo programoms galioja toks dėsnis, jei programa naudinga, tai ja domisi daug žmonių, o tai reiškia greitą klaidų ištaisymą, didelę programos bendruomenę, kuri gali jums padėti rasti atsakymus į reikiamus klausimus. Todėl jei projektas nebeatnaujinamas, tai greičiausiai pagalbos gauti bus sunku, o ir klaidų niekas nepadės ištaisyti, todėl patartina surasti analogiška, kas netūrėtų būti sunku.

Dar viena bėda, ypač su senomis programomis, duomenų formatai, todėl dažna programa išsaugo savo duomenis specifiniu, tik jai suprantamu formatu. Dažnai programos nemoka dirbti su duomenų bazėmis ir išsaugoti duomenis jose, o dirbančiosios išsaugo savo duomenis specifiniu būdu, tai yra tik jom pačiom suprantama lentelių hierarchija, ir joms pačioms suprantamu geografiniu duomenų saugojimo būdu.

Žinoma visus atsakymus galima rasti jų dokumentacijoje. Tačiau manau tikrai neprotinga gaišti laiką prie kiekvienos programos. Tuo labiau kai jos atlieka pakankamai siauro profilio užduotis ir norint pilnai funkcionuojančios sistemos prireiktų ne vienos programos, ir iškiltų rimtos problemos dėl duomenų konvertavimo. Todėl labai svarbu pasirinkti programas kurios laikosi nors kokių standartų.

Pats minimaliausias, mano nuomone, atributas atviro kodo programai yra ESRI kompanijos sukurto SHP failo formato palaikymas. Tai ypatingai palengvina duomenų apsikeitimo problemą ir galima sakyti pirmasis žingsnis suderinamumo problemos sprendimui, tačiau to nepakanka. Kaip jau minėjau ESRI jau senai kuria GIS programas, kurios susilaukė didelio populiarumo visame pasaulyje, todėl jos duomenų formatai, kurie taip pat daugumoj specifiniai(neatviri), yra labai paplitę. Todėl šio failo formato palaikymas suteikia labai dideles galimybes.

Jei jūs nusprendėte pradėti dirbti su atviro kodo GIS programom siūlyčiau domėtis šiomis:

QGIS (Quantum GIS) – C++ parašyta programa, atvaizduoti rastrinius duomenis, dirba su SHP failo formatu, PostGIS duomenų baze, taip pat palaiko WFS ir WMS standarto servisus. Tai labai sparčiai populiarėjanti atviro kodo programa. Pagrindinės to priežastys būtų, jog ji pakankamai nesenai pradėta, todėl laikosi pačių naujausiu standartų, ir gali dirbti su įvairiais duomenimis. Taip pat ji mažai užima vietos, greitai dirba, lengvai perprantama vartotojo sąsaja, bei veikia visose platformose. Joje mažai įrankių skirtų GIS duomenų analizei, tačiau ji turi veikiančia įskiepių sistema, todėl blogiausiu atveju galite susikurti ar susirasti internete įskiepį patys.

GRASS- Atviro kodo programa parašyta C kalba. Tai yra labai didelis GIS analizės įrankių paketas, turintis didelį įrankių pasirinkimą tiek vektoriniams tiek rastriniams duomenims. Kaip minėjau anksčiau GRASS buvo pradėta kurti dar 1982 m. ir labai populiari tarp GIS profesionalų, tačiau savo pozicijas prarado prasidėjus grafinių aplinkų "erai". Tai turbūt vienintelė ir silpniausia jos vieta, nes dauguma jos atliekamų operacijų yra valdomos praktiškai X11 terminalo lange, todėl vartotojas yra priverstas įvedinėti pakankamai daug duomenų klaviatūra, kas darbą daro nepatogų, nenašų ir sudėtingą. Dar viena problema lėmusi jos populiarumo sumažėjimą, turbūt tai, jog ji skirta UNIX šeimos operacinėms sistemoms. Taip pat yra jos versija skirta Linux, tačiau windows aplinkoje ji dirba su Linux aplinką imituojančiu paketu cygwin, kurį paprastiems windows vartotojams, pratusiems prie savaime įsidiegiančių ir susireguliuojančių paketų, įdiegti labai sunku.

UDIG, tai Java kalba parašyta programa. Tačiau ji nuo kitų programų skiriasi tuo jog naudoja Eclipse platformą. Tai yra turi pakankamai gerai standartizuotą ir suprojektuotą įskiepių technologiją. Šios technologijos dėka iš pakankamai standartines GIS funkcijas turinčios programos galima padaryti kiekvieno vartotojo poreikiams pritaikytą įrankį.

Tarp tinklo serverių teikiančių prieiga prie GIS duomenų tinklų pagalba galima paminėti šiuos:

GeoServer - Taip pat Java kalba parašyta programa naudojanti GeoTools biblioteką. Servisas yra tiekiamas laikantis OGC specifikacijų WMS(Web Map Server) ir WFS(Web Feature Service). Šio serverio pagalba galima nesunkiai, vien užklausos eilutę HTTP protokolu formuojančių įrankių pagalba, atlikti įvairias manipuliacijas su GIS duomenimis, bei nesunkiai įdiegti GIS funkcionalumą į savo tinklapį. Internete gausu jau paruoštų PHP ar kita scenarijų kalba parašytų programų, galinčių atlikti pagrindines Gis duomenų manipuliacijos funkcijas.

UMN Mapserver - analogiškas serveris, taip pat teikiantis paslaugas OGC specifikuotų servisų pavidalu. Tačiau parašytas C kalba, yra spartesnis, ir ko gero iš visų analogiškų serverių palaikantys, daugiausiai įvairių duomenų formatų.

Duomenų bazes skirtas GIS galima būtų paminėti dvi, tai PostgreSQL\PostGIS ir MySQL. PostreSQL, bei MSSQL duomenų bazės palaiko geometrijos tipą numatytą OGC specifikacijoje "Simple feature specification for SQL". PostgreSQL yra senbuvė šioje srityje, palyginus su MySQL, ir turi daug didesnį vartotojų ratą. Daugelyje mano minėtų programų galinčių dirbti su duomenų bazėmis MySQL palaikymas dar tik kuriamas, todėl reiktų gerai pasvarstyti ar verta kaip duomenų bazę skirtą GIS rinktis MySQL.

Bibliotekų skirtų darbui su GIS duomenim yra taip pat platus pasirinkimas, ypač didelis pasirinkimas C kalba parašytų bibliotekų:

OGR/GDAL. Tai faktiškai dvi atskiros bibliotekos tačiau dėl tam tikrų istorinių aplinkybių jos sujungtos į vieną. OGR biblioteka skirta darbui su vektoriniais duomenimis, o GDAL skirta rastriniams duomenims, kartu jos suteikia, ją naudojančiai programai labai didelę GIS duomenų formatų įvairovę. Jas naudoja labai daug atviro kodo GIS programų, tokių kaip GRASS, QGIS ir kitos.

Proj4 – C kalba parašyta biblioteka, skirta duomenų transformacijoms, galinti transformuoti duomenis, tarp įvairių koordinačių sistemų, sferoidų, atlikti postūmius ir kitas manipuliacijas

GEOS – C++ kalba parašyta biblioteka, "įkūnijanti" geometrinių figūrų objektus bei operacijas su jomis, tokias kaip Intersect(),Union(),Buffer(), bei predikatus :Relate(), Touches() Disjoint() ir kitus.

Java kalba parašytos bibliotekos yra :

GEOAPI – projektas skirtas suvienodinti Java kuriamų bibliotekų sąsajas(Interfeisus) įvairioms standartinėms GIS operacijoms atlikti. GEOAPI kūrėjai tikisi, jog vieną dieną pavyks visus Java projektus įtraukti į šį projektą ir standartizuoti visas bibliotekas. GEOAPI biblioteką sudaro tik Java sąsajų(Interfeisų) rinkiniai standartinėms GIS procedūroms, visos sąsajos kuriamos laikantis ISO ir OGC standartų bei specifikacijų.

JTS – Java Topology Suite, analogiška C++ GEOS bibliotekai savo paskirtimi. JTS įgyvendina OGC "Simple feature Specification for SQL"[17] geometrijos modelį su visom operacijom ir predikatais. Kadangi geometrines funkcijas kurti labai sunku JTS įgijo dideli populiarumą ir praktiškai visi atviro kodo projektai ją naudoja, iškarto įgydami standartinių geometrinių figūrų rinkinį, bei funkcijas be didelių sunkumų.

WKB4J – Java biblioteka atliekanti manipuliacijas su OGC specifikacijoje numatytais formatais WKB(Well-Known Binary format) ir WKT(Well-Known Text format) kuriais kuoduojamos geometrinės figūros informacija praktiškai visose OGC "Simpe feature specifikation for SQL" suderinamose duomenų bazėse. Taip pat šiuos formatus naudoja kelios bibliotekos.

GeoTools2 – Java kalba parašyta biblioteka skirta GIS duomenų apdorojimui. Taip pat turinti savyje jau paruoštų komponentų GIS aplikacijoms ar apletams kurti. Viena iš jos stipriųjų pusių yra tai jog ji suderinama su OGC standartais. Kas daro šią biblioteką ir ją naudojančias programas tarpusavyje suderinamomis, o tai jog ir komercinių produktų kompanijos diegia šį standartą ir į savo programas, leidžia suprasti jog ateityje ši biblioteka turės labai didelę vertę. Jau dabar ją naudoja nemažai rimtų programų.

Kaip minėjau, čia ne visos programos, kurias man pavyko rasti ir išbandyti, čia sudėjau programas, kurios yra pakankamai plačiai pritaikomos ir gali būti panaudotos be didelių sunkumų. Bibliotekos ir programos, kurių funkcijos yra labai ribotos ir skirtos tik siaurai sričiai, čia nepaminėtos. Beje aš negaliu teigti, jog jos geriausios. Pavyzdžiui, uDig programa atsirado truputį daugiau nei prieš metus, tačiau jau spėjo susikurti ne mažą vartotojų ratą, todėl negaliu užtikrinti, kad ir šiuo metu nėra kuriamas koks nors naujas projektas turintis daugiau galimybių ar kažkokių naujovių. Galbūt tokios palyginti naujos technologijos kaip .NET taip pat pasiūlys ką nors naudingo GIS. Pavyzdžiui jau yra pradėtas analogiškos GeoTools bibliotekos C# kalba kūrimas.

## 1.5 Tyrimo uždaviniai

Problemai suformuluoti paimkime paprastą mažos įmonės modelį, turinčios tris padalinius, arba bent jau žmones atliekančius tris funkcijas, duomenų rinkimą, projektavimą bei analizę, ir duomenų suvedima į kažkokias laikmenas. Tokio tipo modelis praktiškai atitinka nemažai įmonių, kaip pavyzdžiui atliekų surinkimo įmonė: duomenų rinkimas – vairuotojai bei darbininkai fiksuojantys kiemų adresus bei kažkokius ypatumus, kaip surenkamų atliekų kiekį

namų kiemuose ar panašiai. Analizė – administracija, analizuojanti pavyzdžiui kokį maršrutą geriausia pasirinkit siekiant optimizuoti kuro sąnaudas atliekoms surinkti. Įvedimas – GIS darbuotojas(-ai) atnaujinantis miesto planų duomenis ir t.t. Taip pat toks modelis tinka ir kitoms įmonėms, čia norėčiau apsiriboti tik tokiomis, kurioms nebūtina trimatės erdvės analizė, tai yra tokioms įmonėms kurios pavyzdžiui nedirba su vandeniu, kur reikia skaičiuoti nuolydžius ir t.t. Toks apribojimas būtinas, kadangi mano darbe nagrinėjama programinė įranga šios funkcijos neturi. Taigi čia turime bent jau tris klientus kuriems reikalingi duomenys, taip pat duomenys turi būti visiems prieinami vienu metu. Todėl programinė įranga turi tūrėti galimybę keliems klientams dirbti vienu metu.

Problemai suformuluoti paimsiu įmonės kurioje dirbu pavyzdį. Manau šis modelis gali tikti bet kuriai mažai įmonei, nes turi pagrindinius elementus. Tai būtų duomenų rinkimą, suvedimą ir analizę. Duomenų rinkimą, atlieka darbus mieste organizuojantys darbuotojai, duomenų įvedimą atlieku įmonėje aš. O analizė yra atliekama, bei naudojama geografinė informacija, darbuotojų projektuojančių ir ruošiančių naujus darbus. Kol kas šie darbai atliekami naudojant tik vieną ESRI kompanijos programą ArcView 8.2. Jos pagalba kuriama GIS duomenų bazė. Šiai duomenų bazei naudojamas MS Access failas. Norint duomenis pateikti vartotojams duomenys yra eksportuojami į SHP failus, jie pateikiami tinklu, ir naudojant nemokama ESRI ArcExplorer programą, vartotojai gali peržiūrėti GIS duomenis bei atlikti minimalią analizę pagal atributus, sudėtingesniems darbams tenka naudotis MS Access įrankiais. Toks darbo modelis tenkina minimalius poreikius, tačiau nėra tobulas dėl sekančių priežasčių:

- Neužtikrinamas duomenų saugumas. Kadangi operacinėje sistemoje leidimas skaityti failus, reiškia ir leidimą kopijuoti, kas ne visada yra pageidaujama.
- Darbuotojai patys savarankiškai negali iki galo atlikti reikiamų operacijų be vienas kito pagalbos. Pavyzdžiui reikiant sudaryti sudėtingesnę analizę tenka kreiptis į MS Access duomenų bazę, kas reiškia papildoma darbą man.
- Prarandami reliacinio duomenų bazės modelio privalumai.
- Nėra galimybių tinkamai atvaizduoti duomenis vartotojams.
- Vartotojai negali patys taisyti atributinių duomenų.

Didžioji dalis šių problemų yra susiję su galimybės saugoti duomenys duomenų bazėje nebuvimu, kuri būtų prieinama visiems, kita – negalima sukurti unikalios konkretiems duomenims atvaizduoti skirtos vartotojo sąsajos. Iš atviro kodo Gis programų analizės matyti, jog egzistuoja atviro kodo duomenų bazės. Tačiau ArcView programa neturi galimybės dirbti su šiomis duomenų bazėmis. Tuo tarpu vartotojams skirtų programų, su įskiepių diegimo ir kūrimo galimybe, leidžiančia susikurti unikalią vartotojo sąsają, bei dirbančia su atviro kodo Gis duomenų baze, surasti ir pritaikyti problemų nebūtu. Analizuojant šią problemą, pirmiausia tyriau galimybę ArcView programą pritaikyti darbui su PostgreSQL duomenų baze. Šiam tikslui pasiekti buvo panaudota Microsoft COM technologijos galimybės ir ArcView integruotas VBA(Visual Basic for Applications), kurių pagalba buvo bandoma įdiegti papildomą funkcionalumą. Buvo išbandyti du variantai. Pirmiausia buvo bandoma panaudoti jau sukurtą projekto PgArc (http://sourceforge.net/projects/pgarc ) funkcionalumą. Kadangi šis įskiepis gali tik perkelti duomenis iš PostgeSQL duomenis į ArcView ir atvirkščiai, tačiau negali duomenų bazėje įrašyti pakeitimų, buvo sukurtas papildomas kodas leidžiantis pataisytus duomenis įkelti atgal. Šis procesas bei smulkūs jo rezultatai aprašyti Priede Nr.1 “ArcView – PostgreSQL sąveika” . Atlikus pataisymus paaiškėjo, jog integruoto VBA kodo pagalba sukurtas įskiepis dirba labai lėtai, bei negali užtikrinti kelių vartotojų saugaus darbo vienu metu. Todėl šis variantas nepašalina visų minėtų trūkumų.

Antras būdas - sukurti COM komponentą ir pabandyti jį kaip sluoksnio objektą ArcMap programos hierarchijoje. Deja išanalizavus ArcMap programos objektinį modelį paaiškėjo, jog vieno objekto nepakaks. Taip pat reikia sukurti įrankius ir papildomus objektus kurie atliktų operacijos su sukurtu sluoksnio objektu. Galų gale buvo aišku jog tai per didelis darbas ir dėl savo apimčių tikrai negali būti gera alternatyva. Analogiškai ir PostgreSql OLEDB tvarkyklės

tobulinimas, suteiksiantis galimybę atvaizduoti duomenis ArcMap programoje taip per didelis darbas analogiškam funkcionalumui pasiekti. Vienoje elektroninio pašto diskusijoje buvo prieita, mano nuomone, labai teisinga išvada dėl OLEDB tvarkyklės varianto – PostgreSQL OLE DB tvarkyklės tobulinimas siekiant įdiegti pilną geometrijos skaitymo/rašymo funkcionalumą yra uždavinys prieš kurio sprendimą reikia gerai pamastyti ar ne pigiau įsigyti visą trūkstamą programinę įrangą iš ESRI kompanijos.

Apibendrinant galima padaryti išvadas jog pagalbinės, mano nagrinėto įrankio ESRI ArcView 8.2, priemonės (VBA ir COM) nėra nei pakankamai išvystytos, nei tinkamai dokumentuotos, jog suteiktų galimybę įdiegti į programą tokį rimta funkcionalumą, kaip naują duomenų šaltinį. Todėl ArcView tobulinimas, kaip būdas minėtai problemai spręsti yra netinkamas. Šios problemos buvo vienos iš priežasčių paskatinusios pradėti atviro kodo GIS analizę, tokių būdu ieškant išeities bei naujų perspektyvų ir galimybių mažai, bent jau Lietuvoje, tyrinėtoje srityje.

Todėl mano darbo uždaviniai būtų sekantys:

- Išanalizuoti GIS standartus.
- Išanalizuoti atviro kodo programas su intarpais(plug-ins).
- Realizuoti atviro kodo GIS įskiepį.
- Ištirti atviro kodo GIS sistemos technines galimybes.

# 2 ATVIRO KODO GIS PROGRAMINĖS ĮRANGOS ANALIZĖ

## *2.1 Įvadas*

Vystantis Informacinėms technologijoms GIS taip pat neliko nuošalį ir jos teikiamas naujoves pritaikė savo reikmėms, tokiu būdu GIS programinę įranga galimą skaidyti į tris keturias rūšis pagal jų veikimo terpę:

- Duomenų bazių aplikacijos, tai daugiau SQL standarto funkcijų išplėtimas, nei atskira savarankiška programa. Toks SQL išplėtimas leidžia atlikti GIS užklausas paprasčiausia SQL kalba.
- Tinklinės – tai vietiniame tinkle, arba internete veikiančios aplikacijos įgalinančios perduoti GIS informaciją standartiniais www protokolais, tokiu būdu suteikiančiais galimybę dirbti standartinių interneto naršyklių pagalba.
- Klientinės, arba desktop aplikacijos- tai kliento kompiuteryje veikiančios specialiai GIS skirtos programos.
- Mobiliosios aplikacijos - klientinių aplikacijų atmaina skirta nešiojamiems įrenginiams, tokiems kaip delniniai kompiuteriai, dažniausiai skirtos navigacijai.

Visos šios išvardintos dalys, veikia sėkmingai kaip atskirai taip ir kartu, tačiau jų naudojimas kartu sudaro labai galingą sistemą. Tam kad ši sistema tvarkingai veiktų, reikia specialių standartų reglamentuojančių informacijos tarp komponentų mainus.

Iš tiesų GIS naudoja begales su standartais susijusių elementų. Vieni standartai susiję su platesnėm panaudojimo sritim, kaip ryšiui su duomenų bazėmis, ar tinklo klientais palaikyti, kiti siauresnėms sritims, kaip duomenų diske išsaugojimo formatai. Juos būtų galima suskirstyti į tokias sritis:

- Duomenų saugojimo įvairiose laikmenose formatai.
- Duomenų perdavimo tinkle protokolai.
- Duomenų bazių valdymo SQL kalbos išplėtimo standartai.

Informacinių technologijų aušroje, kai kompiuterius turėjo tik labai didelės ir galingos kompanijos. Įvairūs duomenų saugojimo ir perdavimo standartai buvo kuriami vienos įmonės mastu, skirti įmonės vidaus reikmėms. Kai prasidėjo personalinių kompiuterių laikai, ir pradėta

gaminti jiems skirtą programinė įranga, dėl tokios standartų kūrimo tradicijos dažnai kildavo įvairių formatų vadinamieji „karai“, kai konkuruojančios firmos neatskleisdavo ir nesuteikdavo teisės kitoms firmoms naudotis jų formatais. Nuo tokių karų visada nukentėdavo tik vartotojas, kuris netik, kad buvo „pririštas“ prie savo programinės įrangos, bet ir negalėjo jos modifikuoti, kad pasiimtų savo duomenis ir perkeltų kitur. Todėl ėmė kurtis įvairios standartizavimo organizacijos, siekiančios suvienodinti standartus, kaip galima platesniu mastu. Tokius standartus GIS srityje kuria OGC(Open Geosptial Consorcium). Ši organizacija jau vienija apie 309 ivairių sričių organizacijas, tokias kaip ESRI, MapInfo, ORACLE, GOOGLE, NASA, ESA ir daugelį kitų. OGC kuriamiems standartams teikiamas labai didelis dėmesys, nes jos standartai yra laisvai prieinami, todėl labai greitai įgyvendinami ypač atviro kodo programose. Nors komerciniai produktai nelinkę pasiduoti, tačiau jie priversti taikytis su esama situacija, todėl daug jų jau palaiko šiuos standartus, bent jau dalinai, pavyzdžiui skaitymo režime.

