KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Egidijus Švedas

Elektroninio pašto automatizuotos procesų sistemos sudarymas ir tyrimas

Magistro darbas

Darbo vadovas

prof. dr. E. Kazanavičius

Kaunas, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Egidijus Švedas

Elektroninio pašto automatizuotos procesų sistemos sudarymas ir tyrimas

Magistro darbas

Vadovas

prof. dr. E. Kazanavičius

2011-05

Atliko

IFM-9/1 gr. stud.

Egidijus Švedas

2011-05-27

Recenzentas

lekt. dr. A. Liutkevičius

2011-05

Kaunas, 2011

TURINYS

1. ĮVADAS

Šiandien tikriausiai nerastume tokio žmogaus, kuris nebūtų girdėjęs apie elektroninį paštą: dauguma mūsų jį naudoja kasdien, o kai kuriems tai yra ir svarbus darbo įrankis. Būtent dėl šios priežasties elektroniniam paštui reikia skirti nemažai dėmesio, o tiems, kurie per dieną gauna po kelis šimtus laiškų, tai nėra paprasta ir užima daug laiko.

Darbui su elektroniniu paštu palengvinti yra sukurta nemažai pagalbinių programų. Vienos iš jų skirtos siauram vartotojų ratui, kitos yra atviro kodo ir lengvai konfigūruojamos pritaikant vartotojo poreikiams. Tačiau vartotojams, naudojantiems Microsoft® Exchange Server® ar Microsoft® Outlook® programinę įrangą universalios ar atviro kodo elektroninio pašto sistemos dažniausiai nėra tinkamos, o apdorodama didelius elektroninės korespondencijos kiekius Microsoft® Outlook® programa neretai stringa bei reikalauja laiko el. laiškų indekso sudarymui. Tai sukelia problemų, kadangi norint greitai išsiųsti ar paskaityti svarbų laišką, kartais skubant tiesiog neužtenka laiko sulaukti, kol įjungta elektroninio pašto programa suindeksuos naujai gautus kelis šimtus laiškų.

1.1 Darbo objektas

Asmenims ar įmonėms, per dieną gaunantiems po kelis šimtus laiškų su įvairiais prikabintais elektroninių dokumentų priedais, laiškų skaitymas, jų rūšiavimas ir atsakymų rašymas atima daug laiko. Dauguma šiam darbui naudojamų populiarių elektroninio pašto programų taip pat sunkiai tvarkosi su dideliais kiekiais laiškų ir augant pašto apimčiai veikia lėtai bei nepatikimai.

Pagrindinė to problema yra laiškų saugojimui ir indeksavimui naudojama failinė sistema, kai laiškai saugomi kataloguose kompiuterio diske, o greitai jų paieškai naudojamas indeksų failas. Šiame faile saugomi tam tikra tvarka surikiuoti laiškų indeksai ir nuorodos į kiekvieną laišką. Esant dideliam kiekiui laiškų šis failas plečiasi, o norint atidaryti laišką savo pašto dėžutėje, kompiuteris kas kartą išnaudoja resursų kreipdamasis į didelės apimties failą ir jį apdorodamas. Taip pat bėgant laikui kompiuterio kietasis diskas tampa vis labiau fragmentuotas ir laiškai yra saugomi skirtinguose fiziniuose disko sektoriuose. Dėl šių priežasčių labai sulėtėja darbas su pašto sistema, ypač ieškant informacijos laiškų tekstuose.

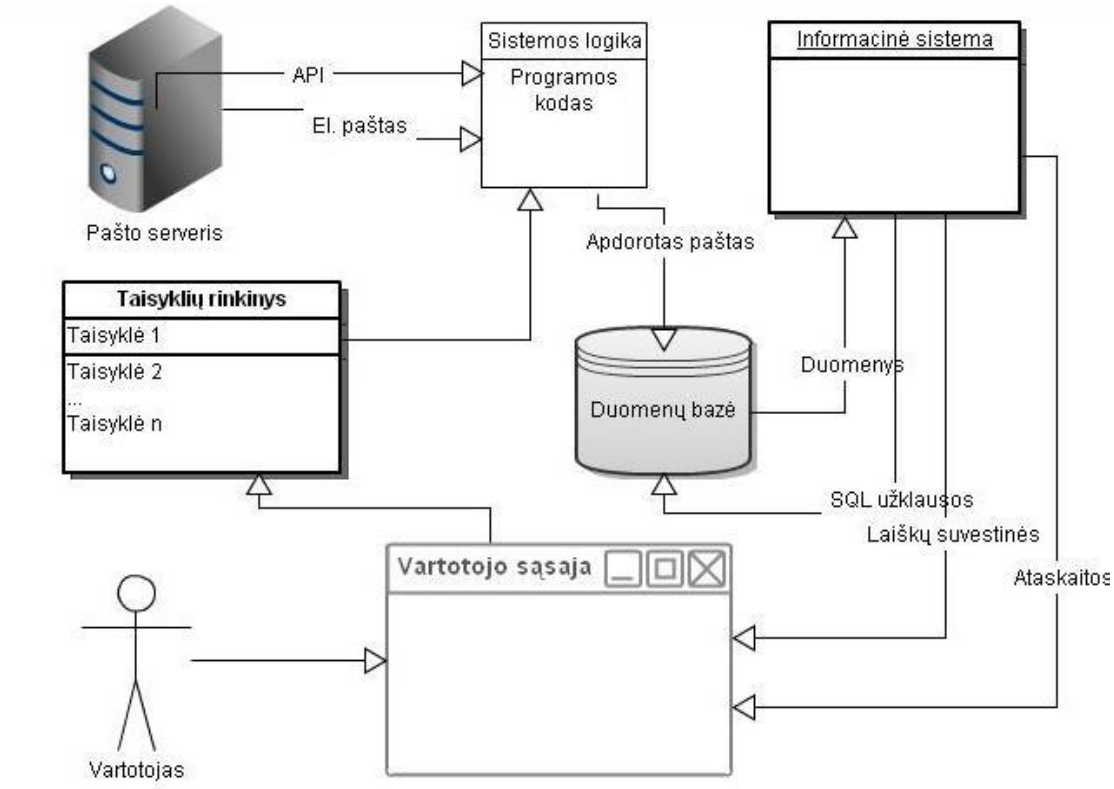
Kitas tokios failinės sistemos trūkumas yra laiškų priedų(angl. attachments) saugojimas kietajame diske, bei jų susiejimas su laiškais. Perkeliant elektroninio pašto sistemos vartotojo paskyrą į kitą kompiuterį dažnai laiškų priedai ir jų susiejimai su laiškais yra pametami.