## 2.2 GIS standartų tyrimas

GIS saugomą informaciją galima išskaidyti į dvi pagrindines rūšis, pagal joje saugomos informacijos pobūdį- vektorinę ir rastrinę. Pagal joje saugomos informacijos matmenis – dvimatę ir trimatę. Kadangi informacijos pobūdžiai iš esmės yra skirtingi, skirtingi ir būdai juos saugoti failuose.

**Rastriniai duomenys**

Kadangi nuo pačios GIS istorijos pradžios visa informacija dažniausiai buvo saugoma failuose ir programinės įrangos pasirinkimas buvo ribotas, bei pradiniai duomenų kaupimo būdai buvo standartizuoti JAV USGS(United States Geology Survey) ir ANSI tai pasėkoje turime jog dauguma atviro kodo programų puikiai supranta tokius failu formatus kaip SHP, DEM, (GEO)TIFF kas leidžia vartotojams be jokių problemų keistis informacija tarpusavyje.

Pradėsiu nuo rastrinių failų formatų. Rastras - tai duomenų rinkinys, kuris yra interpretuojamas kaip tinklelis, dar angliškai vadinamas GRID, kur kiekviename jo langelyje yra tam tikra reikšmė atvaizduojanti tos geografinės vietovės savybes. Jei tai paprasčiausia nuotrauka, ji saugo tame laukelyje atitinkamo objekto ar jo dalies matomą spalvą, jei tai, taip vadinamos, nuotolinės zondavimo technologijos (Remote Sensing) duomenys, tai greičiausiai spalva bus tam tikrų spektrų suminė reikšmė. Toks duomenų modelis dar naudojamas ir reljefo duomenims kaupti. Kadangi saugoti taškinę informacija trijų matmenų failuose būtų labai ne ekonomiška, buvo sugalvotas dar vienas variantas vadinamasis 2,5 dimensija. Tai dvimačiame masyve saugomos aukščių reikšmės, kitaip tariant tai rastras, kurio kiekvieno langelio reikšmė yra tos geografinės vietos aukštis. Tokio tipo duomenys saugomi DEM (Digital Elevation Model) failo formate, kuris buvo sukurtas JAV ir standartizuotas USGS organizacijos [13]. Tokio tipo duomenys apdorojančios programos turi sugebėti dirbti su trimate grafika.

Trimatės erdvės nereikalaujančius objektus paprastai išsaugo GeoTIFF formatu. Kadangi TIFF formatas dažniausiai sutinkamas išsaugant paprasčiausius paveiklėlius, tai dažnai galima rasti ir GIS duomenis taškinės grafikos failų formatuose kaip BMP, PNG ar JPEG. Kai kurie iš jų labai gerai glaudina taškinę informaciją, todėl yra labai patrauklūs, tačiau geografinis jų pririšimas nėra aprašytas, todėl toks variantas netinka. Tačiau GIS technologijose JPED ir PNG tikrai dažnai naudojami informacijos perdavimui, kaip pavyzdžiui WMS paslaugai.

GeoTIFF – tai TIFF failo formatas, kurio specifikacija yra atvira naudojimui, ir geografinės informacijos susiejimo mechanizmo mišinys. GeoTIFF standarto specifikaciją galima rasti http://www.remotesensing.org/geotiff/spec/geotiffhome.html adresu. TIFF – angl. Tag Image File Format kas lietuviškai reikštų tagais paremtą paveiksliuko kodavimo būdą. TIFF standarto specifikaciją galima rasti http://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf

adresu. TIFF yra universalus formatas rastriniams duomenims saugoti. Esmė ta kad paprastai failais turi fiksuoto ilgio jų turinį aprašančią antraštę, dažnai joje būna nurodomas ilgis, spalvų gylys, ir panaši informacija, kuri randama pagal jos adresą (angl. Offset), kas apriboja saugomų duomenų dydį baitais. TIFF turi atatinkamus laukus, tačiau jie yra ieškomi ne pagal konkretų duomenų adresą, o pagal tegą, todėl TIFF formatą nesunkiai galima pritaikyti saugoti reikiamo tipo informaciją. Jis efektyviai gali talpinti tiek 8, 16, 24 ar daugiau bitų spalvas, kas labai patogu gis technologijoms. Taip pat TIFF faile galima saugoti informaciją apie duomenų glaudinimo metodą, kas leidžia atsiradus naujiems metodams juos nesunkiai integruoti į TIFF failo formatą. Lygiai taip tegų pagalbą TIFF išsaugo ir geografinę informaciją [14]. GeoTIFF ne vienintelis tokių būdu rastrinius duomenis saugantis formatas, analogiškai duomenys yra saugomi ir ESRI TIFF failo formatuose. Kadangi TIFF specifikacija yra atvira tai yra ir laisvai platinamų įrankių konvertuoti GeoTIFF į ESRI TIFF ir atvirkščiai. Labai patogu ir tai jog specialių įrankių pagalba labai paprastai geografinę informaciją galima iš GeoTIFF išsaugoti paprastame tekstiniame faile, o apdorojus TIFF paveiksliuką kokia nors rastrinei grafikai skirta programa kaip „GIMP“ ar „Adobe Photoshop“, ją vėl įkelti į TIFF failą. Tai labai palengvina darbą su prastos kokybės skenuotais planšetais, ar kitais topografiniais žemėlapiais, kai dažnai tenka šalinti popieriaus susidėvėjimo žymes.

## Vektoriniai duomenys

Vektorinių duomenų saugojimo formatų lyderis yra kompanijos ESRI SHP formatas. Specifikacija adresu www.esri.com/library/whitepapers/pdfs/shapefile.pdf. SHP buvo sukurtas ESRI coverage formatui pakeisti ir duomenų tarp vartotojų keitimuisi palengvinti. SHP failai pasižymėjo savo paprastumu lyginant su ESRI coverage formatu, bei didesniu duomenų apdorojimo laiku. Taip jau susiklostė jog SHP nepakeitė ESRI coverage formato, jis ir toliau naudojamas, bet nuo šio formato paskelbimo laikų neatsirado populiaresnio, ir praktiškai visos atviro kodo programos turi šio failo formato palaikymo galimybes.

SHP (shapefile) formatas, palaiko tiek dvimatę tiek trimatę erdves. Pagrindinės figūros : taškas, linija, plotas. Šiame formate taip pat numatyta ir figūrų atributų susiejimo galimybė, bet nėra numatytas topologijos išsaugojimas. Norint atvaizduoti SHP faile saugoma figūrą, reikia dirbti su trim failais turinčiais plėtinius: shp, dbf, shx. Faile su plėtiniu SHP saugoma informacija apie figūrą. Ten nurodoma jos užimama vieta erdvėje(Angl. Bounding Box), jos unikalus numeris ir pačios figūros geografinės koordinatės. Figūros atributinė informacija saugoma DBF failo formate. Tai standartinis lentelių išsaugojimo formatas Windows operacinėje sistemoje. Pagal pigūros ir DBF unikalius numerius susiejama figūra su jos atributine informacija sąryšiu vienas prie vieno. Na o faile su plėtiniu SHX saugoma indeksavimo informacija, tiksliau ten randasi informacija apie figūros vietą SHP faile ir užimamą vietą erdvėje [15].

## Duomenų bazės

Duomenų bazių srityje pasirinkimas platesnis. Visos šiuo metu geometrijos paliakymą turinčios duomenų bazės vadovaujasi OGC( Open Geospatial Consorcium) 1999m. išleista “Simple feature specification for SQL“ (SFS) specifikacija, tačiau su tam tikrais nukrypimais. 2005 OGC organizacija išleido šios specifikacijos apibendrinimą, tiksliau buvo sujungtos kelios į viena vietą ir suderintos su ISO organizacija: „Simple Feature Access – Part 2 - SQL Option“ [16].

Kadangi ši specifikacija yra nauja ir nėra kol kas įgyvendinta, toliau apžvelgsiu senesnį jos variantą, kurį galima lengvai rasti OGC puslapyje http://www.opengeospatial.org/docs/99-049.pdf.

Šios specifikacijos tikslas - aprašyti būdą kaip geografinius vektorinius duomenis įrašyti į duomenų bazę. Geometrija bus aprašoma duomenų bazėje kaip paprasta lentelė (Geometrijos

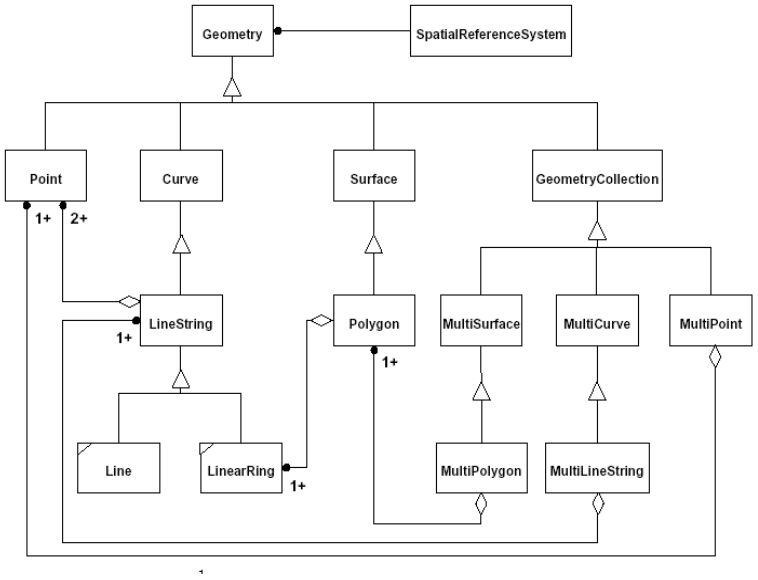
lentelė, arba *feature tabel*), o kiekviena geometrinė figūra toje lentelėje bus atvaizduojama atitinkama eilute. Atributai, susiję su saugoma geometrija bus saugomi atskiroje lentelėje, kuri tūrės išorinį raktą(FOREIGN KEY), kuris riš atributų lentelę su geometrijos lentele. Geometrijos lentelėje bus specialūs laukai laikantys informaciją apie geometrinę figūrą, jos koordinates. Ši specifikacija yra skirta SQL92 standartui, o kaip žinome SQL92 standarte nebuvo numatyta kaip saugoti geometrinius duomenis, taigi čia bus aprašyti du tokių duomenų saugojimo variantai numatyti OGC:

SQL92 – geometrijos duomenis bus saugomi atskiroje lentelėje, kurioje duomenis skaidomi į koordinates ir užrašomi atskiruose stulpeliuose, taip pat ši lentelė turi išorinį raktą rišantį ją su Geometrijos lentele.

SQL92 With geometry types- tai SQL standarto išplėtimas, palaikantis geometrinių figūrų tipą. Šiuo atveju geometrija telpa viename stulpelyje, kurio tipas yra geometrinių figūrų tipas (trumpiau – geometrija). Šis SQL išplėtimas yra ne vien naujo tipo įvedimas, bet kartu ir naujų funkcijų, skirtų darbui su šiuo tipo duomenimis numatymas.

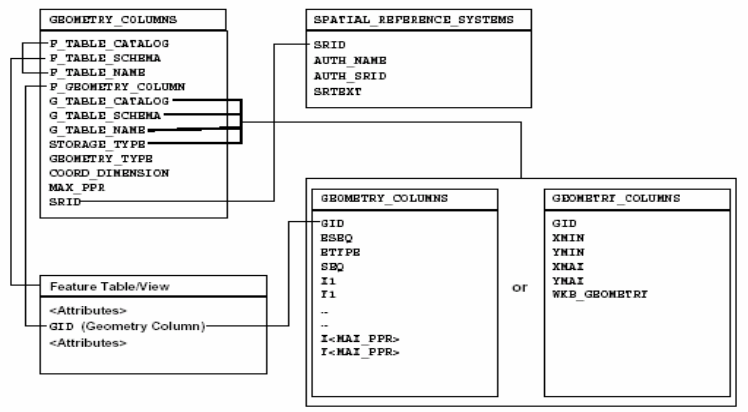
## Geometrijos modelis SQL92

Kokia sistema bus naudojama kuriant duomenų bazės struktūrą jau aprašiau, o kaip atrodys pats geometrinių figūrų skaidymas ir koks yra jų objektinis modelis aptarsiu dabar. 2.2.2.1 paveikslėlyje yra parodyta diagrama objektinio geometrijos modelio. Geometriją sudarys pagrindinės keturios klasės: Point (taškas), Curve (kreivė), Surface (plokštuma), Geometry collection (geometrinių figūrų rinkinys). Geometrinių figūrų rinkinys, yra ne bet kokių figūrų, o būtinai vieno tipo (pvz. : taškų linijų ar paviršių). Šie keturi tipai yra pagrindiniai ir paprasčiausi (simple), o iš jų yra išvedami taip vadinami sudėtiniai tipai (pvz.: LineString, ar MultiPoint). Kiekvienas objektas yra susijęs su klase „Spatial Reference System“ - išvertus reiškia koordinačių sistema, kuri aprašo objekto būvimo vietą ir dimensiją. Standarte taip pat nurodyta, kokius metodus bei metodų perduodamų ir grąžinamų parametrų tipus turi turėti kiekvienas diagramos objektas, tik įvykdžius visus šiuos reikalavimus bus pasiektas pilnas suderinamumas.




2.2.1 pav. OGC geometrijos objektinis modelis

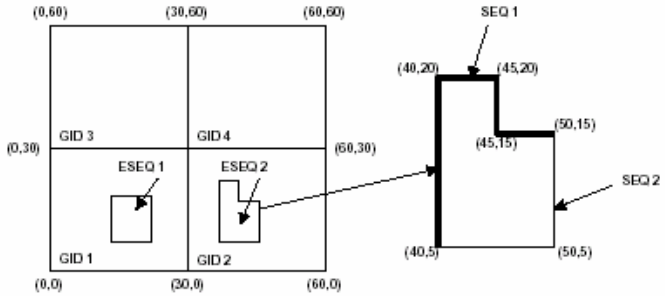
Norint šį objektinį modelį patalpinti į duomenų bazę pagal SQL92 standartą. Duomenų bazės struktūra turėtų būti kuriama pagal 2.2.2 paveiksle parodyta pavyzdį.




2.2.2. pav. Duomenų bazės lentelės

Kaip ir minėjau apie SQL92 standarto pritaikymą geometrijos saugojimui čia kiekviena figūra yra saugoma lentelėje „GEOMETRY_COLUMNS“ (meta duomenys), ir yra surišta su koordinačių sistemos lentele „SPATIAL_REFERENCE_SYSTEM“ ir taip pat su atributų lentele „Feature Table/View“. Kadangi SQL92 neturi specialaus lauko tipo geometrijai(geometrinės figūros koordinatėm) saugoti, yra sukuriama papildoma lentelė „GEOMETRY_COLUMNS“, jos tikslas yra saugoti savyje informaciją apie geometrijos tipą ir pačias koordinates. Ji nėra surišta su koordinačių sistemos lentele, nes jos saugomi duomenys nepriklauso nuo koordinačių sistemos. Ji saugo koordinates, o kokioje sistemoje jos bus naudojamos, jau kitos „meta duomenų“ lentelės darbas.

Koordinačių sugojimui SQL92 standarte vėl numatyti du variantai. Taip vadinamas normalizuotas geometrijos (normalyzed geometry) variantas ir binarinis (binary geometry). Normalizuotos geometrijos variantas - tai lentelė kiekvienam geometrijos tipui (taškui kreivei) su reikiamu laukų skaičiumi paprasčiausios figūros užrašymui ir su išoriniais raktais, skirtais geometrijos lentelei pririšti. Pavyzdžiui, paveikslėlyje 2.2.3 yra pavyzdys normalizuotos lentelės varianto, kur ETYPE laukas skirtas geometrijos tipui saugoti, kadangi geometriją gali sudaryti daugiau nei vienas lentelėje saugomas primityvas, tai turės turėti ir ESEQ lauką skirtą surišti du primityvus į vieną. Ir jei figūra sudaryta ne iš vieno primityvo, tai jų rašymo tvarka saugoma SEQ stulpelyje.




| GID | ESEQ | ETYPE | SEQ | X0 | Y0 | X1 | Y1 | X2 | Y2 | X3 | Y3 | X4 | Y4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 3 | 1 | 0 | 0 | 0 | 30 | 30 | 30 | 30 | 0 | 0 | 0 |
| 1 | 2 | 3 | 1 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 10 | 10 | 10 |
| 2 | 1 | 3 | 1 | 30 | 0 | 30 | 30 | 60 | 30 | 60 | 0 | 30 | 0 |
| 2 | 2 | 3 | 1 | 40 | 5 | 40 | 20 | 45 | 20 | 45 | 15 | 50 | 15 |
| 2 | 2 | 3 | 2 | 50 | 15 | 50 | 5 | 40 | 5 | Nil | Nil | Nil | Nil |
| 3 | 1 | 3 | 1 | 0 | 30 | 0 | 60 | 30 | 60 | 30 | 30 | 0 | 30 |
| 4 | 1 | 3 | 1 | 30 | 30 | 30 | 60 | 60 | 60 | 60 | 30 | 30 | 30 |

2.2.3 pav. Normalizuotos lentelės pavyzdys ir saugomos figūros

Analogiškai saugomi duomenys ir binariniame variante. Esmė yra ta, kad ten figūrą sudarantys primityvai saugomi specialiu formatu ir lentelėje saugomas binarinio tipo lauke. Figūros yra užkoduotos WKB Geometry formatu ( Well Known Binary Representation for Geometry). Šis formatas - tai struktūra, kurioje yra tam tikri laukai nurodantys, koks primityvas yra saugomas ir jo koordinatės. Binarinio tipo duomenų užrašymo lentelė pavaizduota paveikslėlyje 2.2.4.

| GID | XMIN | YMIN | XMAX | YMAX | GEOMETRY |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 30 | 30 | < WKBGeometry> |
| 2 | 30 | 0 | 60 | 30 | < WKBGeometry > |
| 3 | 0 | 30 | 30 | 60 | < WKBGeometry > |
| 4 | 30 | 30 | 60 | 60 | < WKBGeometry > |

2.2.4. pav. Binarinio formato lentelė, 2.2.3. pav figūroms

Kadangi SQL92 nėra geometrijos palaikymo tai jame nėra ir funkcijų skirtų su juo dirbti. Visai kitaip reikalai atrodo „SQL92 With Geometry Types“.

„SQL 92 With Geometry Types“ yra specialiai į leistinus SQL laukų tipus įtraukti geometrijos tipai: Geometry, Point, Curve, LineString, Surface, Polygon, GeometryCollection, MultiCurve, MultiLineString, MultiSurface, MultiPolygon, MultiPoint. Taigi kurdami lentelę jūs vietoj visų koordinačių laukų nurodote tiesiog kokio tipo geometrija, iš aukščiau paminėtų bus saugoma jūsų lentelėje. Tokiu būdų jūs sukursite lentelę vieno tipo geometrijai saugoti. Žiūrint į 2.2.2 pav. schemą jums nebereikės kurti papildomai „GEOMETRY_TYPES“ lentelių, taipogi šiame „SQL 92 With Geometry Types“ standarte numatytos ir funkcijos darbui su geometrijos tipais, kurios pagreitina tiek programuotojo darbą tiek pačios programos veikimą.

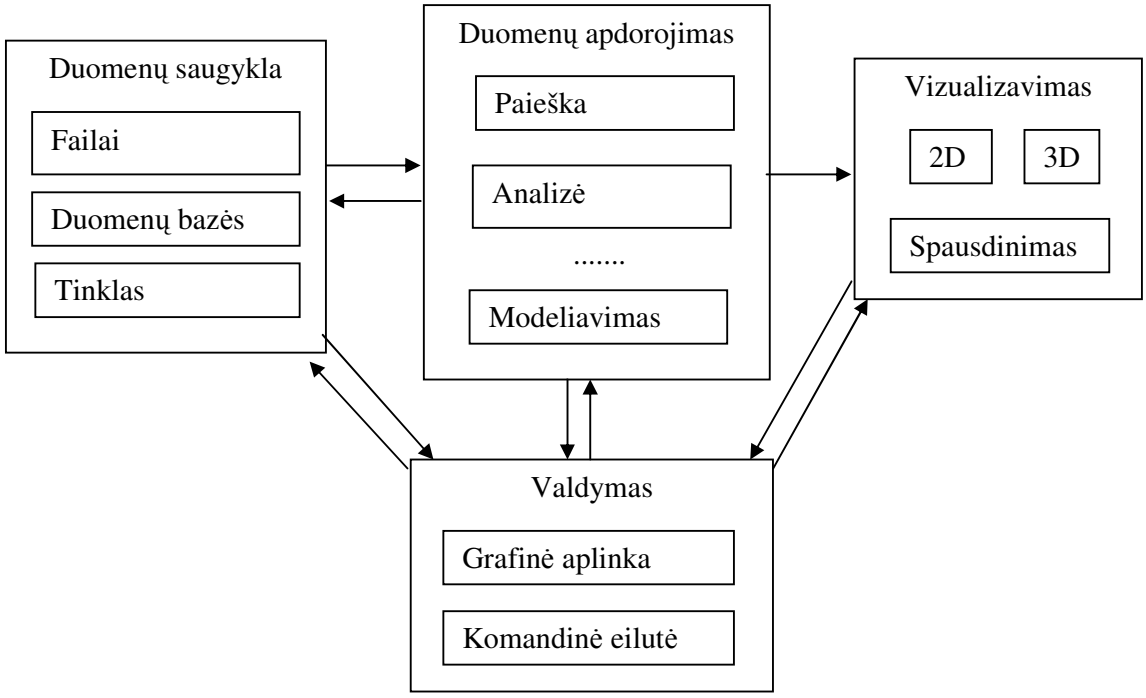
Deja ši specifikacija nenumato keleto labai svarbių dalykų, kaip pavyzdžiui, trijų matmenų dimensijos, kol kas specifikacijoje realizuota tik 2D erdvė, taip pat nenumatytas ir ketvirtasis matmuo „M“ dar vadinamas tiesiog „matmeniu“(angl. Measure). „Matmuo“ - tai geometrinės figūros reikšmė, pavyzdžiui kelio ženklo M matmuo galėtų būti kelio kilometras, kuriame jis randasi. Todėl kiekvienas gamintojas sukuria savo sprendimą spragoms užpildyti. Atviro kodo programinėje įrangoje pirmaujanti duomenų bazė yra PostgreSQL/PostGIS. PostgreSQL yra objektinė-reliacinė duomenų bazių valdymo sistema(ORDBVS), o PostGis – PostgresSQL funkcijų ir duomenų tipų rinkinys, suteikiantis duomenų bazei galimybę saugoti ir apdoroti GIS informaciją. PostgreSQL visiškai suderinama su minėta specifikacija, be to geometrinių domenų modelis yra išplėstas ir turi palaikymą tiek 3D matmenims tiek M matmeniui. Numatytas ir tipologijos palaikymas, nors kol kas ji tėra beta versijoje, kas reiškia apie programinio kodo nestabilumą. Tačiau PostGIS turi OGC specifikacijoje numatyta funkcijų rinkinį kuris leidžia pilnai dirbti su geometrine informacija. OGC specifikacija leidžia PostGIS naudoti kaip GIS duomenų saugojimo DB ne tik su atviro kodo programomis, bet ir su komercinėmis tokiomis kaip MAPINFO. Dar viena atviro kodo duomenų bazė, palyginus naujokė yra MySQL. Ji taip pat pilnai suderinama su OGC minėta specifikacija, tačiau neturi 3D ir M matmens palaikymo. Abi duomenų bazės palaiko R-Tree indeksavimą, padedantį pagreitinti geometrinių figūrų paiešką.

Minėtos specifikacijos nepanaikina skirtumų tarp duomenų bazių, žinoma kiekviena jų turi tiek savo specifinę prisijungimo tvarkyklę, savo technines savybes, kaip pavyzdžiui vienu metu dirbančių vartotojų skaičius, tačiau specifikacijoje aprašytos funkcijos ir lentelių išdėstymas suteikia galimybę skirtingose duomenų bazėse gauti vienodus rezultatus tuose pačiuose duomenų pavyzdžiuose visiškai nekeičiant pačios užklausos. Todėl tinkamai suprojektavus programinę įrangą, vienos duomenų bazės pakeitimas kita gali būti visiškai

nepastebimas, nereikalaujantis jokių modifikacijų programos dalių užsiimančių duomenų analizę.

## 2.3 Atviro kodo GIS architektūros tyrimas

Atviro kodo Gis programos nėra eilinės programos, jos sudėtingos vien dėl savo paskirties. Sunkumas tame jog jose labai daug specifinių terminų matavimo ir transformavimo algoritmų kurie yra specifiniai kitai mokslo sričiai – Geografijai. Dabar netgi kai kurios aukštosios mokyklos pradeda kurti naujas specialybes kaip Geoinformatika. Taip pat GIS reikalinga dvimatės, trimates grafikos, vektorinės algebros elementai. Tokiu būdu GIS programos kūrimas reikalauja įvairių sričių specialistų susibūrimo, kas nėra labai lengva, tuo labiau turint omeny, jog atviras kodas paprastai kuriamas laisvu laiku, jei specialiai jo nefinansuoja kokia kompanija. Nors atrodytu, jog atviras kodas niekaip negalėtų tokių sudėtingų programų sistemų sukurti visgi paradoksalu, jog atviro kodo licencija, atviri standartai, būtent ir suvienija įvairius kodo „gabalėlius“ į sudėtingas ir sėkmingai funkcionuojančias programas. Tai pakankamai aiškiai galima pastebėti nagrinėjant atviro kodo GIS programų struktūrą. Paimkime ne konkrečią programą, o šabloną, atvaizduojantį būdingas GIS programoms dalis.




2.3.1 Pav. Šabloninė GIS architektūra

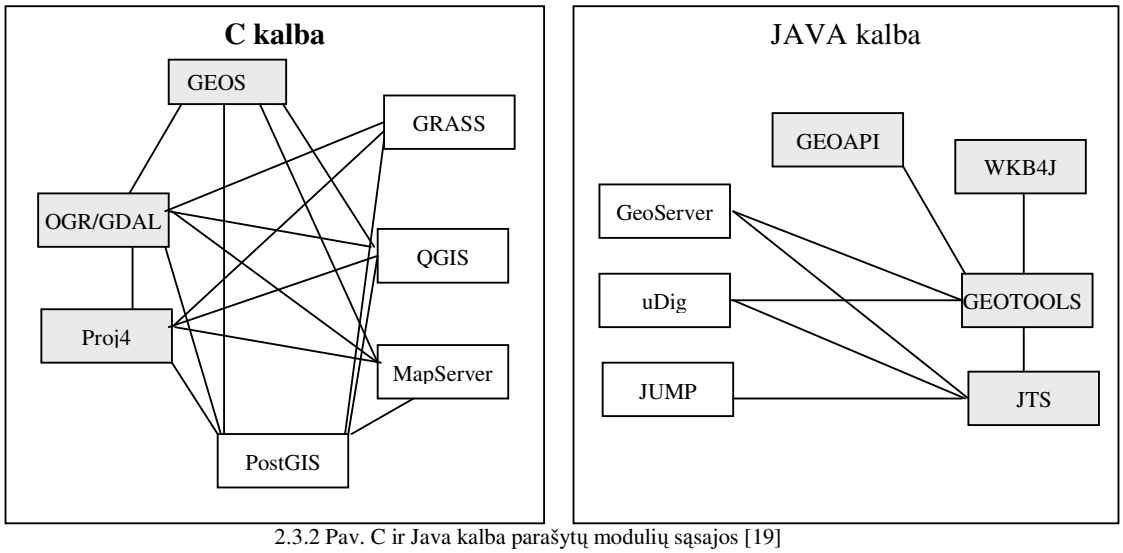
Paveikslėlyje 2.3.1 pavaizduota šabloninė GIS programos struktūra praktiškai nesiskiria nuo bet kurios kitos programos, nes taip pat turi duomenų saugojimo galimybę, jų apdorojimą ir rezultatų vizualizavimą bei programos eigos valdymą. Problema atsiranda tuomet, kai domenų saugojimo būdų yra labai daug. Nors aukščiau analizuotuose GIS formatuose yra tik keli – populiariausi, Gis duomenų formatų yra be galo daug (vien GDAL\OGR palaiko 59 skirtingus formatus [6]), bent jau kiekvienas komercinis uždaro kodo produktas turi po vieną. Todėl sekti visų formatų pasikeitimus patobulinimus, bei taisyti atsiradusias klaidas, kurias daro be išimčių visi, prireiktų jau nemažos komandos žmonių. Sekanti duomenų apdorojimo dalis dar sudėtingesnė. Šablone paminėjau tik pačias bendriausias operacijas atliekamas su GIS

duomenimis, tačiau realiai kiekviena pramonės ar verslo sritis turi savo GIS duomenų panaudojimo būdų bei tikslų. Todėl čia įvairių įrankių gali būti be galo daug. Pavyzdžiui, GRASS turi virš poros šimtų komandų skirtų įvairiems veiksmams su duomenimis, reikia turėti omenyje, jog jis skirtas apskritai GIS duomenų analizei, neatsižvelgiant į atskiras sritis kaip projektavimą ir t.t.[5]. Tuo labiau kiekvienas iš įrankių reikalaus, kiekvienos iš atstovaujamų sričių, specialisto žinių, kas uždavinį daro dar sudėtingesnį, nes paprastai ekonomikos, logistikos ar kitos srities specialistai greičiausiai nebus gabūs programuotojai, nes programavimas vėlgi atskira šaka reikalaujanti savų specialistų. O jei galų gale rasime reikiamus specialistus ar jie visi programuos viena programavimo kalba? Vizualizavimo bei Valdymo dalys papildomai turės dar vieną problemą – platformą. Jei duomenų saugojimo ir apdorojimo programas/bibliotekas parašėme, ar platforma, kuriai bibliotekos kurtos, turi grafinę aplinką? Ar grafikos bibliotekos suderinamos su Valdymo bibliotekomis. Taigi daug klaustukų, tad kaip juos įveikti ?! Kadangi ši problema ne nauja, internete nesunku rasti atviro kodo įvairių bibliotekų, daugiau ar mažiau tenkinančias iš šablone nurodytų elementų savybes. Jas sukūrė žmonės ar organizacijos savo laiku susidūrusios su jiems tuo metu aktualiomis problemomis, tų bibliotekų kodo paskelbimas atviru suteikia galimybes tuo pačiu keliu einantiems nebespręsti tų pačių problemų iš naujo, o ieškoti naujai atsiradusių problemų sprendimo būdo. Pavyzdžiui, jau nekarta minėta GRASS GIS informacijos analizės programa, geriausia savo srityje, tarp atviro kodo GIS analizės programų. Ji turi savo specifinį vektorinės informacijos išsaugojimo formatą. Žinome kad ji palaiko daugumą populiarių GIS duomenų saugojimo formatų tokiu kaip SHP(ESRI), S-57, SDTS, PostGIS, Oracle Spatial, Mapinfo mid/mif ir kitus. Tad kaip ji tai padaro. Esmė tame, kad GRASS naudoja šiems formatams apdoroti OGR/GDAL atviro kodo biblioteka. Kadaise OGR ir GDAL buvo atskiros bibliotekos, OGR – vektorinių duomenų formatų apdorojimo biblioteka, o GDAL – rastrinių duomenų apdorojimo biblioteka. Tokių būdu, įtraukdamas į savo architektūra šias bibliotekas GRASS projektas įgyja naujų formatų palaikymo galimybes. Be to galima teigti, jog GRASS projekte dirba dar ir OGR/GDAL biblioteką kuriantys žmonės, nes taisydami klaidas, bei tobulindami OGR/GDAL bibliotekų kodą, jie tuo pačiu tobulina ir GRASS programą. Taigi štai pavyzdys, kaip atviras kodas leidžia apjungti skirtingas sritis bei specialistus. Analogiškai pav. 2.3.1 šablone „Duomenų saugykla“ dirba, su minimais elementais „Duomenų analize“, „Atvaizdavimu“ bei „Valdymu“. Žinoma GRASS atvaizdavimui bei valdymui vėlgi naudoja papildomas bibliotekas, kurios analogiškai yra susietos. Tarkim, jog dėl tam tikrų priežasčių OGR/GDAL kūrėjai nusprendė nebetęsti šios bibliotekos palaikymo, toks sprendimas įtakos GRASS projektą minimaliai, kadangi atviro kodo licencija garantuoją OGR/GDAL bibliotekos kodo prieinamumą ir teisę jį modifikuoti, vienintelis sunkumas, papildomos jėgos, reikalingos taisyti pastebėtas bėdas tariamai apleistoje OGR/GDAL bibliotekoje, tuo atveju jei jų atsirastų. Jei šioje situacijoje kodas būtų uždaras, GRASS projektui tektų ieškoti kitos panašios bibliotekos, smarkiai taisyti savo kodą, ir po truputį „kraustytis“ iš senstančios bibliotekos.

Dar viena labai svarbi atviro kodo programų architektūros dalis yra integruota įskiepių sistemą. Praktiškai be jos GIS programa pasmerkta pasenti arba labai greitai prarasti savo paskirtį. Nors GIS kaip kartografinis įrankis gyvuoja jau daugybę metų ir būtent tokia buvo jo pradinė paskirtis, šiandien į tokią programą, turinčią tik viena funkciją niekas nekreiptų dėmesio. Esmė ta, kad GIS potencialas labai gretai buvo suprastas ir įvairios interesų grupės ėmė matyti jame savo verslo perspektyvą. Kaip akivaizdžiausią ir kiekvieną dieną vartojamą, galima būtų paminėti „GOOGLE“. Apie GOOGLE galvoja kaip apie paieškos portalą „www.google.com“ . Tačiau dabar GOOGLE gali surasti ne tik ieškoma informaciją tinklapyje, pagal jūsų nurodytą raktinį žodį, bet ir pagal jūsų norimą vietovę. Pakankamai neseniai atsirado naujas GOOGLE produktas „GoogleEarth“ būtent suteikiantis tokią galimybę. Tai tik vienas iš pavyzdžių kaip verslas gali savo naudai pajungti GIS technines galimybes. Tačiau su minėta programa, jūs kitose srityse nepadirbėsite, nes ji turi savo apibrėžtą paskirtį.

Tokių programų tiek atviro ar neatviro kodo laisvai prieinamų galima rasti be galo daug tačiau plačiau pritaikomos tik tos, kurios išplečiamos. Žinoma atviro kodo galimybės suteikia galimybe programą taisyti ir įdėti reikiamas papildomas funkcijas. Tačiau tam reikės parsisiųsti visą programos kodą, kas rimtesniuose projektuose bus pakankamai ne menki kiekiai, taip pat dokumentacijoje rasti specifines vietas, jas pataisyti, perkompiliuoti visą kodą ir programą paruoštą darbui. Žinoma tai tik idealiame variante. Realiai jūs padarysite klaidų, jas teks taisyti, perkompiliuoti kodą. Didelėse programose kodo apimtis gali būti pakankamai nemaža, kas smarkiai lėtins jūsų darbą. O blogiausia, jog išėjus naujai programos versijai jums teks pakartoti procesą. Tai nėra labai patogu. Daug paprasčiau yra perkompiliuoti kodą susijusį tik su jūsų norimu įdiegti funkcionalumu. Būtent šiam tikslui ir skirta įskiepių (plug-in) technologija. Šis variantas patogus ir tuo, kad pasikeitus programos versijai jums jau sukompiliuotą objektinį kodą tereiks pajungti prie programos, jo iš naujo neperkompiliuojant. Tačiau čia yra vienas trūkumas, nėra vieningos technologijos. Windows operacinėje sistemoje dažniausiai naudojama OLE/COM arba COM+ technologija, kuri yra standartizuota, bet labai stipriai susijusi su operacine sistema. Nuo platformos nepriklausoma analogiška sistema yra CORBA, tačiau ji reikalauja papildomo serviso, kad galėtu funkcionuoti, kas papildomai apkrauna operacinę sistemą. Todėl ypač smulkios ir nuo operacinės sistemos nepriklausomos programos linkę naudoti savo specifinę sistemą, pritaikyta būtent jų poreikiams. Tai nėra labai blogai, kadangi paprastai, priklausomai nuo programos, jos nebūna sudėtingos ir dažniausiai sukurti įskiepį daug paprasčiau nei analogiškų pakeitimų pasiekti keičiant pačios programos kodą. Tačiau pernešti vieną įskiepį iš vienos programos į kitą nėra jokių vilčių. Kad ir kaip ten būtų, gera programos architektūra ir įskiepių sistema jums gali suteikti galimybes taip pakeisti programą, jog, iš pirmo žvilgsnio, bus sunku atpažinti ir pritaikyti ją tokioms sritis, kurių pirmieji programos kūrėjai net neįsivaizdavo.

Nužvelgiant atviro kodo GIS pasaulį per programavimo kalbų prizmę iškarto matyti dvi atskiros GIS šakos, kurios siekia vienodų tikslų. Tos šakos - tai dvejomis skirtingomis kalbomis parašytos kodo grupes C ir JAVA. C kalbos ir C++ parašytos programos yra brandesnės, jose mažai klaidų, kadangi jos atsirado anksčiau, anksčiau atsirado jų GIS projektai didesnės ir jų programuotojų bendruomenės. JAVA bendruomenė turi mažiau projektų, juose daugiau klaidų, tačiau jie labiau koncentruoti, kiekvienas projektas JAVA GIS bendruomenėje turi savo vietą. Kitoje schemoje pavaizdavau pagrindinių programų apžvelgtų ankstesnėje darbo dalyje tarpusavio sąsajas [19].




2.3.2 Pav. C ir Java kalba parašytų modulių sąsajos [19]

2.3.2 paveikslėlyje schemoje pilkiau pažymėti stačiakampiai kurie vaizduoja bibliotekas, o šviesūs stačiakampiai – programas. Iš pirmo žvilgsnio matyti, jog C kalbos pusėje schema daug painesnė. Ir linijų joje daugiau. Taip yra todėl jog kiekviena programa, nors ir atliekanti panašias funkcijas buvo kuriama kaip atskiras produktas, ir naudojo kiekvieną biblioteką tik savo reikmėm, ir tik sau specifiniu būdu, todėl kitiems, kitų programų kūrėjams yra sunku pasinaudoti tokiu kodu, ir žymiai lengviau jį iš naujo perrašyti. Tuo tarpu Java pusėje sąsajų daug mažiau, nes dauguma jų apjungia GeoTools biblioteka, suteikdama ja besinaudojančioms programoms vienodą klasių rinkinį, tokiu būdu programos nesunkiai gali įsisavinti reikiamas kodo, o schemoje panaikinti sąsajas. Ši schema dar patogi tuo jog puikiai atvaizduoja šiame darbe minimų ir nagrinėjamų programų bei bibliotekų tarpusavio sąsajas. Šios schemos įsiminimas lengviau padės ne tik suprasti tolimesnį tekstą, bet ir lengviau orientuotis atviro kodo GIS pasaulyje.Iš šios schemos labai aiškiai galima suvokti ir aukščiau minėtų standartų naudą. Pavyzdžiui uDig ir GRASS parašytos visiškai skirtingom kalbom tačiau puikiai supranta SHP ir GeoTIFF formatus, leidžiančius tarpusavyje keistis duomenimis. PostGIS visom čia išvardintoms programoms nepriklausomai nuo jų kalbos suteikia vienodas galimybes dirbti su duomenų bazėje saugomais duomenimis vien dėl OGC SFS specifikacijos.

## 2.4 Išvados

Išanalizavę atviro kodo GIS srityje pirmaujančius standartus ir formatus galime padaryti vieningą išvadą, jog renkantis programinę įrangą, savo GIS sistemai būtina sąlyga yra sugebėjimas dirbti su šiais duomenų formatais rastrinių duomenų - GeoTIFF formato palaikymas, vektoriniams duomenims – SHP failo formatas. Pageidautina, kad programa galėtų dirbti ir su trimačiais matmenimis, bei papildomai apdoroti DEM failo formatą. Tačiau daugumai sričių tai nėra būtina. Trečią matmenį dažniausiai naudoja su hidrografija dirbantys žmonės, taip pat tai naudinga miesto architektams, nustatyti naujų daugiaaukščių statinių įtaką kraštovaizdžiui.

Jei jūsų sistemoje vienu metu dirbs keletas žmonių, būtiną papildomai vienos iš duomenų valdymo sistemų palaikymas Postgres/PostGIS, arba MySQL, žinoma greičiausiai ateityje jų atsiras ir daugiau todėl būtina sąlyga jas renkantis - OGC „Simple Feature Specification for SQL“ palaikymas.

Taip pat GIS programai būtina turėti kokią nors įskiepius palaikančią sistemą. Tik tokia programa gali būti visiškai pritaikoma iškilusiems uždaviniams spręsti. Grafinė aplinka, kalba, kuria programa parašyta, bei platforma kuriai ji parašyta, gali būti pasirenkama pagal įvairius kitus kriterijus, pavyzdžiui jūsų naudojama operacinę sistemą, savo „religinį įsitikinimą“ programavimo kalbų atžvilgiu ir t.t.