1.2 Darbo tikslas

Pagrindinis magistro baigiamojo darbo tikslas yra ištirti aprašytų problemų egzistuojančius sprendimo būdus, išanalizuoti problemoms spręsti naudojamas technologijas bei metodus ir sudaryti bei realizuoti efektyvesnę automatizuotą elektroninio pašto procesų apdorojimo sistemą. Šiam tikslui pasiekti, magistro baigiamajame darbe keliami šie uždaviniai:

- Išanalizuoti analogiškas sistemas
- Ištirti naudojamus problemos sprendimo būdus, technologijas ir metodus
- Sudaryti kuriamos sistemos modelį
- Pritaikyti efektyvius sistemos integracijos metodus, leidžiančius pritaikyti sistemą jau naudojamoms elektroninio pašto programoms
- Sukurti taisyklių sistemą, leidžiančią klasifikuoti laiškus su minimaliu vartotojo įsikišimu
- Realizuoti sistemą pagal sukurtą modelį
- Įvertinti sistemos efektyvumą matuojant sistemos veikimo parametrus

Numatomas problemos sprendimo būdas remiasi laiškų ir jų priedų saugojimu duomenų bazėje, bei informacinės sistemos naudojimu pateikiant laiškus ir su jais susijusią informaciją vartotojui. Preliminari sistemos schema pavaizduota 1 paveiksle:



1 pav. Bendra sistemos schema

Darbo struktūra susideda iš tokių etapų:

- Analizės dalyje išanalizuojamos jau sukurtos analogiškos sistemos, išnagrinėjami naudojami aptartų problemų sprendimo būdai bei metodikos
- Sistemos modeliavimo dalyje sukuriami detalūs projektuojamos sistemos veiklos procesų ir elgsenos modeliai, kurių pagrindu sudaromas informacinės sistemos duomenų modelis, numatomi sistemos integracijos su vartotojo naudojama programine įranga metodai, modeliuojama elektroninių laiškų tvarkymo taisyklių sistema
- Eksperimento dalyje atliekama praktinė sistemos realizacija, tiriama jos integracija su vartotojo naudojama programine įranga, įvertinami sistemos veikimo parametrai, tiriamas SQL užklausų efektyvumas naudojantis duomenų bazėje saugomu elektroniniu paštu

2 ANALITINĖ DALIS

2.1 Analogiškų sistemų analizė

Įskiepis Xobni[1] -tai nedidelis, turintis nemokamą versiją, Microsoft® Outlook® įskiepis, kuris naršo po pašto dėžutę ir palengvina bei pagreitina laiškų paiešką, bei suranda informaciją apie kontaktus, su kuriais susirašinėjama, naudodamas socialiniuose tinkluose („Facebook“, „Twitter“, „Linkedin“) saugomus duomenis. Xobni® siūlo išplėstas paieškos galimybes, geresnį pašto dėžutės rūšiavimą ir kontaktų integraciją su socialiniais tinklais. Jis neturi jokios gilesnės analizės ar bendravimo elektroniniais laiškais tyrimų ir stipriai remiasi tik integracijos su socialiniais tinklais standartais. Visgi, socialinių tinklų panaudojimo elektroniniame pašte idėja yra gana inovatyvi ir gali turėti didelį potencialą įmonės kontekste, jei socialiniai tinklai yra integruoti įmonės aplinkoje.

Pagrindinė akcentuojama Xobni® savybė yra greita ir efektyvi paieška[2]. Ji suteikia vartotojui galimybę surasti elektroninius laiškus įvedant raktinius žodžius ar asmens vardą. Norint rasti konkretesnius laiškus, Xobni® siūlo išplėstinę paiešką: čia galima įvesti raktinius žodžius, kurių bus ieškoma tik „Kam:“, „Nuo:“, „Tema:“ ir pan. laukeliuose arba tarp prie laiškų prisegtų bylų. Paieška vykdoma iškart vos vartotojui įvedus pirmąjį raktinį simbolį, todėl kartais rezultatus galima gauti dar nespėjus iki galo įvesti norimos užklausos. Įskiepis turi ir papildomų funkcijų, pvz.: įvairi statistika(gaunamo pašto kiekio, pašto gavimo laiko), informaciją apie kontaktus, susirašinėjimų istoriją, prie laiškų prisegtų bylų apžvalgą. Mokama versija „Xobni Plus®“ taip pat turi susirašinėjimų filtravimo funkciją, kuri leidžia matyti susirašinėjimus tik su pageidaujamais kontaktais. Žinoma, įskiepis turi ir įprastas pašto programų funkcijas, tokias kaip nepageidaujamų laiškų filtravimas, katalogizavimas ir kita.

Sistema Xobni nesukuria savo saugyklos elektroniniam paštui, o tik indeksuoja duomenis, jau esančius Microsoft® Outlook® programoje[3]. Šie indeksai saugomi Xobni sukurtoje byloje, kuri plečiasi augant laikų skaičiui. Jei Outlook® nuolatos kreipsis į pašto serverį ieškodamas naujų laiškų, tai privers Xobni® nuolatos iš naujo indeksuoti laiškus, kas pareikalaus papildomų procesoriaus ir tinklo resursų. Taip pat, veikiant skirtingose operacinėse sistemose ir stipriai augant elektroninio pašto apimčiai, paieška ir ypač laiškų indeksavimas trunka juntamai ilgiau.

Šis įskiepis suderinamas su Microsoft® Outlook® 2003, 2007 ir 2010 versijomis. Sistemos Xobni didžiausias privalumas yra paieškos greitis ir funkcionalumas, o kaip trūkumus galima išskirti savos saugyklos neturėjimą ir „failinės sistemos“ naudojimą, kai laiškai ir jų indeksai saugomi bylose. Microsoft® Outlook® įskiepis su pagrindinėmis funkcijomis yra nemokamas. Sistemos versija Xobni Plus® vienam kompiuteriui kainuoja 29.95$ (~86 Lt).

Įskiepis Lookeen - tai dar vienas Microsoft® Outlook® įskiepis, skirtas pagreitinti paiešką šioje programoje. Lookeen® yra paieškos įrankis, kuris nesunkiai integruojamas į Outlook® ir gali ieškoti duomenų tiek tarp laiškų ir jų priedų, tiek tarp kontaktų, užduočių ar užrašų.

Akcentuojama, kad Lookeen® randa duomenis greičiau nei standartinė Outlook® paieška. Įrankis pateikia paieškos rezultatus atskirame lange, taip pat sugeba grupuoti rezultatus pagal norimą kriterijų (datą, siuntėją, gavėją, temą), taip pat galima surasti susijusius laiškus ar susirašinėjimus su konkrečiu asmeniu[4]. Surastus laiškus galima nesudėtingai perkelti, nukopijuoti, ištrinti, peržiūrėti. Įdiegta ir išplėstinė paieška, leidžianti ieškoti informacijos konkrečiose elektroninio laiško dalyse. Yra greitos peržiūros galimybė, kuri leidžia greitai atidaryti prie laiškų prisegtus dokumentus ar paveikslus, taip atrandant sau reikalingą. Įskiepis taip pat turi nesudėtingų statistikų skaičiavimo galimybę (kuriomis savaitės dienomis gaunama daugiausia laiškų, kiek pasikeitė laiškų srautas per metus ir kita). Įdiegus Lookeen, Microsoft® Outlook® įrankių juostoje atsiranda papildoma Lookeen juosta, kurioje yra greitos paieškos mygtukai leidžiantys iškart atrinkti konkrečios dienos ar praėjusios savaitės korespondenciją.

Kaip ir prieš tai apžvelgtas įskiepis, Lookeen® neturi savo duomenų saugyklos, o tik kuria į Outlook® įkeltų laiškų indeksą. Laiškai nėra indeksuojami realiu laiku[5]. Laiškų indeksas nėra atnaujinamas išsiuntus ar gavus naują laišką, perkėlus ar ištrynus jau egzistuojantį. Įrankis leidžia nustatyti tik reguliarų pašto indeksavimą nustatytu laiko intervalu. Tiesa, galima šį intervalą nustatyti ir keletui sekundžių, tačiau tada dažnai bus be reikalo eikvojami kompiuterio resursai.

Šis įskiepis suderinamas su Microsoft® Outlook® 2003, 2007 ir 2010 versijomis. Pastebėti šie įskiepio privalumai: greita kontaktų, laiškų ir įvykių paieška, lengvai pasiekiami susiję laiškai ir susirašinėjimai. Didžiausi trūkumai: „failinė sistema“, kai laiškai ir jų indeksai saugomi bylose bei laiškų indeksavimo realiu laiku nebuvimas. Įskiepio Lookeen

2010 Professional® versijos kaina vienam kompiuteriui - 120 Lt. Taip pat siūloma bandomoji versija veikianti 14 dienų.

2.2 Naudojamų technologijų analizė

Pradinis laiškų apdorojimas. Elektroniniai laiškai paprastai siunčiami kaip grynasis tekstas(angl. plain text) arba kaip tekstas su HTML žymomis. Dauguma sistemų traktuoja laišką kaip žodžių konteinerį ir todėl pirmiausia atskiria nuo teksto visas HTML žymas, jei laiškas nėra grynasis tekstas[6]. Kai kurios sistemos atskirai apdoroja pašto antraštes, sukurdamos atskiras talpyklas kiekvienai antraštei. Pavyzdžiui, „Nuo“ ir laiško kūno dalys talpinamos skirtinguose konteineriuose, skirtuose būtent šioms laiškų dalims, kuriuose žodžiai kartojasi, jei yra sutinkami keletą kartų.

Skaidant laiškus, visi simboliai konvertuojami į tą patį lygmenį, t.y. panaikinami skirtumai tarp didžiųjų ir mažųjų raidžių, panaikinami skyrybos ženklai ir visas tekstas suskaidomas į jį sudarančių žodžių rinkinį[6,8]. Tada vykdomas procesas, kurio metu žodžiai konvertuojami į savo šaknines formas, pašalinant priesagas ir priešdėlius. Tokiu būdu konteineryje sumažėja unikalių žodžių. Šiam procesui įgyvendinti dažnai naudojamas Porter Stemmer algoritmas[7], pavyzdžiui žodžių „pristatymas“, „pristatyti“, „pristatytas“ suvedimui į vieną šakninę formą.

Pradiniame laiškų apdorojime dažnai naudojamas ir ignoruojamų žodžių sąrašas. Natūrialioje kalboje nemažai žodžių yra sutinkami dažnai, bet suteikia mažai informacijos. Tai įvairūs jungtukai, jaustukai ištiktukai ir panašūs žodeliai, tokie kaip „o“, „ir“, „bei“, „štai“ ir pan. Šie žodžiai gali būti pašalinti iš teksto darant prielaidą, kad dėl to nebus prarasta dalis informacijos. Šie ignoruojamieji žodžiai yra nurodyti specialiame sąraše ir gali būti šalinami iš apdorojamo elektroninio laiško.

Dažniausiai po pradinio apdorojimo pasitaikantis elektroninio laiško atvaizdavimas yra atributų vektorius[6]. Kiekvienas unikalus žodis esantis konteineryje yra atributas. Kiekvienam atributui yra priskiriama skaitinė reikšmė, kuri yra žodžio pasikartojimų konteineryje skaitliukas. Tokiu būdu, iš laiškų kolekcijos klasifikavimui yra perduodami vektoriai su pasikartojimo skaitliukais.

Integracijos technologijos. Pranešimų aplikacijos programavimo sąsaja (angl. Messaging Application Program Interface, MAPI)[19] yra Microsoft® Windows® programavimo sąsaja, kuri leidžia siųsti elektroninius laiškus iš bet kurios Windows® aplikacijos ir naudotis kitomis Outlook® funkcijomis. Aplikacijos, kurios gali pasinaudoti MAPI gali būti tiek teksto rengyklės, tiek skaičiuoklės, tiek grafikos aplikacijos ar kitos.

Sąsaja susideda iš standartinio C# programavimo kalbos funkcijų rinkinio[20], kurios saugomos programų bibliotekoje, žinomoje kaip dinaminių nuorodų biblioteka (angl. dynamic link library, DLL). MAPI sąsaja yra komponentiniu objektų modeliu paremta API (Aplikacijos programavimo sąsaja, angl. Application Program Interface). Ji leidžia klientinėms programoms naudotis savo procedūromis, kurios sąveikauja su pašto serveriais. Nors MAPI yra suprojektuota nepriklausomai nuo naudojamo protokolo, paprastai ji naudojama su MAPI/RPC protokolu, kurį Microsoft® Outlook® naudoja bendravimui su Microsoft® Exchange®. Bazinė MAPI yra 12-kos funkcijų poaibis, kurios leidžia programuotojams naudoti pagrindinį elektroninio pašto funkcionalumą. Išplėstinė MAPI leidžia visišką elektroninio pašto sistemos kontrolę kliento kompiuteryje: elektroninių laiškų kūrimą ir tvarkymą, kliento pašto dėžutės tvarkymą, paslaugų teikėjų nustatymų keitimą ir kita. Bazinė MAPI yra pateikiama su Microsoft® Windows®, o išplėstinė – su Microsoft® Outlook® ir Exchange®.

2.3 Elektroninių laiškų klasifikavimo metodų/būdų analizė

Elektroninių laiškų klasifikavimui dažnai naudojami tobulėjantys (angl. learning) algoritmai. Vienas iš dažnai naudojamų yra RIPPER algoritmas. Šis algoritmas aprašytas Cohen(1996), buvo pasiūlytas naudoti su dideliais duomenų rinkiniais[8]. Metodas, naudodamasis žodžių konteineriais ir atributais, kuria tam tikras taisykles, pvz:

$$kt22 \leftarrow \text{'rimas'} \in from \cup \text{'rimas'} \in to$$

Ši taisyklė reiškia, kad laiškas priklauso katalogui „kt22“, jei žodis „rimas“ yra tiek „Nuo“, tiek „Kam“ antraštėse. Tokios taisyklės yra labai tinkamos elektroninio pašto klasifikavimui ir filtravimui, nes daugelis yra jau pritaikytos tokio tipo klasifikavimo taisyklėms, tačiau tokios taisyklių mokymosi sistemos integravimas į egzistuojančias elektroninio pašto programas yra sudėtingas dėl to, kad reikia konvertuoti tokias sistemines taisykles į taisykles suprantamas vartotojui.

TF-IDF (Term Frequency – Inverse Document Frequency) yra dar vienas populiarus klasifikavimo metodas, išsivystęs iš tekstų klasifikavimo[10]. Šio metodo įėjime naudojamas savybių dažnio vektorius. Algoritmas naudoja TF ir IDF kaip du įverčius, kad numatytų savybių (šiuo atveju žodžių) svorį. Termino dažnumas (angl. Term Frequency) vaizduoja žodžio pasikartojima aplanke, o dokumento dažnumas (angl. Document Frequency) rodo žodžio reikšmingumą visoje vieno vartotojo paskyroje. Kiekvieno aplanko panašumo įvertis apskaičiuojamas iš ateinančių laiškų bei IF ir IDF savybių vektorių. Aplankas, kuris turi didžiausią panašumo įvertį laikomas geriausia kategorija naujai gautam laiškui. Tačiau TF ir IDF išraiškos skirtinguose klasifikatoriuose gali gana stipriai skirtis.

Elektroninių laiškų klasifikavimui į pageidaujamus ir brukalus(spam) dažnai naudojamas paprastasis Bajeso(Naive-Bayes)[6,8] metodas. Jo matematinė formuluotė yra tokia[9]:

$$\Pr(S|W) = \frac{\Pr(W|S) \cdot \Pr(S)}{\Pr(W|S) \cdot \Pr(S) + \Pr(W|H) \cdot \Pr(H)} \quad , \qquad (1)$$

kur:

Pr(S|W) – tikimybė, kad laiškas yra brukalas, jei jame randamas nagrinėjamas žodis;

Pr(S) – bendra tikimybė, kad bet kuris laiškas yra brukalas;

Pr(W|S) – tikimybė, kad nagrinėjamas žodis randamas brukaluose;

Pr(H) – bendra tikimybė, kad bet kuris laiškas nėra brukalas;

Pr(W|H) – tikimybė, kad nagrinėjamas žodis randamas ne brukaluose.