# 3 ATVIRO KODO GIS TYRIMAS

## 3.1 Įvadas

Išnagrinėjus sudedamąsias GIS programų dalis būtina išanalizuoti jų sudaromas sistemas, peržvelgti konkrečius panaudojimo būdus. Tam tikslui paimsime konkretų pavyzdį, pabandysime įvertinti sistemos galimybes, perspektyvas, pasižiūrėti programos technines ribas.

Konkrečiai bus nagrinėjama „GeoTools2“ biblioteka, bei jos pagrindu sukurta programa darbui su GIS - „uDig“(User-friendly Desktop Interface GIS). Šios programos galia slepiasi jos architektūroje, tiksliau Ecplipse platformoje, kas uDig leidžia vadinti ne šiaip paprasta programa, o GIS platforma. Nors šiai programai truputi daugiau nei metai, jos pagrindu jau yra sukurtų naujų programų, kaip pavyzdį galiu paminėti “Diva-GIS”. Tai buvo atlikta būtent naudojant įskiepių sistemą, būtent tokiam panaudojimui ši platforma ir yra skirta.

## 3.2 Atviro kodo GIS vystymui skirti įrankiai.

Įrankių pasirinkimas pirmiausiai priklauso nuo pačios programavimo kalbos pasirinkimo, nes dažniausiai, vienas įrankis būna kuriamas tik konkrečiai kalbai, universalių įrankių dažniausiai galima rasti tik programinės įrangos projektavimo srityje. Kadangi GIS yra labai plati sritis, todėl programavimo kalba privalo turėti galimybę dirbti su įvairiomis duomenų bazėmis, palaikyti kaip galima daugiau platformų, našiai dirbti. Tokius reikalavimus puikiai atitinka keturios programavimo kalbos C, C++, JAVA ir C#. Deja C kalba didelėms programoms netinka, žinoma galima kurti, tačiau ji nepalaiko objektinės paradigmos, kas programos architektūrą padarys pakankamai sudėtinga. Kadangi ši kalba labai efektyvi ir labai senai sukurta, jos pagalba parašytų programų be galo daug, tačiau šiuolaikiniam programavimui, ypač kalbant apie sudėtingas struktūras, su ja rašyti programą būtų sudėtinga, bei užimtų žymiai daugiau laiko nei su sekančiomis. Taip pat šia kalba sukurtas kodas yra „pririštas“ prie platformos, todėl norint savo programą pernešti ant kitos platformos gali tekti ją smarkiai pakoreguoti

C++ objektinė programavimo kalba, savo našumu iš išvardintųjų nusileidžianti tik C kalbai. Ji turi be galo daug įvairių bibliotekų, darbui su ja skirtų įrankių, kompiliatorių tiek atviro kodo tiek komercinių. Trūkumas galėtų būti tik jokios integruoto kodo dokumentavimo galimybės nebūvimas, bei prijungiamų bibliotekų direktyva „#include“, kuri gali nuvesti į aklavietę įterpiant pakankamai didelius papildomus failus, ypač nepatyrusius programuotojus, ji kaip ir jos pirmtakė taip pat yra „pririšta“ prie platformos.

JAVA – objektinė programavimo kalba, sukurta kompanijos „SUN“ kartu su JAVA programavimo kalba buvo ir sukurtos ir bibliotekos reikalingos darbui, todėl kartais kalbant apie JAVA, kartu turima omenyje ir pati biblioteka. Bibliotekoje yra visos funkcijos reikalingos pilnavertei programinei įrangai kurti tiek skirtai pavieniams vartotojams, tiek aptarnauti didžiuliams vartotojų kiekiams. Pagrindinis, tačiau ne vienintelis, JAVA programavimo kalbos privalumas yra jos nepriklausomumas nuo platformos. JAVA laikoma pilnai objektine Programavimo kalba, kadangi pas ją, viskas realizuota objektinėje paradigmoje, visa klasių biblioteka yra paveldėta iš vienos bendros klasės, visi net ir paprasčiausi duomenų tipai yra objektai. Taip pat JAVA turi standartizuotą ir unikalią dokumentacijos sistemą „JavaDoc“, todėl kodo dokumentaciją, galima rašyti kartu su programos kodu. Specialūs įrankiai padeda sukompiliuoti dokumentaciją į HTML failus. JAVA kalboje realizuota ir gijų palaikymas, bei specialūs metodai jas valdyti ir kurti. Kadangi JAVA parašytas kodas yra kompiliuojamas ne į išeinamąjį kodą, suprantamą operacinei sistemai, o į batinį kodą, kuris specialaus interpretatoriaus yra vykdomas, JAVA negali prilygti tokiom kalboms kaip C ar C++, todėl labai daug „legendų“ sklando apie JAVA kalbos tariamą lėtaeigiškumą. Dažniausiai šios legendos remiasi labai paprastu pavyzdžiu, JAVA SWING bibliotekos vartotojo sąsajos veikimo sparta. Deja čia būtina pastebėti, jog SWING yra visiškai nuo platformos nepriklausoma biblioteka, kuri leidžia programai ant bet kokios platformos atrodyti ir veikti visiškai vienodai. Tam tikslui ji naudoją operacinių sistemų GUI bibliotekos manipuliacijų galimybes pikselių lygmenį [18] ir kuria visus vartotojo aplinkos komponentus nuo nulio, tikiu būdu kontroliuodama savo komponentus visiškai ir neleisdama operacinei sistemai kištis į jų valdymą ar išvaizdą, bet viso to kaina – didelis našumo praradimas, kas daugelį pradedančių programuotojų atbaido nuo JAVA technologijos.

C# palyginti naujokė tarp mano išvardintų programavimo kalbų, tačiau ją suskurusi „Microsoft“ kompanija jai žada nuostabią ateitį. Iš tiesų ji savo paskirtimi ir veikimo idėja labai primena JAVA. Kaip JAVA veikia ant savo virtualios mašinos JVM, taip C# veikia naudodamas savo veikimo aplinką - .NET platformą . Microsoft žada jog .Net kaip ir JAVA, suteiks galimybe rašyti programas nepriklausomai nuo platformos. Kol kas deja .NET platformai galima

rašyti programas nepriklausomai nuo programavimo kalbos, taip didelis pliusas jog galų gale Windows turi vieningą klasių hierarchiją, ir galima sakyti jog laikai kai konkrečios funkcijos parametrų ir aprašymų reikia ieškoti dokumentacijoje, o po to nesusipainioti įvairiose Windows objektų rodyklėse, laikai praėjo. Tačiau iš esmės JAVA ir C# savo galimybėmis yra panašios.

Kalbant apie sukompiliuoto kodo našumą turbūt niekas nesiginčys dėl C ir C++ našumo, tačiau JAVA ir C# našumo aptarimas dažnai sukelia dideles diskusijas, kurios paprastai nepriveda prie vieningos nuomonės dėl labai paprastos priežasties, tiksliai įvertinti kalbų našumą neįmanomą, nes specifinės sritys turi ne vienodus reikalavimus, todėl kalbos tyrimo rezultatai labai svyruoją priklausoma nuo testuojamo algoritmo savitumo.

Kadangi GIS yra labai įvairi sritis, todėl akivaizdu, jog sunku nuspėti, kokioje platformoje teks dirbti programinei įrangai, taip pat kokius modulius ir kokia kalba parašytus teks prijungti arba integruoti. Šias sąlygas galėtų atitikti abi kalbos C# ir JAVA, nors kol kas C# krosplatformiškumas (pernešamumas) yra tik pažadėtas, tačiau ne jis nulėmė pasirinkimą. Pasirinkimą nulėmė jau egzistuojančios internete bibliotekos, kurios parašytos JAVA kalba. Iš to išplaukia ir pačios programos platforma Elipse, kurios nauda GIS programai dar tik aptarsiu.

## *3.3 Eclipse – Atviro kodo platforma su intarpais.*

Eclipse - programavimo aplinka sukurta atviro kodo pagrindu. Ji universali programavimo aplinka, turinti unikalią įskiepių sistemą. 2001 IBM kompanija viešam naudojimui pateikė įrankių už 40 milijonų JAV dolerių. Pradedant šiais įrankiais, kai kurie Integruotų programavimo aplinkų (IDE –Integrated Development Environment) kūrėjai susibūrė į organizaciją „Eclipse Foundation Inc.“. Eclipse turėjo pati universalią kūrimo aplinka. Terminu „kūrimo aplinka“ noriu pabrėžti, jog ji nebuvo kuriama konkrečiam tikslui, ji tiesiog buvo numatyta kaip universali, įvairioms paskirtims tinkanti platforma. Eclipse - tai tėra tuščia dėžė į kuria galima, naudojant jos įskiepių sistemą įdėti bet kokį reikiamą funkcionalumą. Pavyzdžiui, Eclipse galima pritaikyti tiek JAVA tiek C++ programavimui, arba darbui su CORBA objektais. Standartiniame Eclipse pakete jau būna įdėta virš 80 įvairių įskiepių, toliau reikiamus gali susidėti kiekvienas vartotojas pagal poreikius. IBM dabar naudoja Eclipse kaip pradinį komponentą tokiems savo produktams kaip „WebSphere“ ir „Rational Software“ [9] .

Eclipse Foundation organizacija Eclipse kūrimo kryptis skirsto į projektus ir subprojektus. Taigi Eclipse Foundation turi kelis projektus. Be „Eclipse“ projekto dar yra „Eclipse Tools“, ir dar keli. Tačiau svarbiausias ir yra „Eclipse“projektas. „Eclipse“ projektas turi tris pagrindinius subprojektus, „Platform“, „Java Development Tools“ ir „Plug-in Development“. „Platform“ – arba platformos subprojektas yra atsakingas už pačias esmines Eclipse funkcijas. Jis savaime skaidosi dar į smulkesnius subrojektus, vienas iš jų būtų „core“ arba branduolys, jis atsakingas už pačias Eclipse esmę sudarančias funkcijas, kaip Eclipse startavimas, ir stabdymas, įskiepių suradimą, paleidimą valdymą ir taip toliau. Kiti platformos subprojektai atsakingi už Eclipse išvaizdą. Eclipse naudoja grafinei sąsajai kurti SWT/JFace biblioteką.

„Java Development Tools“ – arba išvertus Java kodo kūrimo įrankiai yra skirti būtent tam kaip ir parašyta. Daugelis apie Eclipse galvoja kaip apie IDE skirtą JAVA‘ai, deja jie klysta, Eclipse nėra 100% JAVA sukurta platforma, ir tai nėra vienintelė jos paskirtis, kaip minėjau. Praktiškai Eclipse nėra priklausoma nuo programavimo kalbos ir neturi tokios vienintelės. Šis subprojektas yra tik vienas iš kelių subprojektų, jis tiesiog yra įdėtas pagal nutylėjimą.

„Plug-in Development environment“(PDE) – tai taip pat vienas iš subprojektų . Eclipse ne daugiau kaip rėmas laikantis įvairius įskiepius, kiekvienas įskiepis platformai suteikia nedidelį funkcionalumą, o kartu visus sudėjus Eclipse tampa laibai išvystytu programavimo įrankiu. Ir žinoma, kadangi įskiepiai yra dalykas, kuris kuriamas programavimo įrankių pagalba

tai šio subprojekto paskirtis ir yra suteikti Eclipse platformos pagalba tobulinti ją pačią, bei pritaikyti ją savom reikmėms. Turbūt nebereikia kalbėti, kodėl būtent JAVA kalba pasirinkta šiam funkcionalumui įgyvendinti.

Dėl šių priežasčių Eclipse pagrindu sukurta uDig programa yra verta platformos vardo, kadangi ji gali būti pritaikoma praktiškai bet kokiam uždaviniui. Kaip nedidelį pavyzdį to kaip galima pasinaudoti šios platformos funkcionalumu, sukūriau įskiepį. Jo kūrimo ypatybės bei savybės yra smulkiau aprašytos priede Nr.2 „Įskiepio kūrimas".

Kaip jau minėjau, Eclipse savo grafinei aplinkai naudoja SWT/JFace biblioteką. Taip pat jau buvau minėjas, kad SWING grafinė aplinka yra labai lėta, nes ji praktiškai nesinaudoja jokiomis operacinės sistemos teikiamomis šiai sričiai funkcijomis. Viskas ko SWING iš operacinės sistemos reikia, tai galimybė nubrėžti ekrane tašką, todėl ji yra labai lėta ir labai nepriklausoma. Tuo tarpu daugeliui programų tokios spartos nepakanka, šiai problemai spręsti ir buvo sukurtas SWT (Standart Windows Toolkit) biblioteka, kuri iš operacinės prašo jau sukurti konkretų valdymo elementą, tokį kaip mygtuką ar teksto lauką. Toks veikimo principas SWT suteikia žymiai didesnę veikimo spartą lyginant su SWING, tačiau SWT prarandą dalį savo nepriklausomumo. SWT sukurta grafinė aplinka turės specifinę operacinės sistemos išvaizdą, be to teks kartu su JAVA kodu naudoti ir specifinį, bibliotekos tarpininkės kodą, kuris priklausomai nuo operacinės sistemos skirsis. Tačiau tai paslėpta nuo vartotojo akių, programuodamas vartotojas matys vienodą JAVA bibliotekų rinkiny savo programoje ir žinoma, pačios programos rašymas nesiskiria, paprasčiausiai parsisiųsdamas JAVA darbo aplinką vartotojas turi rinktis pagal savo operacinę sistemą.

## 3.4 Atviro kodo GIS tyrimas ir testavimas

Tam, kad normaliai dirbti su GIS duomenimis reikia labai ne daug, minimaliam darbui užtenka galimybės taisyti ir skaityti duomenis kokioje nors laikmenoje. Tačiau šiais laikais, kai kompiuteriniu tinklu gali pasigirti ne viena įmonė, atsiranda galimybė vienu metu dirbti su duomenimis ne vienam žmogui. Tačiau deja ne visos programos tai gali. Konkrečiu mano pavyzdžiu įmonė turi GIS duomenis, kurie yra laikomi ir apdorojami ESRI kompanijos įrankiu ArcView 8.2. Turbūt daugelis, kurie yra susidūrę su GIS yra susidūrę ir bent su vienu šios kompanijos įrankiu ar kokiu kitu produktu. Lietuvoje ji labai gerai išreklamuota ir jos įrankiai labai paplitę. Nieko prieš juos neturiu, tikrai puiki programinė įranga, tačiau ne už dyka. Tai būtu dar ne pati didžiausia bėda. Taigi problemos prasideda, kai reikia duomenimis dalintis. Mano atveju visi duomenys yra saugomi ArcView GeoDatabase formatu.

GeoDatabase - tai yra mdb failas, MS Access duomenų bazė, tačiau turinti specialia lentelių struktūrą, bei specialų būdą geometrinių figūrų saugojimui duomenų bazės lentelės lauke. Kadangi ArcView nėra atviro kodo programa, šie formatai ir struktūros nedokumentuoti. Apskritai ESRI ne pirmoji naudojanti šį variantą, tačiau būtent jos programos išpopuliarėjo.

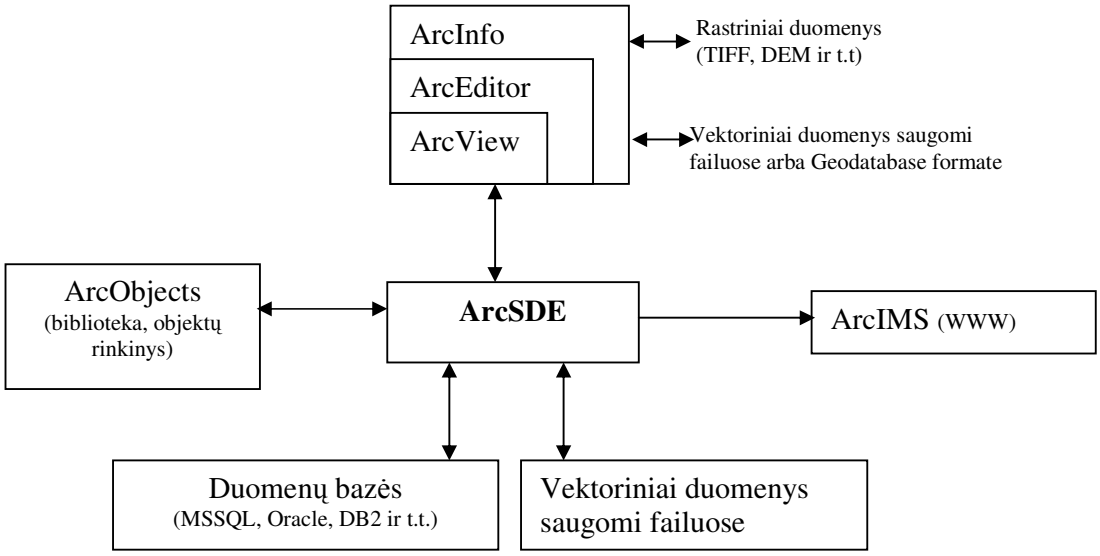
Iš MS Access duomenų bazės pajungti tam tikras lenteles ir sukurti kokius nors programavimo klientus būtų nesudėtinga, tuo labiau, jog duomenis reikia tik pažiūrėti, tačiau kokiu būdu geografiniai duomenys saugomi duomenų bazės laukuose nėra dokumentuota, o su įvairių įrankių pagalba nagrinėti GeoDatabase formatus draudžia licencija. Taigi lieka tik vienas būdas - duomenų eksportas į labai populiarų jau minėta formatą SHP. Tačiau čia atsiranda naujos problemos, duomenų bazėje saugomi duomenys yra pritaikyti reliaciniam duomenų bazės modeliui, o tam, kad juos būtų galima atvaizduoti vieno failo pagalba reikia įvairias lenteles sujungti į krūvą.

Įsivaizduokite paprasta stulpą gatvėje, ant jo gali būti dedami kelio ženklai, kabinami elektros laidai, kabinama reklama, apšvietimo žibintai. Duomenų bazėje tokį stulpą atvaizduotų vienas taškas, jo atributuose būtų įvairūs duomenys apie jo fizines charakteristikas. Tuo tarp informacija apie ant stulpą kabančius papildomus dalykus tūrėtų būti saugoma įvairiose atskirose lentelėse. Tokio modelio reikalauja reliacinis duomenų bazių modelis, ArcView su surištom

lentelėm puikiai dirba ir turi priemones įvairiem atributam ištraukti integruoto VBA kodo pagalba. Taip pat tokiam reliaciniam modelyje daug taupiau ir tvarkingiau išsidėsto informacija, bei paprasčiau atlikti įvairias užklausas. Tačiau šis patogumas virsta problema kai viską reikia eksportuoti į failus, nes kiekvieną tokią lentelę reikia paversti atskiru sluoksniu, ošiam dalykui specialiu įrankiu nėra, tenka darbuotis dalinai ArcView įrankių pagalba, dalinai SQL Access duomenų bazėje.

Kitas klausimas, yra duomenų saugumas. Operacinės saugumo idėja labai paprasta, jei leidai failą skaityti, tai leidai jį ir kopijuoti, o kas nori leisti savo GIS informaciją kopijuoti? Todėl vėlgi tenka naudoti įvairias gudrybes tam, kad apeiti šį operacinės sistemos funkcionalumą, leisti skaityti, bet ne kopijuoti.

Pati problema ESRI produktų, ypač mažoms įmonėms, kurios nori leisti dirbti labai mažam asmenų kiekiui vienu metu, slypi jos produktų sumanyme. Paimkime ir panagrinėkime schemą, kurioje pavaizduotą kaip ESRI kompanijos produktai keičiasi duomenimis.




3.3.1 Pav. ESRI kompanijos produktų tarpusavio sąsajos

Iš schemos matyti, jog kiekvienas produktas skirtas darbui darbo vietoje, toks kaip ArcView, ArcEditor, ar ArcInfo gali dirbti su failais ar GeoDatabase, bet tam kad jie bendradarbiautų reikalingas tarpininkas ArcSDE. Šioje architektūroje būtent jis atlieka duomenų mainų priežiūrą. Nors jis gali atrodyti visiškai nereikalingas, tačiau taip nėra, nes jis dar buvo kurtas tuomet kai nebuvo duomenų bazių palaikančių vektorinius duomenis, tiksliau tuo metu buvo vienintelė Oracle, o jis užtikrino tokį funkcionalumą dar kelioms populiarioms kaip DB2 ir MSSQL. Taip pat ArcSDE gali organizuoti kelių vartotojų darbą su failais vienu metu, tarsi klientai dirbtų su duomenų baze o ne failais. Taip pat ArcSDE suteikia galimybe pajungti į savo sistemą papildomus modulius tokius kaip ArcIMS skirtus duomenų eksportui www protokolu. Tai labai patogu norint paviešinti duomenis, o ArcObjects komponentų ir klasių biblioteka, leis greitai sukurti individualiam poreikiui skirtas aplikacijas ar ActiveX komponentus. ArcObject komponentai su duomenimis elgsis analogiškai kaip ir ArcView ar kitas produktas tai yra, kol jis dirbs vienas, viskas tvarkoje tačiau jei jums prireiks naudotis duomenų baze, ar dalintis keliais failais ar GeoDatabase duomenimis , tuomet jūs būsite priverstas naudotis ArcSDE. Net nežinant tikslių kainų, toks programinės įrangos kiekis, norint pajungti dvi ar tris darbo vietas darbui, mano nuomone, yra per daug ir per brangu. Vien duomenų bazei, jei nuspręsite ja naudotis kaip duomenų saugykla, teks išleisti nemažai pinigų, nes ArcSDE nepalaiko nei vienos

atviro kodo duomenų bazės, motyvuodami tuo jog jos nėra patikimos ir keltų rimtą grėsmę ArcSDE patikimumui.