Šis klasifikatorius naudoja skaitliukus žodžių konteineriuose, tam, kad nustatytų tikimybę ar apdorojamas laiškas yra brukalas. Šiam tikslui yra sudaromas pradinis dažniausiai brukaluose pasitaikančių žodžių sąrašas, kurį metodas veikdamas vis praplečia, taip vis tiksliau galėdamas filtruoti nepageidaujamus laiškus. Kai žodynai galutinai sudaryti, tikimybė, kad naujas laiškas yra brukalas, apskaičiuojama pagal Bajesą kiekvienam žodžiui iš laiško. Normalizuojant ir sumuojant žodžių tikimybę gaunama bendra tikimybė, pagal kurią galima priskirti laišką brukalams. Po mokymo ir didelio įdirbio pasiseka atrinkti iki 95 –97 % brukalų[9].

Teigiama, kad Support Vector Machines(SVM) algoritmas[6] yra tikslesnis už kitus klasifikavimo metodus. Jo įėjime naudojamas dvejetainis savybių vektorius, o esminė metodo idėja yra ta, kad SVM randa hiperplokštumą, kuri geriausiai padalina duomenų taškus į dvi klases, kur klasė nusako ar laiškas yra brukalas ar ne. Įrodyta, kad SVM[16] yra vienas

geriausių teksto klasifikavimo algoritmų. Jis yra pagrįstas Structure Risk Minimization principu iš skaičiavimų teorijos. Pagrindinė idėja yra rasti hipotezę, kuriai galime garantuoti mažiausią paklaidą atskiriant teigiamus ir neigiamus pavyzdžius. Teigiamais pavyzdžiais laikomi tinkami žodžiai, neigiamais – netinkami. Tiesiniame SVM metode, naudojama lygtis w*x-b = 0, kuri reiškia hyperplokštumą, kurioje normalinis vektorius w ir konstanta b yra apibrėžiami distancija nuo hyperplokštumos iki artimiausio teigiamo ir neigiamo pavyzdžio.

Elektroninio pašto rūšiavimas. Yra daug technologijų ir būdų kaip išspręsti elektroninių laiškų katalogizavimo uždavinius. Neapdorotas elektroninis paštas dažnai yra netvarkingas ir be jokios struktūros. Reikia atlikti daugybę valymo, pradinio apdorojimo ir organizavimo žingsnių, kol galima apmokyti klasifikatorius. Taip pat reikia atsižvelgti ir į našumo problemas, kadangi ne visi standartiniai teksto klasifikavimo metodai yra tinkami elektroninių laiškų katalogizavimui ir juos naudojant el. korespondencijos apdorojimo laikas gali išaugti.

Neteminiai katalogai. Katalogas yra laikomas neteminiu, jei jame laiškai yra saugomi nepriklausomai nuo jų turinio. Į šią kategoriją patenka tokie aplankai kaip Gauta, Išsiųsta, Šlamštas ir pan. Šaltinis[14] teigia, jog yra efektyviau atsisakyti tokių aplankų, kadangi jokia automatizuota sistema nepadeda vartotojui klasifikuoti laiškų į šiuos aplankus. Neteminiai katalogai skirstomi į tris pagrindines kategorijas:

- Automatiškai elektroninio pašto programos sukurti aplankai. Čia patenka pvz.: „Sent Items“ sukuriamas MS Outlook® programos.
- Archyvų aplankai, kurie standartiniai vartotojams, priklausantiems tam tikroms organizacijoms. Pvz.: „Visi dokumentai“ arba „Diskusijos“.
- Archyvų aplankai, sukurti paties vartotojo. Tai gali būti toks atvejis, jei, pavyzdžiui, vartotojas kuriuo nors metu tampa nebepajėgus peržiūrėti viso elektroninio pašto srauto, todėl nusprendžia tam tikrą dalį neperžiūrėtų laiškų perkelti į atskirą katalogą vėlesniam apdorojimui.

Pirmųjų dviejų kategorijų aplankai yra panaikinami. Trečiosios kategorijos aplankai nenaikinami todėl, kad tai reikalauja konkrečios vartotojo katalogizavimo strategijos pažinimo, kuri yra skirtinga kiekvienam vartotojui.

Klasifikatorių apmokymas. Daugumoje klasifikatorių nustatymų, esančio elektroninio pašto padalijimas į dalis skirtas algoritmų apmokymui ir testavimui vykdomas atsitiktinai. Deja, praktikoje elektroninių laiškų kiekis nuosekliai auga, todėl atsitiktinis padalijimas gali

sukelti nenatūralias vėlesnės korespondencijos priklausomybes nuo ankstesnės. Taigi, logiškesnis padalijimas būtų pagrįstas laiku: algoritmas apmokomas naudojant senesnius laiškus ir išbandomas naudojant naujesnius. Tokį metodą naudoja Klimt ir Yang[13]: klasifikatorius apmokomas atrenkant pusę anksčiau gautų laiškų iš elektronio pašto aplanko, o išbandomas likusioje naujesnių laiškų dalyje. Rimta šio metodo problema gali iškilti, kai apdorojamų laiškų kiekis yra pakankamai didelis: temos, apie kurias buvo diskutuojama esamu metu gali neturėti nieko bendro su temomis aptartomis praeityje. Kita problema yra ta, kad kaikurie pašto aplankai praeityje net neegzistavo. Autoriai[14] siūlo naudoti nuoseklią laiko skalės atžvilgiu metodiką: išrūšiavus paštą pagal gavimo datą, laiškų klasifikavimo algoritmas apmokomas atrenkant N laiškų ir išbandomas naudojant N sekančių laiškų. Tada vėl apmokomas imant 2N laiškų ir išbandomas su sekančiais N laiškų, kol galiausiai apdorojami visa elektroninio pašto dėžutėje esanti korespondencija. Toks metodas užtikrina praktišką ir aiškų duomenų padalijimą, kuris leidžia stebėti klasifikatoriaus našumą didėjant laiškų apimčiai.

Yra keletas galimų būdų elektroninių laiškų išrūšiavimui. Šiuo atveju, naudojamas įprastas žodžių krepšelio metodas: pranešimai atvaizduojami kaip žodžių skaičiaus vektoriai. Žodžiai yra traktuojami kaip abėcėlės, skaitinių ir specialiųjų simbolių sekos, kurios gali būti tiek laiško antraštėje, tiek jo tekste. Žodžių raidės yra paverčiamos mažosiomis, 100 dažniausių ir tik po kartą pasitaikantys žodžiai yra atmetami, o visi likę kiekviename laiške yra skaičiuojami ir taip sudaro vektorius. Laiškų priedai taip pat pašalinami. Vėlesniuose etapuose galima pritaikyti detalesnį apdorojimą, pavyzdžiui galima atskirai apdoroti skirtingas elektroninių laiškų dalis. Sistema gali atskirai skaičiuoti žodžius, pasirodančius laiškų antraštes, jų turinyje, parašuose ir pan. Klimt ir Yang[13] traktuoja laiško antraštės „Nuo“, „Kam“, ir „Kopija“ dalis kaip atskiras ir akcentuoja „Nuo“ dalies svarbą vektorių sudarymui.

Toliau, kad būtų galima nustatyti kokiam aplankui turi priklausyti laiškas, yra naudojamas tradicinis tikslumo medodas. Bendru atveju, elektroniniai laiškai gali su tam tikra tikimybe priklausyti keliems aplankams, todėl negalima iš karto priskirti laiško tam tikram aplankui, o būtina išrikiuoti visus aplankus pagal tinkamumą konkrečiam laiškui. Kai pasiekiamas toks rikiavimas, sistema gali tiksliau spręsti, kuris katalogas yra tinkamiausias tam tikram laiškui. Elektroninio pašto programos veikimas katalogizavimo atžvilgiu pilnai priklauso nuo naudojamo scenarijaus, t.y. vartotojas gali skaityti laiškus bendrame „Gauti“ kataloge, gali katalogizuoti laiškus pagal poreikį arba sistema sistema gali automatiškai

katalogizuoti laiškus, iš karto sudėliodama elektronius laiškus į atskirus aplankus, kad vartotojas galėtų skaityti gautą korespondeciją pagal kategorijas.

Latentinis Semantinis Indeksavimas[16] (angl. LSI – Latent Semantic Indexing) yra svarbus informacijos išgavimo metodas, kuriame galima automatiškai transformuoti originalius tekstinius duomenis į mažesnę semantinę erdvę, pasinaudojant kai kuriomis akivaizdžiomis ar nematomomis žodžių susiejimo su tam tikrais objektais struktūromis, o taip pat šis metodas buvo sėkmingai pritaikytas ir teksto klasifikavimui. LSI gali išspręsti daugiareikšmiškumo ir sinonimiškumo problemas ir sumažinti triukšmus neapdorotoje dokumentų-termų matricoje. Tačiau teksto klasifikavimui LSI nėra optimalus, todėl, kad jis yra visiškai neprižiūrimas metodas, kuris ignoruoja kategorijų svarbą ir jam dažnai pritrūksta našumo klasifikuojant tekstą, kuomet metodas taikomas visiems apmokymui skirtiems dokumentams[17]. Norint sukurti LSI modelį, pirmiausia reikia sudaryti apmokymo el. laiškus atvaizduojančią matricą. Šioje sistemoje, laiškas traktuojamas kaip dokumentas, o dokumentas atvaizduojamas vektoriumi linijinėje *n* žodžių ar termų erdvėje. Šie žodžiai ar termai yra paimami iš dokumentų. Panašiu būdu, žodis atvaizduojamas vektoriumi *m* dokumentų tiesinėje erdvėje. Dokumentai suformuoja dokumentų vektorių erdvę, dokumentų termas yra atvaizduojamas X matrica, kur kiekvienas stulpelis X_i atitinka žodį ar termą, o kiekviena eilutė X_j atitinka dokumentą:

$$X=\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix}=(X_1,\ X_2,\ \cdots;\ X_n)=\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{pmatrix}$$

Matricos elementas yra termo *j* dažnumas(TF) dokumente *i*, o dokumento dažnumas (DF) yra skaičius dokumentų, kuriuose sutinkamas termas *j*.

LSI yra dimensijos mažinimo metodas. Jis automatiškai skaičiuoja prasmingas semantines asociacijas poaibyje, kuris yra daug mažesnis nei pradinė imtis. Termų-dokumentų matricai X pritaikomas Singular Value Decomposition (SVD) metodas. Jei X yra $m_x n$ matrica, tada X matricos SVD yra

$$X = USV^T,$$

kur U yra $m_x n$ su ortonormuotais stulpeliais, V yra $n_x n$ su ortonormuotais stulpeliais, o S yra įstrižainė, su mažėjimo tvarka išrikiuotomis pagrindinės įstrižainės reikšmėmis.

Tokiu skaičiavimu gaunama geriausia k eilės originalios matricos aproksimacija. SVD dažniausiai apskaičiuojamas naudojant iteratyvias technikas, tokias kaip Lanczos metodas.

Tinkamumo kategorijų metodas. Straipsnio[18] autoriai pastebi tokius aukščiau aptartų metodų trūkumus:

- Pastovaus algortimų apmokymo poreikis, dinamiškai kintant laiškų kategorijoms.
- Klasifikavimo klaidos priklauso nuo vartotojo(kaip jis apmoko algoritmą).
- Gali nepakakti duomenų visų galimų variantų apmokymui.
- Vartotojui yra sudėtinga suprasti ir rankiniu būdu keisti klasifikatorių.

Aprašyti trūkumai bet kokiam siūlomam laiškų rūšiavimo sprendimui iškelia keletą reikalavimų. Pirmiausia, technologija turi būti greita ir sugebėti apsimokyti realiu laiku. Antra, sistema turi daryti kaip įmanoma mažiau klaidų arba veikti tokiu būdu, kad klaidos pastebimai neįtakotų naudojimosi sistema. Trečia, metodas turi veikti patenkinamai, net jeigu yra nedaug duomenų. Šioms problemoms išspręsti siūlomas tinkamumo kategorijų(angl. Relevance categories) metodas[18]. Pagrindinė šio metodo idėja yra užtikrinti tokį pat funkcionalumą kaip įprasti aplankai ar kategorijos. Vartotojai gali priskirti laiškus kategorijoms ar juos iš kategorijų pašalinti, kaip jie ir yra įpratę daryti. Tinkamumo kategorijų skirtumas yra tas, kad visiems esamiems laiškams yra formuojama užklausa, pagrįsta elementais, kuriuos vartotojas jau priskyrė kiekvenai kategorijai. Užklausos rezultatai rodomi atskirame lange ar rėmelyje, surikiuoti pagal kategorijų tinkamumą. 2-ame paveiksle pavaizduotas tinkamumo kategorijų prototipas. Viršutiniame langelyje matomas sąrašas laiškų, kuriuos vartotojas jau pridėjo į kategoriją. Apatiniame langelyje rodomi laiškai, kuriuos sistema randa kaip susijusius su viršuje esančiais. Šie laiškai surikiuoti pagal tinkamumą kategorijai.

2 pav. Tinkamumo kategorijų realizacijos pavyzdys

Šis metodas aproksimuoja kategorijas ir leidžia vartotojui išsaugoti pastovias užklausas pašto duomenų bazėje. Jei tinkamumo algoritmas yra idealus, tada visi tinkami elementai kurie priklauso tam tikrai kategorijai bus išrikiuoti pirmi, o mažiausiai tinkami – paskutiniai. Gija, data, siuntėjas ar kiti laukai taip pat gali būti naudojami tinkamų elementų rikiavimui, taip padidinat duomenų prieinamumą. Tokio metodo privalumai prieš tradicinius klasifikatorius yra šie:

- Tinkamumo koeficiento apskaičiavimo greitis priklauso nuo naudojamo tinkamumo algoritmo greičio.
- Klaidos yra toleruotinos vartotojo. Kol elementai tinkami kategorijai yra rikiuojami netoli nuo sąrašo viršaus, vartotojas nesunkiai juos suras. Keletas netinkamų elementų tik nestipriai įtakoja algoritmo paklaidą. Panašiai elgiasi ir internetiniai paieškų varikliai.
- Tinkamumo kategorijos garantuoja esamų kategorijų ir aplankų išsaugojimą. Kadangi tinkamumo kategorijos yra tik priedas prie egzistuojančių kategorijų, jos gali būti ignoruojamos ir naudojamos tik įprastos kategorijos, neįtakojant sistemos našumo. Tuo tarpu netinkamai veikiantis klasifikatorius gali apsunkinti naudojimąsi aplanku lyginant su aplanko naudojimu be klasifikatoriaus.
- Tinkamumo koeficientus galima apskaičiuoti net jei ir yra nedaug duomenų. Duomenų kiekiui augant, kartu auga ir tinkamumo koeficientų nustatymo tikslumas.

Tinkamumo kategorijos gali būti įgyvendintos skirtingais būdais, jei tenkinamos aukščiau paminėtos sąlygos.