Susidūręs su tokiais sunkumais pirmiausia bandžiau spręsti problemą pasitelkiant ArcView integruota VBA (Visual Basic for Applications). Mano tikslas buvo priverst ArcView piešti sluoksnius saugomus lentelėse atviro kodo duomenų bazėje PostgreSQL, kurios ArcView pagal nutylėjimą nepalaiko. Pasinaudodamas objektinio programavimo savybe, paveldėjimu, bandžiau pakeisti programoje atvaizduojamo sluoksnio objektą, savo sukurtu ir atvaizduojančiu duomenis iš PostgreSQL. Tokiu būdu siekiant, jog programa, leistų taikyti visas joje esančias funkcijas ir įrankius sukurtam objektui. Tačiau greitai paaiškėjo, jog pateikiamos dokumentacijos per mažą. Dokumentacijoje pateiktos informacijos pakanka tik esamų objektų funkcionalumui padidinti, todėl norint įdiegti sluoksnį kuriame būtų atvaizduojamą informaciją iš PostgreSQL, prie jau sukurto sluoksnio tektų kurti ir įrankius skirtus, jo redagavimui, ir kitoms manipuliacijoms, kas tolygu naujos programos sukūrimui. Vienintelis variantas būtų, kiekvieną syki pradedant ir baigiant darbą duomenys kopijuoti iš PostgreSQL duomenų bazės į ArcView. Tam tikslui kiekvieną kartą tektų sukurti standartinį ArcView sluoksnį, perkopijuoti į jį duomenis, o darbo pabaigoje, juos vėl kopijuoti iš ArcView į PostgreSQL lentelę. Kas tenkintų tik labai smulkius sluoksniu, dėl dviejų priežasčių: pirmiausia tai užima labai daug laiko, VBA veikia labai lėtai. Antra – jei jūsų kopijuojamas sluoksnis turi kokį nors sąryši duomenų bazėje, jūs privalote užtikrinti jog jis nebus sugadintas, o tai labai pavojinga ir gali sugadinti jūsų duomenų bazės vientisumą. Mano paminėtas variantas buvo realizuotas projekte PgArc (http://pgarc.sourceforge.net/), tačiau dėl minėtų priežasčių jis neprigijo.

Šios nesėkmės, ieškant sprendimų iškilusioms problemoms, sąlygojo atviro kodo programų, gebančių atlikti reikiamas funkcijas, paiešką. Vienintelis kriterijus buvo, JAVA programavimo klaba, dėl jau minėtų priežasčių. Kadangi rasta buvo ne viena programa, teko atsirinkti, ir būtent geriausias variantas buvo GeoTools2 biblioteka, bei jos pagalba sukurtos programos.

## GeoTools2

GeoTools2 yra mano nagrinėjamos sistemos ašis. Geotools2 tai yra atviro kodo JAVA programavimo kalba parašyta biblioteka. Ji turi modulinę architektūrą, taigi prie jos pakankamai nesunkiai galima prijungti dalis kurios gali atsirasti ateityje. GeoTools2 biblioteka buvo kuriama laikantis visų išleistų OGC specifikacijų ir ketina jų laikytis ateityje. Projekto tikslas yra sukurti rinkinį JAVA kalba parašytų modulių, įgalinančių vartotojus sukurti reikiamas programas darbui su OGC suderinamais servisais, ar programas teikiančias su OGC specifikacijomis suderinamas paslaugas. Geotools2 biblioteką sudaro realizuotos klasės, bei jų programinės sąsajos (interfeisai) skirtos kitoms funkcijoms [8]:

- Geometrinių figūrų abstrakcijai, saugojimui apdorojimui.
- Koordinačių sistemų transformacijoms.
- Koordinačių sistemų perprojektavimui.
- Sluoksnių stiliaus manipuliacijoms.
- Sluoksnių atvaizdavimui.
- Geometrinių figūrų tikrinimui.
- Grafų ir tinklų iš GIS informacijos sudarymui, bei analizei.

GeoTools2 darbui su geometrinėm figūrom naudoja dar vieną atviro kodo biblioteką JTS (Java Topology Suite). Šios bibliotekos pagalba GeoTools2 dirba su vektoriniais duomenimis. Ši biblioteka taip pat buvo kuriama laikantis OGC specifikacijos, ir GeoTools2 suteikia šias galimybes:

- OGC Geometrinių figūrų klasės.
- Standartizuotos operacijos su vektoriniais duomenimis

- Vektoriniai predikatai (DE-9IM modelis[17]).
- Persidengimo funkcijos(sankirta, skirtumas, sąjunga, simetrinis skirtumas).
- Buferio funkcija.
- Ploto bei atstumo skaičiavimo funkcijos.
- Topologijos teisingumo tikrinimo funkcijos.

Taip pat GeoTools2 deklaruoja GeoAPI projekto palaikymą ir jis dalinai yra įgyvendintas. GeoAPI - tai projektas, kurio tikslas sukurti vieningą JAVA bibliotekoms dirbančioms su GIS programinių sąsajų (Interfeisų) ir funkcijų rinkinį, įgyvendinanti visas OGC numatytas GIS specifikacija. Deja kol kas jis yra palyginus mažai naudojamas, greičiausiai todėl, kad daugelis bibliotekų, jau turi savo sąsajas, jų vartotojai yra prie jų pripratę ir todėl sunkiai GeoAPI skinasi kelią į kitus projektus.

Tiesa, būtina paminėti, jog kol kas Geotools2 moka dirbti tik su dvimačiais duomenimis, bet, kaip minėjau, daugumai darbų to pilnai pakanka. Į šį, mano nuomone, nedidelį trukumą galima užmerkti akis dėl labai didelių jos pliusų, tai OGC specifikacijų palaikymą bei jos modulinę struktūrą, leidžiančią pasiimti tas jos dalis, kurios reikalingos darbui. Dėl šių savybių GeoTools2 biblioteka naudojama įvairiuose projektuose, kaip pavienių asmenų taip ir atviro kodo organizacijų. Vieni žymesnių ir didesnių projektų būtų Geoserver ir Udig.

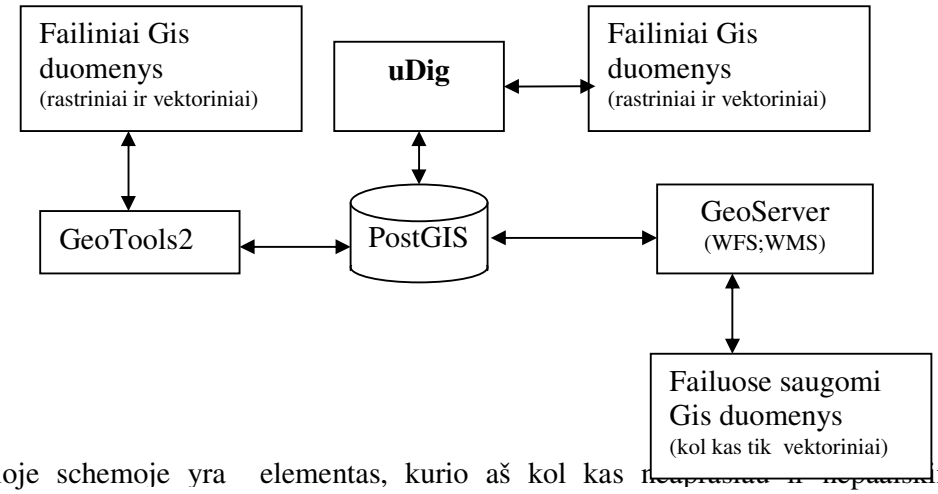
### uDig (User-friendly Desktop Internet GIS)

Viena iš pačių geriausių, mano nuomone, programų sukurtų GeoTools2 bibliotekos pagalba yra uDig. uDig unikali programa tuo, kad kol kas internete buvo galima rasti atviro kodo programų, kurios dirbdavo su tam tikrais duomenų tipais, pavyzdžiui, tik internetu teikiamomis paslaugomis kaip WMS ar WFS, ar skirtos darbui su failais, o uDig programa, kuri gali dirbti iškarto su keliais duomenų tipais.

Udig sukurta naudojant Eclipse platformą be GeoTools2, todėl ji yra nuo platformos nepriklausoma programa. Kadangi Eclipse turi galingą įskiepių sistemą, tai uDig praktiškai gali būti pritaikyta bet kokiems vartotojo poreikiams, o integruotų įskiepių kūrimo sistema PDE, bei atviras kodas ir pilna platformos dokumentacija, padaro šį uždavinį dar lengviau įveikiamą. Todėl uDig pavadinimas programa, praktiškai labai sumenkina jos galimybes, uDig – tai GIS platforma, kurios pagrindu galima kurti programas pritaikytas konkrečiai užduočiai. UDig programa buvo pradėta kurti pakankamai neseniai todėl ji dar netgi neįsisavinusi visų GeoTools2 bibliotekos palaikomų failų formatų tačiau pačius pagrindinius ji puikiai tvarko [7].

| Pavadinimas | Palaikomos funkcijos |
|---|---|
| Vektoriniai duomenys | |
| SHP | Skaito/Rašo |
| Rastriniai duomenys | |
| GeoTIFF | Skaito |
| JPEG | Skaito |
| PNG | Skaito |
| Duomenų bazės | |
| ArcSDE | Skaito/Rašo |
| Oracle | Skaito/Rašo |
| DB2 | Skaito/Rašo |
| PostGIS | Skaito/Rašo |
| Tinklo servisai | |
| WFS | Skaito/Rašo |
| WMS | Skaito |

Taigi jei panaudotumėme jau aptartas programas, būtų galima sukurti sekančią sistemą.




3.4.2 pav. Atviro kodo pagrindu surinkta sistema

Šioje schemoje yra elementas, kurio aš kol kas neapraš[illegible]inau – tai GeoServer. Tai d[illegible] lioteką. GeoServer - tai tinklo serveris skirtas GIS duomenis perduoti tinklu WFS ir WMF servisų pavidalų. WMS(Web Map service) – tai GIS duomenų perdavimas jpeg, png ar kitu taškinės informacijos saugojimo formatu. Labai naudingas ir patogus variantas kuriant viešai prieinamas svetaines su žemėlapiais, užtenka papraščiausia scenarijaus suformuoti užklausą WMS servisui ir jūs savo tinklapyje jau turite žemėlapį. WFS(Web Feature Service) – skirtas perduoti GIS duomenis tinklu WML pavidalu. Šioje schemoje jį paminėjau tik tam jog parodyčiau, kad atviro kodo programos sudaro vieningai veikiančią sistemą, aprėpti visas sritis.

Nesuku pastebėti jog 3.4.1 ir 3.4.2 schemos ne daug kuo skiriasi. Skirtumas na nebent elementų pavadinimuose, bet žinoma funkcijas jie atlieka praktiškai tas pačias, uDig attinka ArcView ir kitus produktus, paminėtus ankstesnėje schemoje. Pačių produktų savybių nenoriu lyginti, kadangi ištiesų jie smarkiai skiriasi, pagrinde vien tuo, kad ši atviro kodo sistema nemoka dirbti su trečiuoju matmenimi ir visi duomenys yra analizuojami tik 2D erdvėje, taip pat nepalaiko ši sistema ir M matmens, ką ankstesnėje schemoje paminėtos programos gali apdoroti. Bei nėra galimybių dirbti su topologija. Tačiau reikia nepamiršti, jog pavyzdžiui, GeoTools2 turi bibliotekas dirbui su topologija, tačiau uDig programoje jos nerealizuotos, lygiai tas pats su 3D ir M matmenimis, PostGis juos palaiko tačiau jų panaudojimas nerealizuotas nei GeoTolls2, nei uDig, nei GeoServer serveryje iš dalies vien dėl to, jog šie matmenys nėra specifikuoti. Reikalui esant galima naudoti kartu su šiomis ir kitas programas, pavyzdžiui GRASS, kuri išspręs visas išvardintas čia problemas. Svarbiausia nepamiršti, jog šios sistemos pranašumas yra atvirumas, jūs kiekvieną trūkstamą dalį galite pasiimti iš bet kokio kito atviro kodo projekto. Pavyzdžiui, uDig gali būti pakeistas QGIS, ar GRASS programa priklausomai nuo poreikio.

## Testavimas

Sprendžiant iš 3.4.2 nubraižytos schemos, tai tūrėtų būti labai patraukli sistema, tuo labiau, kad ji nemokama, turi visus reikiamus įrankius, yra nuo platformos nepriklausoma. Tačiau tam, kad būtume tuo tikri reikia patikrinti kaip ši sistema veikia. Šios sistemos testo smulkus aprašymas yra patalpintas Priede Nr.3 “Testavimas.doc”. O čia tiesiog patalpinau pačią rezultatų suvestinę tam, kad būtų galima palyginti nesigilinant į smulkmenas.

Testuojama programos uDig 1.0.6 versija, duomenų bazių serveris PostgreSQL 8.0.3. Kompiuteriai:

1) Pentium 4 1.9GHz, RAM 512MB Windows 2000, ir Ubuntu Linux 5.10.
2) Pentium 4 2.4GHz, RAM 512MB Windows XP.

Tinklas EtherNet 100Mbs, Tinklo serveris IBM 2x P3 1GHz, RAM 1GB.Rezultatai:

Kai naudojami SHP failai:

| Kompiuteris | Operacinė | Kliento kompiuterio apkrova % | Vidutinis laikas sekundėmis |
|---|---|---|---|
| 1 | Windows 200 | 83% | 9,5 |
| 1 | Linux Ubuntu 5.10 | 65% | 5,4 |
| 2 | Windows XP | 37% | 3,2 |

Kai naudojama PostgreSQL duomenų bazė:

| Kompiuteris | Operacinė | Kliento kompiuterio apkrova % | Serverio vidutinė apkrova % | Vidutinis laikas sekundėmis | Tinklo vidutinė apkrova B/s |
|---|---|---|---|---|---|
| 1 | Windows 2000 | 95% | 3,11% | 70 | 101195,17 |
| 1 | Linux Ubuntu 5.10 | 83% | 6,47% | 12,85 | 637178,7 |
| 2 | Windows XP | 66% | 7,02% | 10,83 | 796192,64 |

Kompiuteriai nėra ypatingai spartūs, tačiau manau, vis dar daug kur naudojami, be to lėtesniame kompiuteryje lengviau pastebėti programų ydas. Taigi 1 kompiuterio Windows aplinkoje programa veikia labai lėtai, naudojant duomenų bazę kaip duomenų šaltinį dirbti praktiškai neįmanoma. Nors Linux aplinkoje darbas spartesnis, tačiau nepakankamai.

Tame pačiame kompiuteryje veikiančių Linux ir Windows operacinių rezultatų skirtumas gali būti paaiškintas tik Java bibliotekos darbo su operacinės sistemos grafine aplinka problemomis. Akivaizdu, kad Windows 2000 Java dirba lėčiau nei Linux.

Akivaizdžiai padidėja kompiuterio apkrova ir darbo laikas dirbant su duomenų baze, tačiau visi parametrai nerodo jog problema būtu duomenų bazės serverio pusėje, nes serverio apkrova minimali. Greičiausiai tai ženklas jog GeoTools2 bibliotekos modulyje darbui su PostgreSQL duomenų baze ne viskas veikia tvarkingai.

Reikia pastebėti, jog ESRI ArcView programa šį darbą Windows 2000 aplinkoje, užkraudama SHP failus atlieka per 3-4 sekundes. Tačiau kadangi jos neįmanoma pajungti prie PostgreSQL ji nebuvo įtraukta ir pilnai testuojama.

Truputi galingesniame kompiuteryje, su Windows XP aplinka, akivaizdžiai geresni rezultatai, taigi galime padaryti išvada, jog su šiuolaikiniu kompiuterio ši sistema turėtų veikti pakankamai gretai, kompiuterio techninės įrangos sąskaita galime paslėpti programos trūkumus.

# 4 IŠVADOS

Akivaizdu kad atviro kodo GIS sistemos veikia pakankamai gerai, jog būtų galima jomis pasinaudoti, ne tik namuose, bet ir darbe. Be to vis naujų projektų atsiradimas rodo jog atviro kodo GIS programinė įranga yra naudinga. Besikeičiantis valstybių požiūris į atvira kodą suteikia naują prasmę patikimumui ir saugumui. Didžiųjų komercinių kompanijų strategijos ir planai atviro kodo atžvilgiu jau keičiasi, todėl labai svarbu turėti aiškius orientyrus naujuose galimybių plotuose. Taigi apibendrinant darbo rezultatus galima padaryti sekančias išvadas:

1. Atlikta GIS atvirų standartų analizė, jos metu nustatyta, jog pagrindiniai duomenų saugojimo formatai atviro kodo GIS srityje yra:
    a. Vektorinių duomenų – SHP.
    b. Rastrinių duomenų –GeoTIFF.
    c. Duomenų bazės – OGC SFS(PostGIS).
    d. Tinklo servisams – WMS ir WFS.
2. Funkcionalumo atžvilgiu ištirta Geotools biblioteka ir uDig programa. Nustatyta Geotools silpnoji vieta darbo našumo požiūriu – SWING grafinė aplinka.
3. Įgyti nauji įgūdžiai dirbant su Eclipse PDE, Geotools bibliotekos sąsajomis, bei ArcMap įrankių kūrime.
4. Pateiktas ArcMap ir PostgreSQL sąsajos realizavimo variantas. Nustatyti realizacijos trūkumai, bei jos panaudojimo galimybės.
5. Sukurtas įskiepis demonstruojantis uDig programos funkcionalumą.
6. Testo metu nustatyta, jog darbe nagrinėtai atviro kodo GIS programinei įrangai reikalinga galingesnė kompiuterinė technika, nei analogiškai komercinei ArcView programai, tam pačiam darbui atlikti.
7. Atlikus analizę nustatyta, jog esant nedideliam vartotojų kiekiui(dirba 3-5 vienu metu) ekonomiškai naudingiau naudoti atviro kodo programinę įrangą.

# Turinys

## Literatūros sąrašas:


1) GIS metraščio projekto tinklapis: http://www.casa.ucl.ac.uk/gistimeline/gistimeline.html
2) Elektroninė atvira enciklopedija : http://en.wikipedia.org/wiki/Landsat_1
3) Nicholas R. Chrisman "History of the Harvard Laboratory for Computer Graphics: Poster Exhibit"
4) Profesoriaus David A. Langrebe asmeninis tinklapis : http://dynamo.ecn.purdue.edu/~landgreb/Landsat.paper.pdf
5) Atviro kodo GIS paketo GRASS tinklapis : http://grass.itc.it/
6) Atviro kodo bibliotekos OGR/GDAL tinklapis :http://www.gdal.org/
7) Atviro kodo programos uDig tinklapis: http://udig.refractions.net/confluence/display/UDIG/Home
8) Atviro kodo bibliotekos GeoTools tinklapis: http://www.geotools.org/display/GEOTOOLS/Overview
9) Atviro kodo platformos Eclipse tinklapis :www.eclipse.org
10) IT saugumo kompanijos tinklapis: http://www.sourcefire.com/products/downloads/secured/sf_swoss.pdf
11) OSI tinklapis : http://www.opensource.org/docs/definition.html
12) Europos sąjungos veiklos planas: http://europa.eu.int/information_society/eeurope/2002/action_plan/pdf/actionplan_en.pdf
13) JAV nacionalinės geologijos tarnybos tinklapis: http://www.usgs.gov/
14) Atviro standarto projekto GeoTIFF tinklapis: http://www.remotesensing.org/geotiff/geotiff.html
15) ESRI kompanijos, SHP formato specifikacija: www.esri.com/library/whitepapers/pdfs/shapefile.pdf
16) atviros tarptautinės GIS standartizacijos organizacijos tinklapis: www.opengeospatial.org
17) atviros GIS SFS specifikacijos dokumentas: http://www.opengeospatial.org/docs/99-049.pdf
18) Matthew Scarpino, Stephen Holder „SWT/JFace IN ACTION" , MANNING, 2005
19) Refractions kompanijos atviro kodo GIS ataskaita http://www.refractions.net/white_papers/ oss_briefing/2005-02-OSS-Briefing.pdf

# Anotacija

Šiaulių universitetas
Informatikos fakultetas
Informatikos katedra

Magistro studijų baigiamasis darbas
Pavadinimas:

ATVIRO KODO GIS TECHNOLOGIJŲ TYRIMAS

Autorius: Andrius Božnis

Gis yra viena iš sparčiausiai besivystančių informatikos mokslo sričių. Savyje ji apjungia grafinį vaizdą ir duomenų bazių sistemas, tokiu būdu GIS tampa plačiai pritaikoma bei labai reiklia ir specifine sistema. Komercinių GIS programų rasti nesunku, jų funkcionalumas dažniausiai yra gerai išreklamuotas ir žinomas, tuo tarpu atviro kodo programos lieka pamirštos.

Darbe atliekama analizė internete platinamos atviro kodo programinės įrangos, tiriant atskirų projektų tarpusavio santykius, programavimo kalbas. Taip pat analizuojami atvirūs GIS standartai. Analizuojant bandoma surasti svarbiausius, funkcionalumo požiūriu, programinės įrangos bruožus. Detaliau tiriama, analizės metu nustatytus kriterijus atitinkanti, programa, realizuojamas įskiepio pavyzdys atviro kodo GIS platformai.

# Summary

Šiauliai University
Faculty of informatics
Department of informatics

Master's final work
Title:

ANALYSIS OF OPEN SOURCE GIS SOFTWARE

Author: Andrius Božnis

GIS is one of the most perspective information technology sciences sphere. GIS conjuncts the digital image analysis and data base systems. This makes GIS wide applicable and very high skills demanding system. There is a lot of commercial GIS software which is well advertised and which functionality is pretty well known, while open source software is forgotten.

In this diploma work is made analysis of available open source GIS software on the Internet, in the scope of different projects interrelations, programming languages. Also analysed open GIS standards. More detail analysis made on software selected by criteria, found by previous analysis. Demonstrating plug-in creation process on open source GIS platform.

CD turinys:

| Katalogas | Failas | Aprašymas |
|---|---|---|
| BIN | AttributeViewer.zip<br>pgArc.zip | Paruoštas įskiepis diegimui |
| DATA | | |
| Plugin_test | *.shp | Įskiepio testavimui paruoštas duomenų rinkinys |
| SHP | *.shp | UDig testavimui skirtas duomenų rinkinys |
| SQL | *.sql | UDig testavimui skirtas duomenų rinkinys, reikalingas PostgeSQL lentelėm sukurti |
| Testo_duomenys | *.xls | Atlikto testo rezultatai excel formate |
| DOC | AndriusB_Atviro_kodo_GIS_tyrimas.doc | Darbo aprašymas |
| | Diegimo Instrukcija.doc | Eclipse darbinei aplinkai įdiegti skirta instrukcija |
| | Priedas Nr.1 ArcView-PostgreSQL sąveika | |
| | Priedas Nr.2 Įskiepio kūrimas.doc | |
| | Priedas Nr.3 Testavimas.doc | |
| | Vartotojo instrukcija.doc | |
| | Javadoc.zip | Įskiepio klasių dokumentacija |
| INSTALL | | |
| windows<br>linux | | Paketai reikalingi Eclipse ir uDig darbo aplinkoms įdiegti |
| SRC | AttributeViewerProject.zip | Įskiepio Eclipse projektas, su išeities kodais |

# Priedai

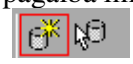
# *ArcView – PosgreSQL sąvaika*

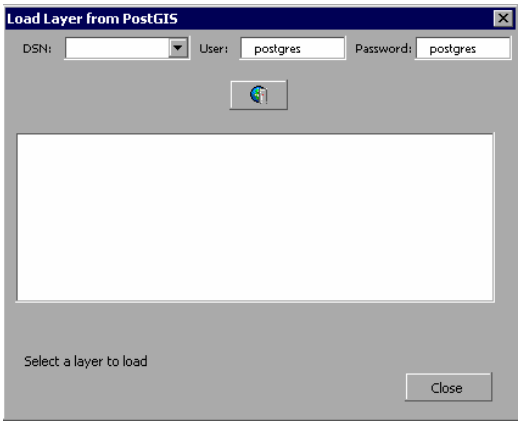
# Įvadas

ArcView galimybės yra labai didelės, ji turi labai daug įvairiausių įrankių, ir nors savo ESRI kompanijos produktų linijoje yra pats kukliausias paketas tačiau jame yra absoliučiai viskas, kas reikalinga našiam darbui su GIS informacija. Galimybė dalintis duomenis vienu metu su kitais vartotojais yra vienintelis, mano nuomone, trūkumas šiame pakete. Žodis trūkumas čia nėra labai tikslus apibūdinimas, kadangi iš tiesų ArcView gali dirbti kartu su kitais vartotojais, tačiau tam reikalingas papildomas komponentas – ArcSDE, bei duomenų bazė. ESRI produktų sąveiką čia nebus nagrinėjama, apie ją daug informacijos yra gamintojo tinklapyje www.esri.com , taip pat apie tai užsiminta magistrinio darbo aprašyme, tyrimo dalyje. Aiškumo dėlei pasakysiu jog šis komponentas, ArcSDE, atlieka tam tikrą vertėjo funkciją tarp programos ir duomenų bazės. ArcSDE leidžia išsaugoti duomenų bazėje GIS informaciją, net tuo atveju jei naudojama duomenų bazė neturi tam galimybių. Tačiau šis komponentas(ArcSDE) mano nuomone yra brangus ir nenaudingas, jei reikia leisti dirbti tik 2-3 vartotojams, kadangi dar daugelis funkcijų lieka nepanaudotos, o už jas visviena jau esi sumokėjęs. Šiame kontekste buvo atliktas tyrimas ieškant būdų sujungti ArcView paketo ArcMap programą ir atviro kodo duomenų bazę PostgreSQL, siekiant taip įgyvendinti kelių vartotojų darbą, vienu metu, su GIS duomenimis.

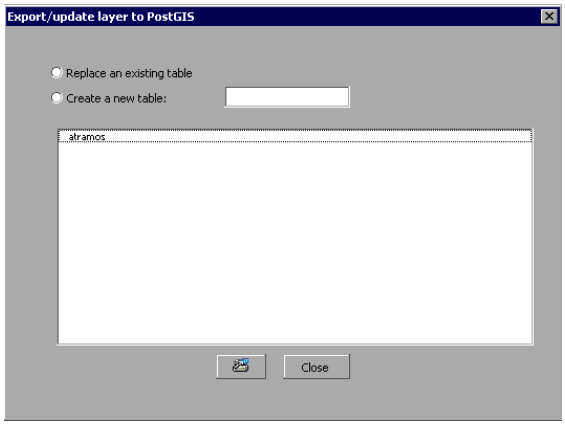
# PgArc

Pirmasis bandymas tai pasiekti buvo panaudojant jau esanti projektą PgArc, kuris randasi adresu http://sourceforge.net/projects/pgarc. Šis įskiepis yra skirtas ArcView paketo ArcMap programai. Jis, naudodamasis ArcMap, komponentais sukuria įrankį galinti importuoti duomenis į ArcMap iš PostgreSQL duomenų bazės arba juos eksportuoti į minėtą duomenų bazę. PgArc nėra populiarus projektas, jo neatnaujina jau daugiau nei tris metus ir jame esančios klaidos neištaisytos, tačiau kadangi jis yra atviro kodoatlikti reikiamas modifikacijas galima pačiam.. Visas kodas yra parašytas Visual Basic kalba ir taip pat yra jo variantas skirtas vidiniam ArcMap naudojimui naudojant integruotą VBA(VisualBasic for Application) interpretatorių. Būtent šiuo variantu ir naudosiuos patogumo dėlei. Prisijungimui prie duomenų bazės jis naudoja ODBC, todėl reikia turėti Postgre <-> ODBC tvarkyklę norint juo pasinaudoti.

Šio įrankio pagalba labai patogu perkelti PostgreSQL esančius duomenis į ArcMap, tačiau jis negali išsaugoti pakeitimu duomenų bazėje. Taip yra todėl, kad jis duomenis perkelia visus iš karto. Tam kad būtų aiškiau tarkime, jog turime PostgrSQL duomenų bazėje GIS duomenų sluoksnį. Mums reikia šiame sluoksnyje pataisyti vienos figūros atributinę informaciją. Tuomet mes turime PgArc įrankio pagalba importuoti sluoksnį į ArcMap. Šiam tikslui mes paspaudžiame pažymėtąjį mygtuką  ir iškritusiame lange - duomenų importo lange (žr 1. pav) pasirenkame ODBC duomenų sluoksnį. Tuomet ArcMap programoje sukuriamas pasirinkto sluoksnio atvaizdas.




1.Pav. Duomenų importas

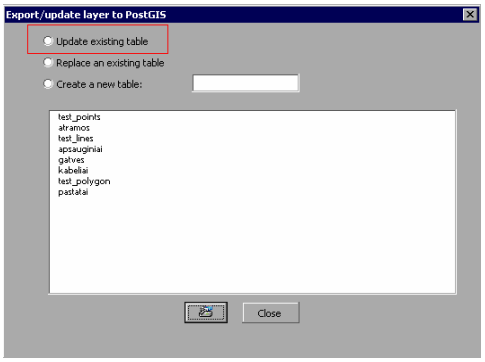



2.Pav. Duomenų eksportas

Šis sluoksnio atvaizdas yra ne kas kitas kaip SHP formato failas, sukurtas PgArc, ir į trauktas į ArcMap sluoksnius. Failas randasi operacinės sistemos standartiniame Temp kataloge. Todėl išjungus programą, sekantį kartą jo pasižiūrėti gali ir nepavykti. Taip pat šalio šio failo sukuriamas ir PgArc specialus tekstinis failas *.dbp kuriame saugoma informaciją apie sukurto sluoksnio ryši su PostgreSQL lentele. O būtent atviru tekstu ODBC vardas, vartotojas ir slaptažodis, figūros geometrijos tipas, figūros koordinačių sistemos identifikacinis numeris ir duomenų importo data. Ši informacija yra būtina tam jog vėliau sluoksnį galėtume importuoti atgal į duomenų bazę be jokios papildomos informacijos įvedimo. Tačiau reikia nepamiršti jog šiame faile atviru tekstu saugomas slaptažodis. Tai nėra gerai ir reiškia jog naudoti tą patį kompiuterį keliems vartotojams turintiems skirtingas teises duomenų bazėje yra nesaugu. Pati geometrija kaip jau minėjau saugoma SHP faile, o tai iš tiesų yra gerai, kadangi tai leidžia duomenų redagavimui naudoti gausius ArcMap įrankius, taip pat be iškraipymų konvertuoti ir perkelti į kitas programas.

Tarkime jog mes pataisėme reikiamą informaciją. Kaip pakeitimus užregistruoti PostgreSQL duomenų bazėje. Likęs, nepanaudotas, mygtukas įrankių juostoje atveria duomenų eksporto, atgal į PostgreSQL duomenų bazę, langą, pavaizduota antrame paveikslėlyje. Jau iš esamų duomenų eksporto pasirinkimo galima suprasti kaip tie duomenys atsidurs duomenų bazėje. Taigi šis įskiepis turi tik du metodus, tai sukurti naują lentelę(Create a new table) su nurodytu pavadinimu ir perrašyti esamą(Replace an existing table). Kas atsitinka su duomenimis pasirinkus pirmąjį variantą turbūt ir taip aišku, tiesiog duomenų bazėje atsiranda lentelė nurodytu pavadinimu. Perrašymo metodas, pirmiausia sukuria laikinėją lentelę, į kurią perkopijuojama perrašomosios lentelės senoji informacija, tuomet iš perrašomosios lentelės informacija ištrinama, ir į ją bandoma įkelti ArcMap sluoksnio kopiją. Jei operaciją baigiasi sėkmingai, o tai programa nusprendžia pagal tai, ar kopijuojamas sluoksnis ir lentelė turi po vienodą kiekį įrašų, tuomet laikinoji lentelė panaikinama iš duomenų bazės, jei ne, atstatoma perrašomosios lentelės informacija iš laikinosios.

Jei turėti omenyje, jog mūsų sluoksnis duomenų bazėje yra pavienė duomenų lentelė, tai šie metodai yra abu tinkami duomenims išsaugoti, tačiau reliacinėje duomenų bazėje toks variantas labai retas. Todėl jei panaudosite lentelės perrašymo būdą, dalis informacijos „surištose“ reliaciniais saitais lentelėse bus ištrinta arba, dar blogiau, bus pažeistas duomenų vientisumas, ir




3. Pav. Patobulintas duomnų eksportas

įvesta begalė klaidų nes unikalus figūros numeris pasikeis.

Norėdamas išspręsti šią problemą nutariau patobulinti šio įskiepio kodą, įdėdamas dar vieną duomenų eksporto metodą duomenų atnaujinimo metodą(žiur. Pav. 3 Update existing table), bei suteikdamas įskiepiui galimybę duomenis eksportuoti į PostgreSQL duomenų bazę ne tik iš SHP failų, bet ir iš ArcMap Access duomenų bazės vadinamo GeoDatabase formato.

Duomenų atnaujinimo metodo esmė - unikalių sluoksnio figūrų identifikacinių numerių išsaugojimas. Jis netrina ir neperrašo duomenų be reikalo, taip neleisdamas duomenų bazei suteikti duomenų įrašams naujus identifikacinius numerius. Gavęs komandą pradėti duomenų atnaujinimą įskiepis atlieka tris žingsnius. Pirma – patikrina ar ArcMap programos eksportuojamame sluoksnyje neatsirado naujų figūrų, jei tokių atsirado tuomet jas prideda. Antra- patikrina ar nepasikeitė jų atributinė ir geometrinė informacija. Šis žingsnis labai smarkiai eikvoja kompiuterio resursus, kadangi jam būtina kiekvieną ArcMap sluoksnio figūros koordinates transformuoti į tekstinį WKT formatą ir po to palyginti su duomenų sluoksnių duomenų bazėje, lygiai tokia pati procedūra atliekama ir su atributine informacija. Ir paskutinis trečiasis žingsnis – tikrinama ar sluoksniuose nebuvo pašalintos tam tikros figūros. Tokia veiksmų tvarka leidžia užtikrinti jog nei vienas duomenų pakitimas neprasmuks nepastebėtas ir procedūros pabaigoje tūrėsime visiškai vienodus du sluoksnius, vieną ArcMap programoje, kitą PostgreSQL duomenų bazėje. Tačiau šis metodas turi ir trūkumų kurie kyla iš ArcMap darbo principo su unikaliais figūrų identifikaciniais numeriais – ArcMap neleidžia su jais atlikti jokių manipuliacijų. Jie suteikiami vos tik sukuriama figūra ir vienintelė operacija kuria galima atlikti, tai jų nuskaitymas, todėl nėra jokios galimybės atlikti atvirkščia procedūrą minėtajai, tai yra, pagal PostgreSQL duomenų bazės pakitimus atnaujinti ArcMap sluoksnį.

## Diegimas

Pirmiausia jūsų kompiuteryje turi būti įdiegta ESRI ArcView 8.x programinė įranga, tuomet norint patikrinti šios programos veikimą jums reikia išskleisti PgArc.zip failą esantį kompaktinio disko bin kataloge. Išskleidę failą, jo sukurtame katalogo pakatalogyje src rasite pgarc.mxd failą, kurį reikia spragtelti pelės pagalba. Pasileidus ArcMap programai, jos įrankių juostoje, bus atsiradusi nauja įrankių juosta su dviem 1. pav. pavaizduotais mygtukais.

## Apibendrinimas

Šis įskiepis iš tiesų yra labai paprastas savo veikimo principu. Kadangi jis duomenis perkopijuoja i SHP sluoksnį, tai leidžia maksimaliai išnaudoti ArcView paketo įrankių galimybes, tačiau ryšys tarp programos ir duomenų bazės įmanomas tik į vieną pusę, tai yra, pagal ArcMap duomenis atnaujinti duomenų bazėje esančius duomenis. Atnaujinimo metodo didžiausias trūkumas, darbo sparta. Visual Basic interpretatorius nėra pats sparčiausias kas savaime sulėtina programos veikimą, be to atnaujinimo metu visuose trijuose žingsniuose turi būti atlikta visų lentelės įrašų peržiūra, darbo laikas dar padidėja. Kompakte esančių duomenų sluoksnio „test_points“ atnaujinimas naudojant P4 1,9GHz kompiuterį trunka apie 30 min. Tai labai ilgas laikas, tačiau reikia nepamiršti jog šį veiksmą greičiausiai teks atlikti tik kartą per darbo dieną. Taip pat būtina pažymėti, jog ryšiui su duomenų baze nenaudojamos transakcijos, kas kelia rizika sugadinti duomenis jų atnaujinimo metu. Turint omeny jog atnaujinimas gali trukti iki 30 min. ar net ilgiau tai labai svarbi problema. Taip pat įskiepis visiškai neužtikrina saugaus darbo keliems klientams su duomenų baze vienu metu. Dar viena problema, kurią paminėjau vos pradėdamas šio įskiepio aprašymą – įskiepio bendruomenės nebuvimas, šiame projekte per paskutinius tris metus neįvyko jokių pakitimų, nors akivaizdžiai matosi jog jame pilna klaidų, o tai reiškia, jog juo niekas nesidomi. Todėl iškilusias problemas naudojant įskiepį tektų spręsti savo jėgomis.

Iš šio apibendrinimo galima padaryti išvadą jog šis įskiepis tiktų tik darbo modeliui, kai duomenų bazę redaguoja tik vienas žmogus, o kiti naudojasi duomenimis, be tesės juos

modifikuoti. Toks būtu vienintelis saugus jo pnaudojimo būdas. Todėl nors įskiepis veikia, tačiau jis absoliučiai nenaudingas, kadangi tokį pat darbo modelį turime ir naudodami standartinę Access duomenų bazę, taip vadinamu GeoDatabase formatu, o duomenis eksportuodami į .SHP formatą ir juos perduodami vartotojams.

# Įskiepio AttributeViewer kūrimas

# Įvadas

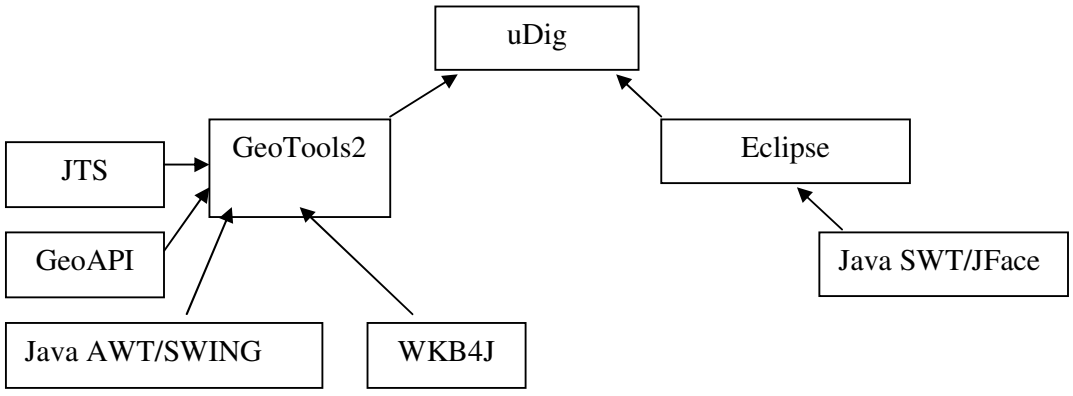
Jei jau spėjote susipažinti su uDig programa turbūt pastebėjote, jog ji kol kas nėra labai turtinga savo įrankiais, duomenų redagavimo srityje. Su esamais įrankiais tikrai sunku užsiimti normaliu darbu, kadangi visi įrankiai skirti dirbti tik su pavieniais objektais žemėlapyje ir esi priverstas baksnoti kiekvieną objektą, ne tik tą kurio informaciją nori pasižiūrėti, bet ir tuos kurių informacija nori redaguoti. Toks monotoniškas darbas greitai pabosta. Todėl buvau priverstas sukurti darbą palengvinantį įskiepį. Tokią įskiepio idėją pasiskolinau iš panašų funkcionalumą turinčių programų kaip ESRI ArcView, MapInfo. Taip pat tai puiki proga susipažinti su Eclipse įskiepiu sistema, uDig struktūra, bei biblioteka GeoTools2, kuri yra varomasis variklis šios programos.

Kalbėdamas darysiu prielaidą, jog jūs žinote kaip sukurti patį paprasčiausią uDig įskiepį. Taip darysiu tam, jog nereiktų pasakoti elementarių dalykų, kuriuos galime lengvai rasti ir Google pagalba. Siūlyčiau paskaityti informaciją esančią nuorodoje http://udig.refractions.net/confluence/display/DEV/3+Plugin+Tutorial čia trumpai ir labai aiškiai parašyta, kaip tai padaryti.