Pašto klasterizavimas. Yra įrodyta, kad elektroninio pašto klasterizavimas į prasmingas grupes gali stipriai supaprastinti laiškų apdorojimo procesą. Anksčiau apžvelgtas laiškų skirstymo į katalogus metodai gali palengvinti pašto organizavimą, tačiau vartotojas turi pats sukurti prasmingus katalogus, į kuriuos bus skirstomi laiškai. Elektroninio pašto klasterizavimas nusako principą, kaip šiuos katalogus sistema gali sukurti pati, be vartotojo įsikišimo.

Modelis ESVM (angl. Email Vector Space Model)[15] yra sukurtas specialiai elektroninio pašto apdorojimui. Pirmiausia, iš laiško atskiriamas temos laukas ir laiško turinys, tada, jei jie buvo užkoduoti, atkoduojami ir pašalinami formatavimo simboliai. Iš laiško turinio atrenkami dažniausiai jame naudojami žodžiai, traktuojami kaip laiško charakteristikos. Žodžio (w) svoris (b) matuojamas naudojantis formule:

$$b_{w_i} = f_{w_i} \Big/ \sum_{j=1}^{n} f_{w_j} ,$$

kur „f“ reiškia žodžio dažnumą tekste.

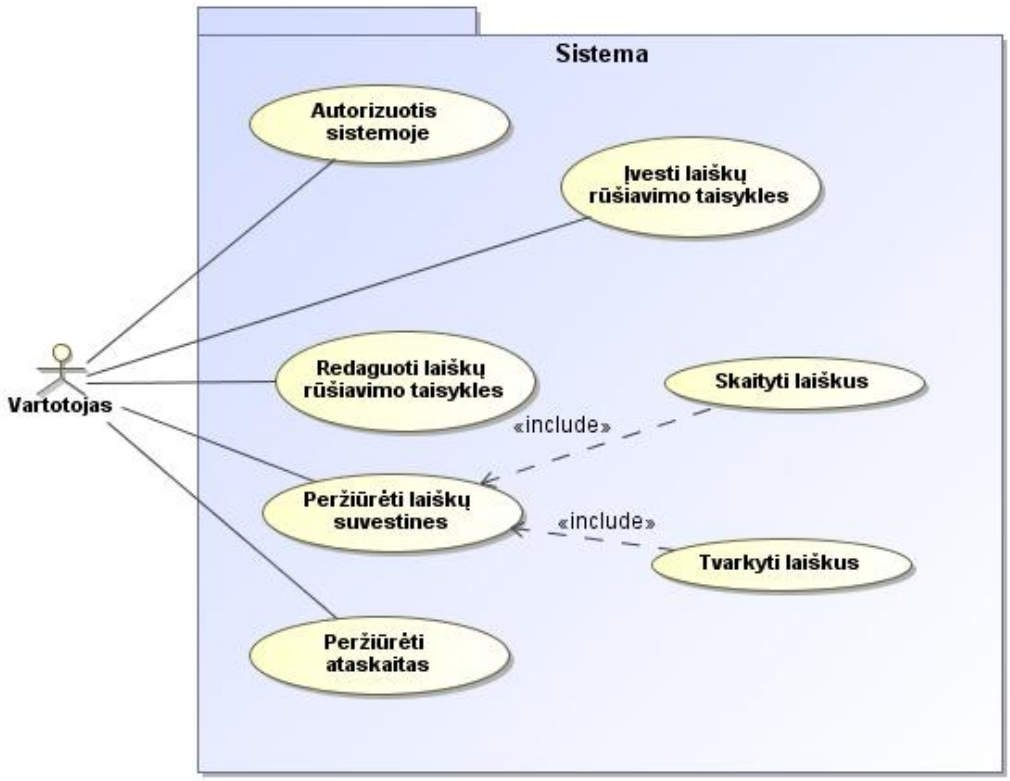
3 SISTEMOS MODELIS – STRUKTŪRA

3.1 Sistemos modeliai

Šioje sistemos modeliavimo dalyje sudaromi pagrindiniai panaudojimo atvejų, veiklos procesų ir sistemos elgsenos modeliai. Modeliai atvaizduoja sistemos elementų tarpusavio ryšius ir procesų vykdymo sekas.

3.1.1 Panaudojimo atvejų modelis

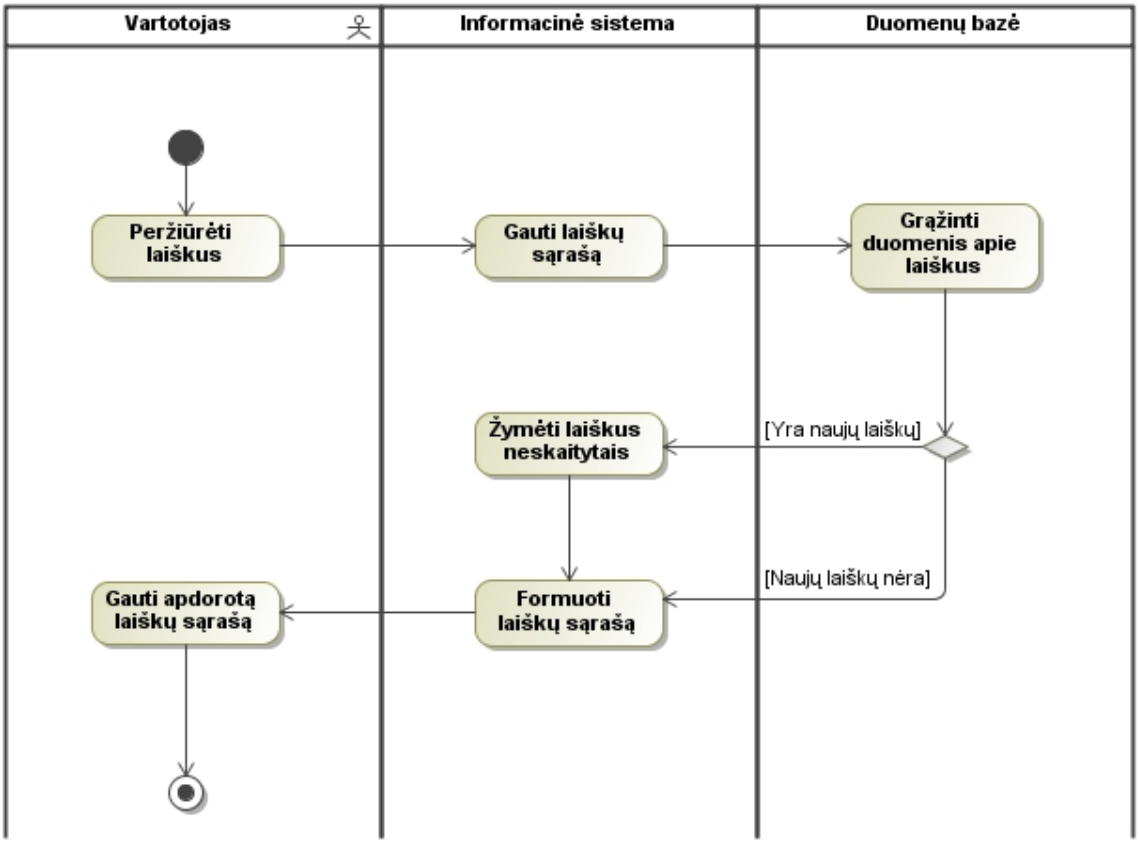
Pagrindiniai projektuojamos sistemos panaudojimo atvejai pavaizduoti 3-iame paveiksle: Autorizuotis sistemoje; Įvesti laiškų rūšiavimo taisykles; Redaguoti laiškų rūšiavimo taisykles; Peržiūrėti laiškų suvestines(šis PA apima ir kitus du PA: Skaityti laiškus ir Tvarkyti laiškus); Peržiūrėti ataskaitas.