Taip pat prieš pradedant kalbėti turiu priminti, jog viskas yra taikoma tik stabiliai uDig versijai 1.0.6 ir jos stabiliai SDK versijai 1.0.5. Kadangi programa yra pakankamai nauja, o jos tobulinimas vystosi labai sparčiai, todėl yra didelė tikimybė, jog mano sukurtas įskiepis naujesnėse versijose neveiks visiškai, ar bent dalinai.

# uDig struktūra

Turbūt jau pastebėjote, jog uDig yra Java kalba parašyta programa. Kuri susideda iš didelio skaičiaus įskiepių, taip yra todėl, kad uDig naudojasi Eclipse platforma grafinei aplinkai įgyti. Be grafinės aplinkos uDig įgyja ir Eclipse būdingą lankstumą, bei perima jos struktūros ypatumus. Pav. 1 parodytos bibliotekos įeinančios į uDig programos sudėtį




Pav.1 uDig struktūra

Nors Eclipse pusė atrodo labai kukliai, taip ištikrūjų nėra, tačiau mums kur kas svarbesnė GIS bibliotekų hierarchija, nei Eclipse.

Taigi sudėjus abi bibliotekas į vieną krūvą, galime teigti, jog gavome GeoTools2 bibliotekos variantą įskiepių plotmei. Tačiau iš tiesų GeoTools2 biblioteka nesikeičia, tiesiog atsiranda papildomas sluoksnis, kuris leidžia naudojantis įskiepiais, GeoTools2 bibliotekos funkcijas perkelti į Eclipse grafinę aplinką, o tai ir yra galutinis produktas – uDig. Taigi čia jau

pakankamai aišku, jog norint sukurti įskiepį, reikia naudotis Eclipsės įskiepių sistema, ir dirbti su GeoTools klasėmis.

Kuriant įskiepį Elipsėje labai svarbūs yra išplėtimo taškai (Extension points), kadangi tai yra įėjimas į unikalią Eclipse registro sistemą, suteikiančia galimybę prieiti prie išplėtimo taškuose numatytų klasių ir objektų, programos veikimo metu. uDig turi savo išplėtimo taškų kolekciją, kuri suteikią galimybę valdyti ir prisijungti prie tokių esminių GIS programai elementų kaip žemėlapiai, sluoksniai, duomenų saugyklos ir t.t. Visi dabar egzistuojantys uDig išplėtimo taškai surašyti šiame tinklapyje: Išplėtimo taškai. Išplėtimo taškų kiekis priklauso nuo programuotojų poreikių, todėl jis po truputi didėja. Nepasitvirtinę variantai pašalinami, kai kurie dalinai pasikeičia, todėl būtina pasitikrinti ar reikiamas išplėtimo taškas tebenaudojamas, ypač turint omenyje uDig vystimosi tempus.

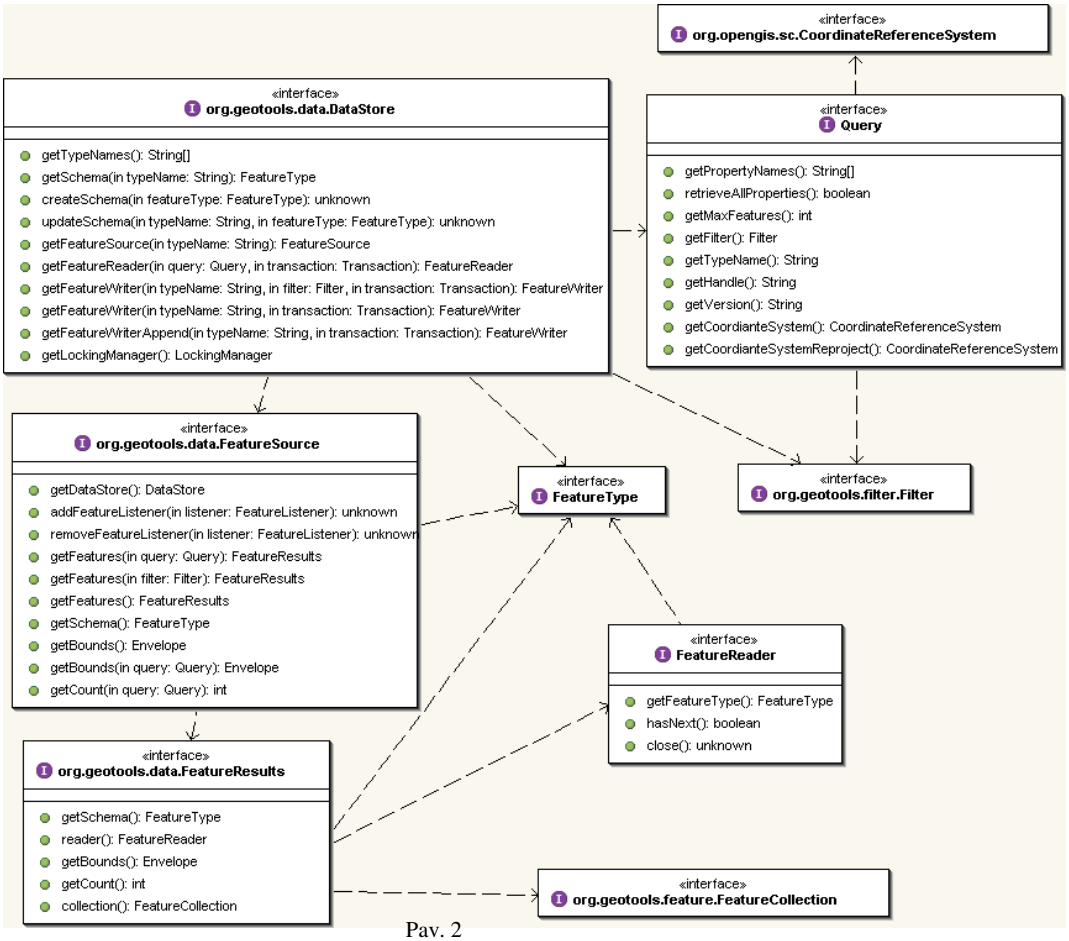
Žinoma, kuriant galima naudotis ir Eclipse teikiamais išplėtimo taškais. Būtent toks variantas ir naudojamas mano kuriamame įskiepyje.

## GeoTools2 bibliotekos struktūra

GeoTools2 – modulinė biblioteka, ji labai didelė, turi daug moduliu suteikiančių jai labai didelių galimybių. Pavyzdžiui, norint įdiegti bibliotekai naujo, dar bibliotekoje nenaudojamo, duomenų šaltinio palaikymą teriktų sukurti papildomą modulį, kuris tūrėtų realizuoti, bibliotekos numatytą duomenų struktūroms, sąsaja. Todėl naujo funkcionalumo įdiegimas niekaip neįtakotų, jau sukurtų, bibliotekos pagrindu programų. Tai taip pat leidžia bibliotekai lengvai atsikratyti nereikalingo „antsvorio“ – užimamos disko vietos.

Taigi kaip minėjau bibliotekos modulių yra labai daug, todėl jų visų neaptarinėsiu, o tik paminėsiu mano įskiepyje naudojamų struktūrų ypatybes. Įskiepis dirba su atributiniais duomenimis, todėl jis visų pirma naudoja GeoTools2 duomenų apdorojimo klases, taip pat duomenų filtracijai naudojamos filtrų klasės.

Visos su duomenimis dirbančios klasės realizuoja Pav. 2 atvaizduotą sąsajų hierarchiją (paveikslėlis paimtas iš (http://www.geotools.org/display/GEOTOOLS/Data+Reading ).




Pav. 2

Taigi matome, jog visi duomenys bibliotekoje yra saugomi specialiuose objektuose *DataStore*. Tai ypatingi objektai, jie savyje gali laikyti kelių tipų duomenis. Juose duomenys randami pagal jų pavadinimą. Pavyzdžiui, jei mes dirbtume su SHP failu, tai nuorodą į jo duomenų šaltinį gautume nurodę jo failo vardą. Smulkesni objektai, kurie saugo tik vienos rūšies informaciją yra *FeatureSource*. Jie yra paimami iš *DataStore* duomenų. Šie objektai jau tiksliai žino, kokias figūras saugo, jų tipus. Šie objektai gali pagal nurodytus filtrus atrinkti reikiamus geometrinius objektus ir pateikti juos specialiame objekte *FeatureResult*. Kuris savo ruožtu gali suteikti dar vieną specialų objektą skirtą duomenų skaitymui iš *FeatureResult* objekto.

Geometrinio objekto tipo informacija saugoma *FeatureType* objekte, būtent iš jo kuriamas įskiepis gauna informaciją apie sluoksnyje esančių laukų kiekį, jų tipus, pavadinimus. Ši informacija dažnai vadinama schema. Žinant šią informaciją, jau iš *FeatureReader* objekto, *next()* metodo pagalba gauto objekto *Feature,* kuris schemoje nepažymėtas, galima sužinoti jo atributinės informacijos reikšmes.

Būtent įskiepyje ir naudojama tokia darbo schema, tik jos viršūnė yra ne *DataStore* objektas, o *FeatureSource*, kadangi iš pasirinkto sluoksnio mes gauname iš karto šią nuorodą. Tokia schema ir suteikia bibliotekai lankstumo, nes kiekvienas modulis realizuoja šią schemą. Ir kadangi programoje operuojama tik sąsajų ( Java interfeisų) rodyklėmis, dėl polimorfiškumo savybių, kurias suteikia objektinis programavimas, mano programai visiškai nebūtina žinoti šių sąsajų realizacijos, ir ji vienodai gerai skaito tiek iš failo, tiek iš duomenų bazės.

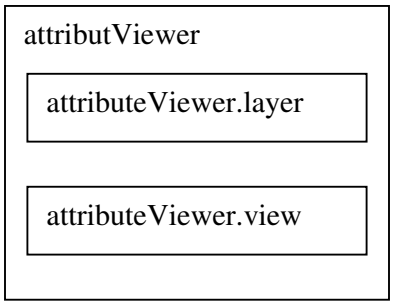
Filtrai GeoTools bibliotekoje taip pat sudaro sąsajų (Java interfeisų) hierarchiją, kurios privalumais naudojasi praktiškai visi su duomenim manipuliuojantys objektai. Filtrų hierarchijos nebraižysiu, kadangi aš naudoju savo darbe vos tik kelias klases, tačiau ją galima rasti šiuo adresu: http://www.geotools.org/display/GEOTOOLS/Filters. Savo darbe naudoju tik vieną iš

*Filtre* sąsajos išvestą naują sąsają – *FidFilter*. Kuri reiškia filtraciją pagal unikalųjį geometrinio objekto numerį, bet apie tai kalbėsiu vėliau.

Kadangi niekas negali išsiversti be duomenų, tai šią duomenų struktūrą taip pat labai aktyviai naudoja ir kiti moduliai. Dar vienas svarbus aspektas mano rašomam praktiniam darbui yra žemėlapio sluoksnis, kurio pažymėtais objektais manipuliuoja įskiepis. Sluoksnio objektas kaip šaltini naudoja būtent *FeatureSource* objektą, kuris turi labai patogią filtracijos funkciją, ir pagal nustatytą Filtro klasę gali pažymėti jame saugomus objektus ir atvaizduoti ant žemėlapio. Todėl tam tikrų komandų metu aš naudojuos šiais ypatumais, nesunkiai savo įskiepiui suteigdamas labai naudingas ir, iš pirmo žvilgsnio, sudėtingas funkcijas.

## Įskiepio struktūra ir veikimo principai

Savyje įskiepis realizuoja dviejų bibliotekų sąjungą. Iš GeoTools2 bibliotekos naudodamasis išplėtimo taškais įskiepis gauna informaciją, o Eclipse platformos suteiktos grafinės aplinkos pagalba, ją atvaizduoja, bei suteikia patogų ir išvaizdų valdymą. Įskiepis sudarytas iš trijų paketų žiūr. Pav. 1. *attributViewer* pačiame hierarchijos viršuje yra viena klasė *AttributeViewerPlugin*, kuri būtina norint prisiregistruoti Eclipse įskiepių sitemoje. Taip ji vaidina labai svarbų vaidmenį pačio įskiepio egzistavimui, kadangi būtent jos vidinių statinių kintamųjų pagalba yra realizuotas duomenų perdavimas iš žemėlapio sluoksnio į lentelę. Sekantis paketas *attributeViewer.layer*, yra skirta „prisirišti“ prie žemėlapio sluoksnių, tai atlieka specialus išplėtimo taškas: *layerOP*. *LayerTool* klasė realizuoją šio išplėtimo taško teikimą sąsają *IOp*. Šis išplėtimo taškas suteikia galimybę atlikti operacijas su sluoksniais. Būtent dėl šios priežasties paspaudę dešinį plės mygtuką matome meniu Anglysis->AtributeViewer. *IOp* turi vieną vienintelį metodą, kuris iškviečiamas pasirinkus minėtą meniu punktą, bei jam perduoda pasirinkto sluoksnio objekto rodyklę. Ši rodyklė yra perduodama AttributeViewerPlugin klasei ir ten įrašomą į statinį kintamąjį, prie kurio turi priėjimą lentelę realizuojanti klasė. Baigdamas darbą metodas iškviečia lentelę realizuojančia klasę.



Pav. 1

Paketas *attributeViewer.view* realizuoja lentelę, bei visas jos atliekamas operacijas. Tam tikslui jo pagrindinė klasė, *LayerViewer,* realizuoją Eclipse išplėtimo tašką *org.eclipse.ui.views* ir jo sąsają *ViewPart*. Tokiu būdu ši klasė įregistruojama į Eclipse peržiūros langų hierarchiją. Pavyzdžiui, kol dar nepasirinkote sluoksnio, pagrindiniame uDig meniu pasirinkite Windows->Show View-> Other. Atsiradusiame lange vėl spragtelkite ant grupės pavadinimu Other ir ten rasite mūsų lentelę *AttributeViewer*. Jei ją iškviesite, ji vietoj savo turinio, parodys užrašą „Nepasirinktas sluoksnis“. Visą atributinę sluoksnių informaciją ši klasė gauna iš *AttributeViewerPlugin* klasės *src* ir *layer* kintamųjų, kurie dar nėra inicializuoti, todėl vietoj lauktos lentelės - perspėjimas.

Šiame pakete taip pat yra visos klasės atsakingos už visų šios lentelės atliekamų operacijų vykdymą. Jei būti tiksliam, jos tik reaguoja į įvykius, o operacijos vis tiek atliekamos *LayerViewer* klasės metodų pagalba. Taip buvo specialiai sugalvota, kadangi įvairioms operacijoms kartais reikia kelių duomenų šaltinių, todėl tektų kuriant klases reaguojančias į įvykius, perdavinėti joms duomenis jų konstruktorių pagalba. Tačiau problema ta, jog tos klasės bus kuriamos tik vieną kartą, vadinasi perduodami duomenys taip pat bus tik vieną kartą, o jei jie pasikeis jų nebebus kaip perduoti, teks galvoti naujus mechanizmus. Palikus visas operacijas *LayerViewer* klasės viduje problema išsispendžia pati, kadangi kai duomenys keičiasi šiai klasei apie tai yra pranešama iš karto todėl ji visada dirba su naujausiais duomenimis.

## *Lentelės funkcijos*

Iš viso numatytos 3 funkcijos :

1. Lentelėje esamų eilučių atrinkimas pagal žemėlapyje pažymėtus objektus.
2. Žemėlapio objektų pažymėjimas remiantis, lentelės pasirinktomis eilutėmis.
3. Duomenų redagavimas.

Pirmąjį funkcionalumą lentelėje galima pasiekti pasirinkus meniu punktą „Selected“. Šis funkcionalumas įgyvendinamas naudojantis *Filtre* sąsaja realizuojančiais objektais. Kadangi duomenis objekte *FeatureSource* galima trinkti *Filter* realizacijos pagalba, o taip pat ir *Layer* objekte, kuris duomenis gauna taip pat iš *FeatureSource* objekto, man liko tik įdiegti lentelės eilučių atrinkimą pagal *Filtre* ir į mano naudojamą lentelės duomenų surinkimo klasę *TableContentProvider*. Joje sukūriau specialų kintamąjį, bei jo *setFilter* ir *getFilter* metodus, o *getElement* metode įdėjau eilutę, kuri iš *FeatureSource* ištraukia tik filtro kriterijus atitinkančias eilutes. Tokių būdu užtena tik nusiųsti filtrą paimtą iš sluoksnio klasės į *TableContentProvider* klasę ir mano lentelėje bus tik filtrą atitinkančios eilutės.

Antrasis funkcionalumas įvykdomas pasirinkus meniu punktą „show“. Jis savo paskirtimi yra priešingas pirmajam ir truputį sudėtingesnis nei prieš tai aptartasis, kadangi filtro kriterijus reikia sukurti pagal pažymėtas lentelės eilutes. Tačiau naudojant Geotools2 bibliotekos klasę *FidFilter* tai ne taip ir sudėtinga. *FidFilter* yra speciali filtro atmaina, savyje laikanti filtruojamų geometrinių figūrų unikalius numerius. Taigi visas darbas yra tiesiog sužinoti visų pažymėtų eilučių FID laukų reikšmes ir jas sudėti į *FidFilter* klasę, bei ją nusiųsti sluoksnio objektui.

Trečiasis funkcionalumas, pats problemiškiausias. Nepaisant mano aptarto duomenų apdorojimo mechanizmo nagrinėjant GeoTools2 biblioteką. uDig turi savo nuosavą transakcijų mechanizmą. Tiesa tai versijai, kuriai aš kūriau įskiepį, ji truputį šlubavo. Transakcijos veikia tik naudojant geometrijų redagavimui. Atributnei informacijai ji tiesiog išsaugo duomenis iš karto kai jie įvedami į langelį. Na bet tai nėra taip ir blogai. Pats redagavimas nėra pastoviai įjungtas jis įjungiamas labai paprastai. Tiesiog lentelės komponentui, neperduodamas atitinkamų redaktorių sąrašas, kol vartotojas nepasirenka meniu punkto Editing->star edit, išjungiant, atvirkščiai, sąrašas pašalinamas. Neturėdamas redaktorių sąrašo lentelės komponentas neleidžia pradėti redagavimo. Tai viena iš SWT/JFace lentelės komponento savybių. Tačiau net išredagavimui įsijungus du laukai, kurių vardai FID ir the_geom, vis vien saugomi nuo redagavimo, kadangi tai specifiniai laukai, kuriuos tvarko pati GeoTools2 biblioteka, kai atliekamas geometrinių figūros parametrų keitimas.

Redagavimo metu iškyla dar viena problema. Tai duomenų tipų konvertavimas. Lentelės darbo metu, reikia atlikti dvi užduotis, pavaizduoti atributinius duomenis, o vėliau pataisytus jų variantus įrašyti į atitinkamas vietas. Pirmasis uždavinys nekelia problemų, kadangi paversti bet kokį duomenų tipą į žmogui suprantamą *String* nėra jokių problemų. Tačiau pataisytus duomenis iš *String* perversti į jų pradinį tipą yra sudėtinga, nes ši užduotis susideda iš kelių sudėtingų žingsnių. Pirmas žingsnis – nustatyti į kokį duomenų tipą reikia konvertuoti duomenis. Antras žingsnis- teisingai konvertuoti. Ir paskutinis žingsnis - įrašyti. Šis funkcionalumas realizuotas *TableCellModifier* klasėje. Metodas *verifyInpu()t* tikrina duomenų tipą ir priklausomai nuo to koks nustatytas duomenų tipas, siunčia specialiai tam duomenų tipui skirtai funkcijai, kuri patikrina ir įrašo reikšmes. Jei vis dėlto duomenys vartotojo buvo įvesti neteisingai, lentelėje apie tai nepraneša, o tiesiog palieka galioti senąją reikšmę.

Nors atrodo dauguma didelių problemų išspręsta, tačiau yra ir neįveiktų sunkumų.
Versija, kuriai skirtas įskiepis deja negali patikrinti, kokio ilgio duomenys gali būti įrašyti duomenų šaltinyje, o tai labai svarbu *String* ir *Date* duomenų tipams. Dabar palikta galioti tokia taisyklė, jog nauja *String* tipo reikšmė negali būti įvedama ilgesnė už senąją, o *Date* tipo laukai bandomi iššifruoti į reikšmes pagal nutylėjimą.

# Apibendrinimas

Didelis GeoTools2 bibliotekos funkcijų pasirinkimas ir lanksti Eclipse įskiepių sistema daro uDig programą labai patraukliu darbo įrankių. Tai ką šiame pavyzdėlyje sukūriau yra tik labai maža dalis galimybių, kurias GeoTools2 ir Eclipse sąjunga suteikia GIS programuotojui. Dėl šių savybių uDig drąsiai galima vadinti GIS platforma. Nors akivaizdu, jog kol kas uDig turi klaidų, tačiau jos nėra kritinės, jas nesunkai galima apeiti, arba pašalinti aktyvios programuotojų bendruomenės pagalba. Kaip minėjau, mano pavyzdžiai sukurti stabiliai versijai 1.0.6., tačiau šiuo metu jau galima parsisiųsti 1.1.M8 versiją, kurioje paspartintas žemėlapių piešimas bei įtrauktas ne menkas kiekis klaidų pataisymų. Spartus bibliotekos GeoTools2 ir uDig programos vystymas rodo, jog šie projektai užsitarnauja vis daugiau programuotojų simpatijų, kas anksčiau ar vėliau prives iki puikios kodo kokybės.

# uDig programinės įrangos testavimas

## Tikslas

Čia aprašomas atliktas testas, kurio metu buvo bandoma nustatyti uDig programinės įrangos gebėjimą atvaizduoti geografinius duomenis ekrane. Taip pat buvo lyginama darbo sparta naudojant informaciją, kuri randasi lokaliai kompiuterio diske ir informacija esanti PostgreSQL duomenų bazėje. Testo metu buvo stengiamasi nustatyti, kokias apkrovas patiria kompiuterių centriniai procesoriai, bei kompiuterinis tinklas. Tokiu būdu įvertinti programos keliamus reikalavimus kompiuteriui norint atlikti konkretų darbą, bei nustatyti, kiek klientų teoriškai vienu metu galėtų dirbti tinkle, nekeldami vieni kitiems nepatogumų.

## Testo įrankiai

Testavimo metu buvo naudojama paskutinė stabili uDig 1.0.6 versija. Duomenų bazė serveryje – PostgreSQL 8.0.3 versija. Testavimui naudoti duomenų pavyzdžiai randasi kompaktinio disko kataloge /data/SHP SHP failais ir PortgreSQL duomenų bazei kataloge \data\SQL. Testavimo duomenis sudaro trys sluoksniai – taškų, linijų ir plotų. Testavimo duomenų kiekis atitinka, mano praktikoje pasitaikančios realios duomenų bazės dydi. Taškinių objektų 23000; linijinių objektų 11000, ir plotų 8000 vienetų. Duomenys sluoksniuose yra išdėstyti atsitiktine tvarka. Toks testavimo duomenų modelis tūri užtikrinti realaus atliekamo darbo imitaciją.

Testavimo rezultatai fiksuojami windows aplinkoje naudojant Performance įrankį. Atliekant matavimus kas sekundę ir fiksuojant juos *.csv formato faile. Failuose įrašomi kompiuterio procesoriaus bendra apkrova %, bei tinklo bendras persiustų baitų kiekis. Užfiksuoti duomenys yra apdoroti ir perkelti į *.xls formatą, bei randasi kompakto \data\testo_duomenys\xls kataloge. Linux aplinkoje parametrams fiksuoti naudojama dstat programa, taip pat kas sekundė užrašinėjanti duomenis į failą.

Prieš atliekant testą, kompiuterių vidiniai laikrodžiai buvo suderinti tarpusavyje naudojant SNTP paslaugą. Tokiu būdu yra lengviau surasti tarpusavio kompiuterio sąveikos rezultatus sekant užfiksuotus rezultatus pagal laiką.

## Testavimo įranga

Testavimui buvo naudojami tik 2 kompiuteriai ir vienas serveris, jų parametrai:

| Nr | Procesoriaus daznis Ghz | Atminties kiekis MB | Procesoriaus pavadinimas | OS |
|---|---|---|---|---|
| 1 | 1.9 | 512 | P4 | Win2k/Linux ubuntu |
| 2 | 2.4 | 512 | P4 | Win XP |
| serveris | Du po 1 | 1000 | P3 | Win2k server |

Kompiuteriai sujungti kompiuteriniu tinklu per maršrutizatorių vitos poros kabeliu, tinklo sparta teoriškai siekia 100Mb/s, praktiškai tinklas pasiekia 8008907B/s. Deja surinkti pakankamai kompiuteriu, tam kad apkrauti pilnai tinklą, nepavyko. Sukurti programas, kurios naudotųsi lygiagrečiomis gijomis ir siųstu užklausas į serverį buvo galima, tačiau testo esmė būtent įvertinti braižymo spartą, kadangi Java aplinkos silpniausia vieta yra būtent darbas su grafine aplinka, ir ypač naudojant SWING biblioteką. Todėl tokių programų kūrimas ir

testavimas tam kad įvertinti tinklo apkrovą ar serverio pajėgumą iškraipytų duomenis, nes tokios programos nenaudotų grafinės aplinkos, ir duomenis kauptų operatyviojoje atmintyje. O jei parašyti analogiškas programas naudojančias grafinę aplinką ir atvaizduojančias braižomus objektus, kaip matysime vėliau iš testo rezultatų, daugiau nei vienos viename kompiuteryje, nebūtų įmanoma paleisti, turint omenyje naudojamą techniką, dėl per didelės apkrovos centriniam procesoriui. Todėl testuodamas apsiribojau tik šiais kompiuteriais.

## Testavimo metodika

Prieš pradedant testą, pirmiausiai užkraunama uDig programa bei nustatomi sluoksniai. Tuomet paleidžiama kompiuterio resursus fiksuojanti programa Performance, arba dstat(Linux). uDig programa iš karto nupiešia visus sluoksnius, tam kad ji iš naujo pakartotų piešimą, naudojama funkcija *redraw*, kuri randasi File->redraw, arba analogiškas mygtukas įrankių juostoje. Kadangi kompiuteris nuolat ką nors dirba, tai failuose užregistruojami apkrovos pakitimai, net tuomet, kai atrodo, jog niekas neveikia. Tam, jog būtų paprasčiau atskirti kompiuterio apkrovą sąlygojančius pašalinius veiksnius nuo testuojamos programos veikimo, programos piešimas aktyvuojamas kiekvienos naujos minutės pirmosiomis sekundėmis. Programai pabaigus piešti, laukiama apie 10 sekundžių tam, kad matytųsi aiškus apkrovos sumažėjimo tarpas užfiksuotuose duomenyse, tuomet paruošiamas programos langas naujam bandymui ir sulaukus naujos minutės pradžios, inicijuojamas sluoksnių perpiešimas. Tokia procedūra kartojama nemažiau penkių kartų. Atlikus procedūras, stabdomos kompiuterio parametrus fiksuojančios programos ir surenkami jų sukurti failai.

## Rezultatai

Surinkus programų užfiksuotus duomenis, įvertiname vidutinę kompiuterio bei tinklo apkrovą, priklausomai nuo testo tipo. Kiekviename suformuotame *.xls faile surandamos apkrovimo parametrų padidėjimo grupės, kurios prasideda ties naujos minutės pradžia, žiūrint laiko grafą excel lentelėje. Iš pradžių suskaičiuojama vidutinės vertės kiekvieno apkrovimo metu, o vėliau bendros vidutinės reikšmės, kurios surašytos ir apskaičiuotos excel lentelių pabaigoje. Suvedus duomenis į lenteles gaunami tokie rezultatai:

Kai naudojami SHP failai:

| Kompiuteris | Operacinė | Kliento kompiuterio apkrova % | Vidutinis laikas sekundėmis |
|---|---|---|---|
| 1 | Windows 200 | 83% | 9,5 |
| 1 | Linux Ubuntu 5.10 | 65% | 5,4 |
| 2 | Windows XP | 37% | 3,2 |

Kai naudojama PostgreSQL duomenų bazė:

| Kompiuteris | Operacinė | Kliento kompiuterio apkrova % | Serverio vidutinė apkrova % | Vidutinis laikas sekundėmis | Tinklo vidutinė apkrova B/s |
|---|---|---|---|---|---|
| 1 | Windows 2000 | 95% | 3,11% | 70 | 101195,17 |
| 1 | Linux Ubuntu 5.10 | 83% | 6,47% | 12,85 | 637178,7 |
| 2 | Windows XP | 66% | 7,02% | 10,83 | 796192,64 |

## Išvados

Iš pateiktų rezultatų pirmoje lentelėje, kur atvaizduojami darbo su SHP failais esančiais kompiuteryje sparta, akivaizdu, jog kompiuteris Nr1. su Win2k operacine sistema dirba labai lėtai (9,5 sekundės) dirbti su tokiu kompiuteriu yra labai sunku. Labai keista, jog tas pats kompiuteris su Linux operacine sistema dirba beveik du kartus greičiau. Tačiau ir ši sparta nepakankama. Kompiuterio Nr. 2 rezultatai žymiai geresni. Turint omenyje, jog dažniausiai dirbama tik su nedidele dalimi duomenų bazės, galima drąsiai teigti, jog su šis variantas tinkamas darbui.

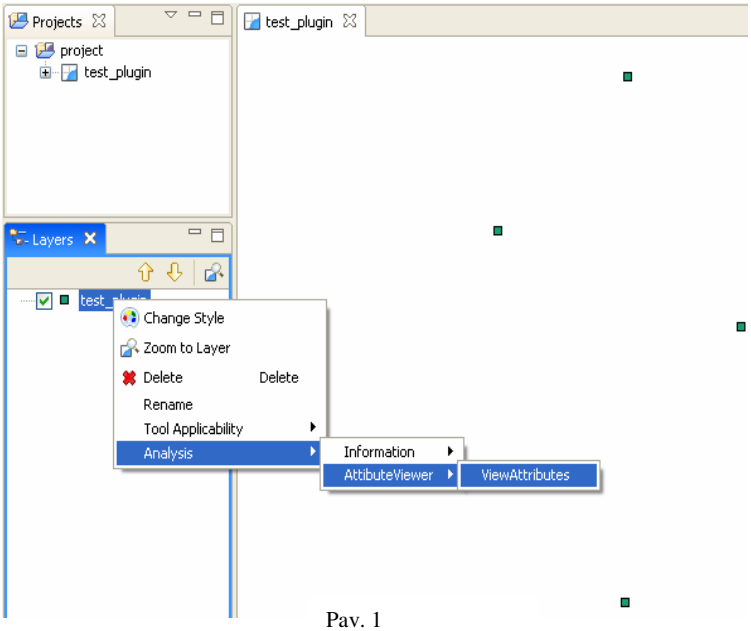
O dabar paanalizuokime darbo su duomenų baze rezultatus. Akivaizdu, jog šie rezultatai žymiai prastesni. Nei vieno kompiuterio rezultatai duotoje situacijoje nėra pakankami normaliam darbui užtikrinti. Pažvelgus į tinklo duomenis matyti, jog pats greičiausias kompiuteris išnaudoja truputį mažiau nei 10% tinklo pajėgumo, tuo tarpu serveris, galima sakyti, atostogauja. Taigi galime daryti išvadą, jog klientinės programos nesugeba greičiau apdoroti duomenų, todėl sugaištamas labai ilgas laikas. Sudalyvavus pašto konferencijoje paaiškėjo, jog šioje versijoje uDig specialiai optimizuotas SHP failams, o visiems kitiems duomenų šaltiniams piešimo metu optimizacija netaikoma. Todėl aišku kodėl SHP failų atvaizdavimas užima trumpesnį laiką. Vienintelė paguoda, jog žadama taip pat optimizuoti piešimą ir kitiems duomenų šaltiniams. Akivaizdu, jog šioje versijoje bent su mano testuotais kompiuteriais dirbti naudojant duomenų baze sudėtinga. Tačiau labai tikėtina, jog investuojant į galingesnę kompiuterinę įrangą galimi žymiai geresni rezultatai.

# AttributeViewer įskiepio Vartotojo Instrukcija

Tarkime, jog jūs teisingai įvykdėte visas įdiegimo instrukcijas ir pagaliau turite veikiančią programą uDig su įdiegtu joje įskiepiu AttributeViewer, kas lietuviškai reikštų atributinės informacijos peržiūrėjimo įskiepis. Taip pat kalbėdamas darysiu prielaidą, jog jūs mokate dirbti su uDig programa. Tai pakankamai lengva, ypač jei jau turite patirties dirbti su bet kokiomis kitomis GIS programomis. Jei ne, siūlyčiau pasidomėti tinklapyje http://udig.refractions.net/confluence/display/UDIG/Documentation esančiu Vartotojo vadovu (angl. User Documentation) skyreliu, ten rasite ir video medžiagos apie programos naudojimą. Mums reikės tik pačių elementariausių operacijų:

- ✓ Žemėlapio sukūrimo.
- ✓ Sluoksnio sukūrimo žemėlapyje.
- ✓ Pažymėti objektus sluoksnyje, atžymėti pažymėtus objektus.
- ✓ Naudotis informacijos ištraukimo, pritraukimo/nutolinimo, bei žemėlapio stumdymo įrankiais.

Paleidę programą, sukukite žemėlapį ir įdėkite naują sluoksnį. Sluoksnis gali būti bet kokio tipo, tačiau siūlau pasinaudoti kompaktiniame diske esančia informacija /data/plugin_test/test_plugin.shp failu. Tai taškinių objektų sluoksnis, jį sudaro kelios figūros, tam kad būtų lengviau orientuotis. Dabar spragtelkite naująjį sluoksnį dešiniu pelės klavišu. Pasirinkite Anglysis->AttributeViewer->ViewAttributes taip kaip parodyta Pav. 1

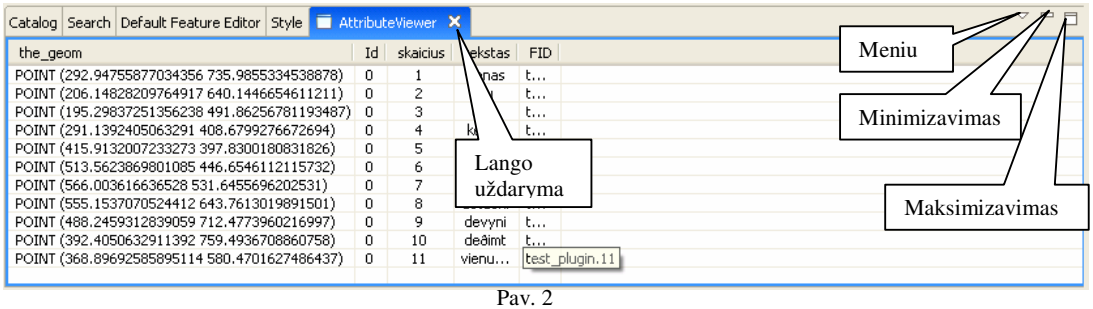



Pav. 1

Atlikus šias komandas uDig lango apačioje atsivers naujas puslapis AttributeViewer, parodytas Pav.2. Pagrindiniai lango valdymo mygtukai nurodyti paveikslėlyje. Informacija atvaizduojama lentelės pavidalu. Kiekviena lentelės eilutė - atskira žemėlapyje matoma figūra. Lentelė visada turi du laukus the_geom ir FID. Pirmasis - saugo geometrijos informaciją užkoduotą SFS (Simple Feature for SQL specifikation) numatytu WKT (Well-Known Text format) fromatu.

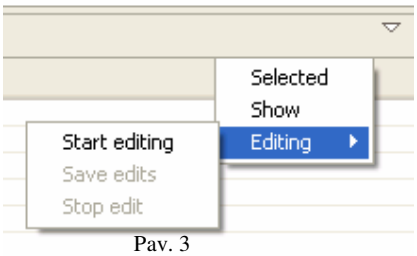
Plačiau apie tai galima rasti informacijos http://www.opengeospatial.org/docs/99-049.pdf. specifikacijoje.

Šie laukai redaguojami automatiškai keičiant figūrų geometrijos parametrus, specialių įrankių pagalba, todėl jų redagavimas įskiepyje uždraustas, visi kiti laukai vartotojo valioje. Pagal nutylėjimą lentelėje atvaizduojamos visos sluoksnyje esamos figūros. Maniu pagalba šį funkcionalumą galima lengvai valdyti.



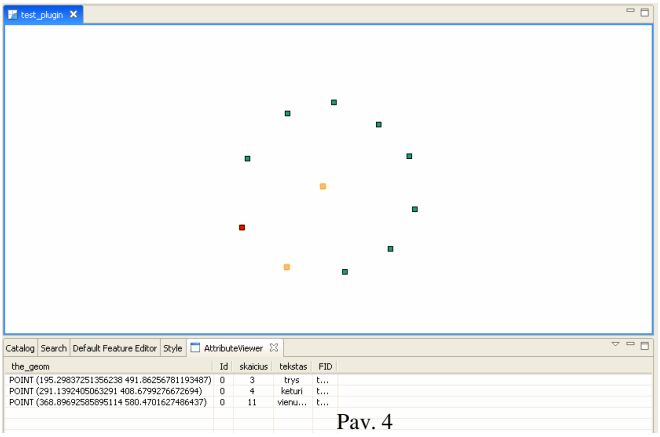
Pav. 2

**Meniu.** AtributeViewer turi specialų integruotą meniu matomą Pav.3.Jis turi du meniu punktus ir vieną submenių, kuris savo ruožtu turi dar tris meniu punktus.

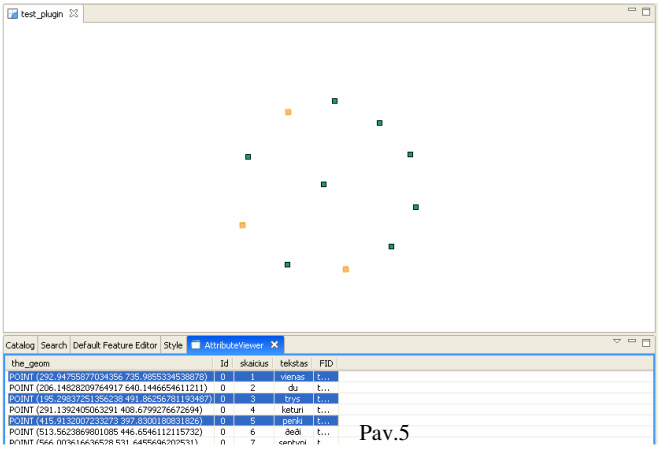


Pav. 3

**Selected** - pirmasis meniu punktas yra skirtas lentelėje atvaizduoti tik tuos objektus, kurie yra pažymėti žemėlapyje. Komandos rezultatas aiškiai matyti Pav.4. Tai labai naudinga dideliame žemėlapyje, kai lentelėje didelis skaičius eilučių, sunku rasti reikiamas eilutes. Įvykužius operaciją, meniu pavadinimas iš Selected pasikeičia į ALL. Pasirinkus meniu punktą ALL lentelėje vėl rodomi visi įrašai.


Pav. 4

**Show** – žemėlapyje pažymi lentelėje pasirinktas eilutes atitinkančias figūras. Rezultatą įvykužius šią komandą galite matyt Pav. 5 Tai padeda identifikuoti eilutę lentelėje atitinkančią figūrą ir padeda išvengti klaidų redaguojant atributinę informaciją.


Pav.5

**Redagavimas.** Šį įskiepio funkcionalumą valdo Edit submeniu punktai. Turbūt jau pastebėjote, jog kol kas nei vieno lauko reikšmės negalėjote pataisyti. Taip yra todėl, jog pagal nutylėjimą įskiepis neleidžia taisyti. Iš tiesų tai labai patogu, dažniau tenka informaciją peržiūrėti nei ją keisti. Tuo tarpu turbūt kiekvienas prisimins atveju, kai netyčia yra paspaudęs klavišą klaviatūroje, tiesiog netyčia atsirėmęs, ar uždėjęs ant klaviatūros knygą. Būtent tokiems atvejams ir yra skirta ši apsauga. Šį funkcionalumą nusižiūrėjau iš garsios kompanijos ESRI, ArcView produkto.

Norint pradėti redaguoti duomenis reikia aktyvuoti vienintelį neužblokuotą meniu punktą **Start Editing.** Dabar minėtasis punktas papilkėjo, o atsiblokavo sekantys du, buvę pilki. Taip pat dabar galite pasirinkę stulpelį kopijuoti keisti ir kitaip manipuliuoti jo turiniu. Manipuliacijos išsisaugo laukelyje, kai tik jūs iš jo išeinate. Jei lauko reikšmė nepasikeitė, vadinasi įvesta informacija neatitinka lauko duomenų tipo ar yra per ilga. Norint išsaugoti pakeitimus į sluoksnio duomenų failą, reikia iškviesti menių punktą **Editing->Save edits.** Norint baigti redagavimo režimą, reikia rinktis **Editing-> Stop edit.** Visos šios operacijos negalioja the_geom ir FID lentelės laukams dėl aukščiau minėtų priežasčių.

**Pastabos:**

- ✓ Jei AtributeViewer buvo iškviestas ne minėtu būdu, o tiesiog kaip parastas Eclipse view elementas naudojantis pagrindiniu Eclipse meniu Windows->Show View->Other, arba paskutinį kartą prieš išjungiant programą AttributeViewer langas nebuvo uždarytas ir atsidarė kartu su naujai atidaryta uDig programa, tokiu atveju lange pasirodys paprasčiausias užrašas „Nepasirinktas Sluoksnis“.
- ✓ Jei žemėlapyje yra keli sluoksniai, ir AttributeViewer įskiepio langas jau atidarytas, būtina jį uždaryti prieš naudojant įskiepį kitam sluoksniui. Kitu atveju jis nereaguos į komandą ir dirbs su prieš tai atidarytu sluoksniu.