3 pav. Panaudojimo atvejų modelis

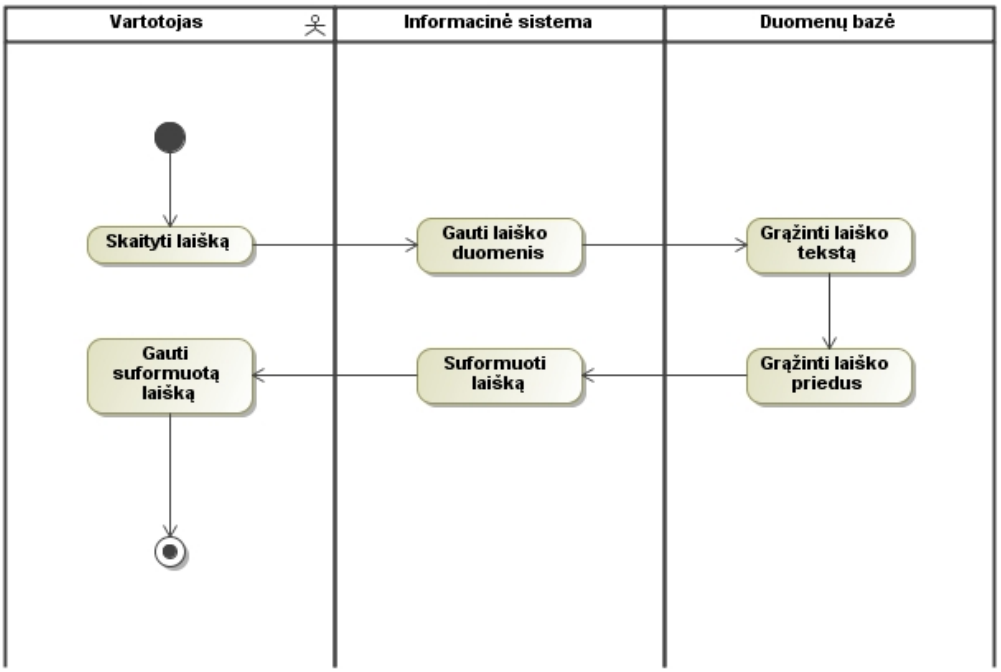
3.1.2 Veiklos procesų modeliai

Veiklos proceso „Laiškų peržiūra“ modelis pavaizduotas 4-ame paveiksle. Vartotojas prisijungęs prie sistemos nori peržiūrėti gautus elektroninius laiškus. Informacinė sistema kreipiasi į duomenų bazę, pateikdama laiškų gavimo užklausą. Duomenų bazė grąžina informaciją apie gautus laiškus, informacinė sistema savo ruožtu atitinkamai pažymi vartotojo dar neperskaitytus laiškus ir parodo pilną sąrašą vartotojui.



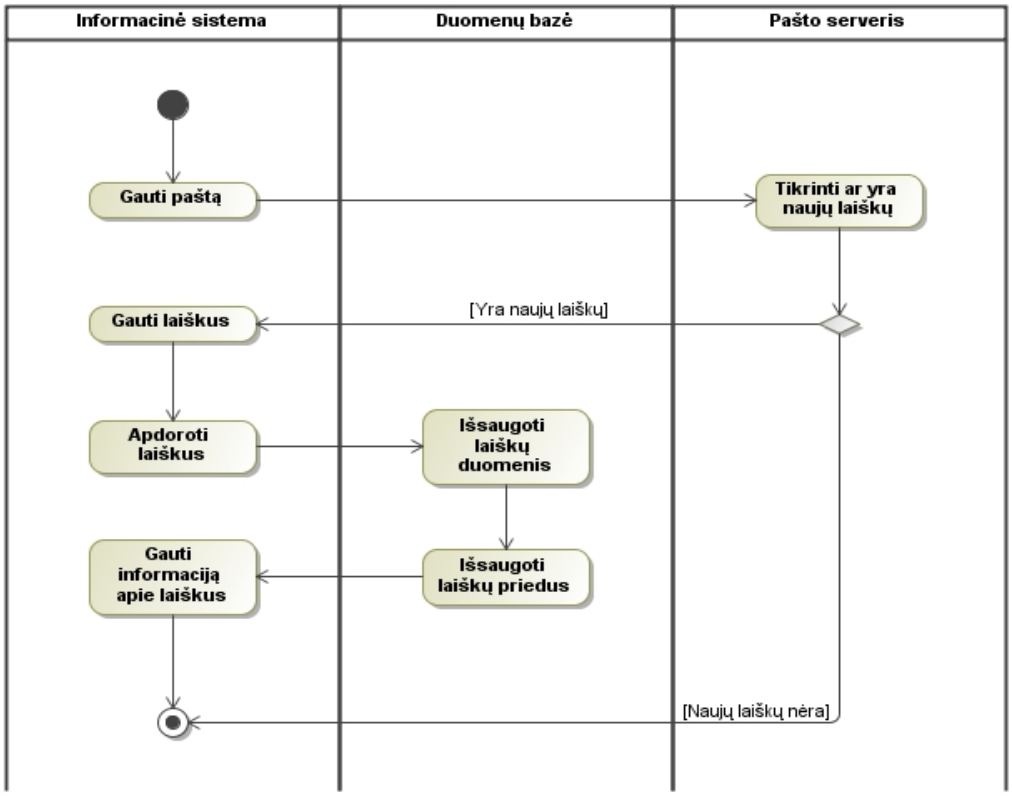
4 pav. Veiklos procesas „Laiškų peržiūra“

Veiklos proceso „Laiško skaitymas“ modelis parodytas 5-tame paveiksle. Vartotojas, pasirinkęs peržiūrai norimą laišką, mato į informacinėje sistemoje, kuri laiško tekstą ir jam priklausančius priedus gauna iš duomenų bazės.



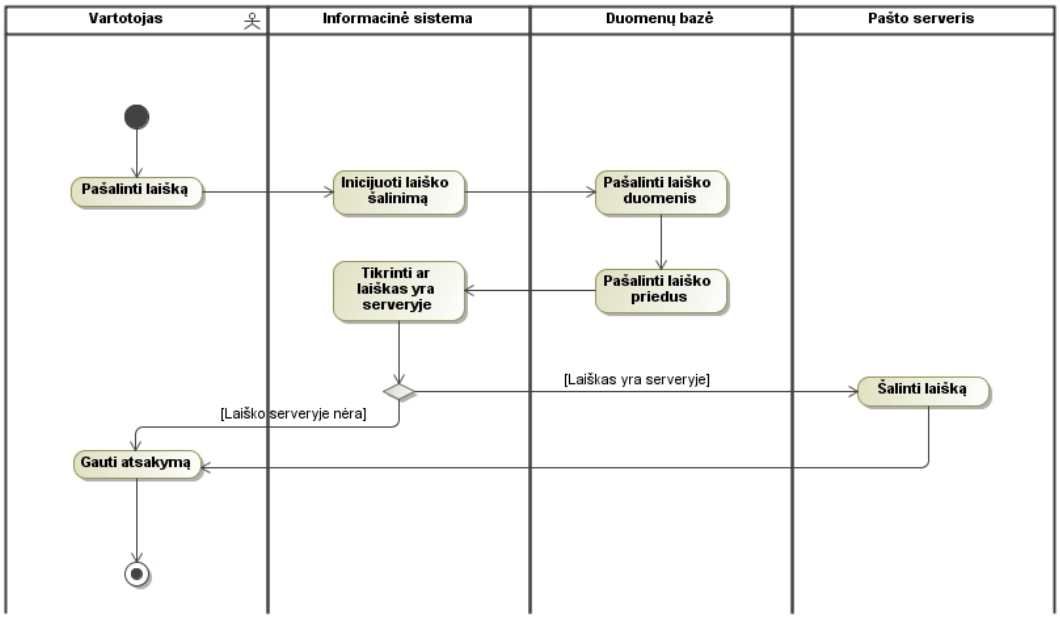
5 pav. Veiklos procesas „Laiško skaitymas“

Veiklos proceso „Pašto tikrinimas“ modelis pavaizduotas 6-tame paveiksle. Šis veiklos procesas vyksta be vartotojo įsikišimo. Sistema pati periodiškai susisiekia su pašto serveriu ir, jei jame yra naujų laiškų, išsaugo laiškus duomenų bazėje.



6 pav. Veiklos procesas „Pašto tikrinimas“

Veiklos proceso „Laiško šalinimas“ modelis pateikiamas 7-tame paveiksle. Vartotojui norint pašalinti pasirinktą laišką/us, sistema, priklausomai nuo nustatymų pašalina laiško informaciją ir jo priedus iš duomenų bazės ir/arba pašto serverio.

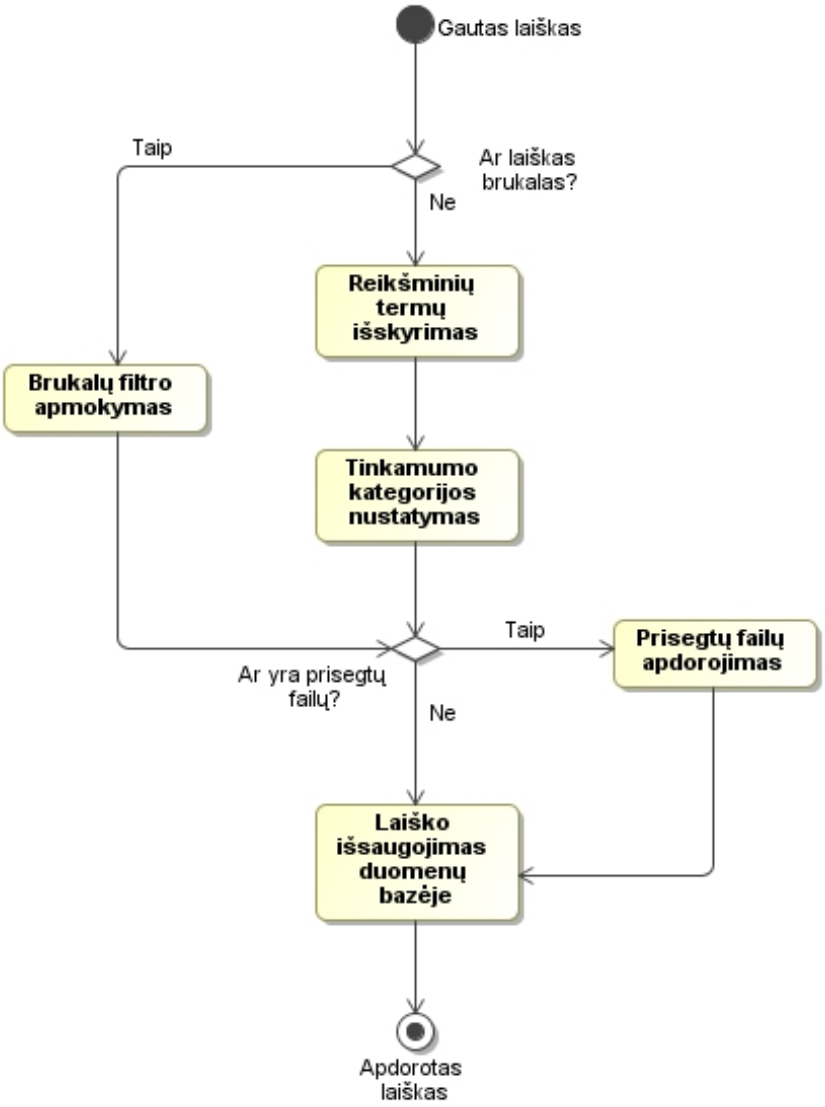


7 pav. Veiklos procesas „Laiško šalinimas“

3.1.3 Sistemos elgsenos modeliai

Šiame skyriuje pateikiami sistemos elgsenos modeliai, kurie aprašo pagrindinius sistemoje vykdomus procesus ir sistemos būsenas.

Pagrindinis sistemos uždavinys gavus naują elektroninį laišką, yra tinkamas jo apdorojimas. Pirmasis žingsnis apdorojant naują el. laišką yra laiško tikrinimas brukalų filtru. Jei laiškas įvertinamas kaip brukalas, vykdomas brukalų filtro apmokymas, naudojant apdorojamame laiške rastus žodžius. Tolesnis tokio laiško apdorojimas neatliekamas, laiškas priskiriamas brukalų kategorijai ir išsaugomas duomenų bazėje. Jei laiškas praeina brukalų filtrą kaip pageidaujamas, vykdomi tolesni apdorojimo veiksmai. Elektroninis laiškas nagrinėjamas gramatiškai, išškiriant jo reikšminius terminus (termus) ir pagal juos priskiriant laišką tinkamiausiai kategorijai. Jei gautas laiškas turi prisegtų failų, jie apdorojami atskirai ir išsaugomi duomenų bazėje, susiejant juos su laišku, kuriam priklauso. Apdorojus laišką ir su juo susijusius prisegtus failus, laiškas ir su juo susijusi informacija išsaugomi duomenų bazėje. Bendras sistemos procesų modelis parodytas 8-tame paveiksle.



8 pav. Bendras sistemos procesų modelis

Toliau pateikiami detalizuoti svarbiausių iš bendro sistemos procesų modelio išskirtų procesų modeliai. Elektroninio laiško apdorojimo procesas „Reikšminių termų iššskyrimas“ susideda iš keleto smulkesnių procesų. Pirmiausia iš viso el. laiško teksto atmetami iš anksto sudarytame žodyne esantys nereikšmingi žodžiai (įvardžiai, jungtukai, jaustukai, ištikttukai ir kiti). Tada laiško tekste likę žodžiai laikomi „žodžių krepšeliu“, o išvedus šių žodžių kamienines formas gaunami laiško termai. Suskaičiuojamas kiekvieno termo pasikartojimo dažnumas laiško tekste ir atrenkamas iš anksto nustatytas dažniausiai pasikartojančių termų skaičius N. Atrinkti termai laikomi reikšminiais apdorojamo laiško termais ir yra toliau naudojami procese. Reikšminių laiško termų išskyrimo proceso modelis parodytas 9-tame paveiksle. Toliau iš reikšminių el. laiško termų yra formuojama užklausa, pagal kurią iš duomenų bazės yra atrenkami saugomi el. laiškai, panašūs į nagrinėjamą. Užklausos rezultatai nufiltruojami, paliekant nustatytą panašiausių laiškų skaičių. Kiekvienam iš šių laiškų yra apskaičiuojami panašumo su nagrinėjamu

laišku koeficientai, pagal kuriuos sistema sprendžia, kuriai kategorijai priskirti nagrinėjamą el. laišką. Laiškas priskiriamas tai kategorijai, kuriai priklauso ir panašiausias laiškas iš atrinktų pagal užklausą. El. laiško priskyrimo kategorijai proceso modelis pavaizduotas 10-tame paveiksle.

9 pav. Reikšminių laiško termų išskyrimo proceso modelis

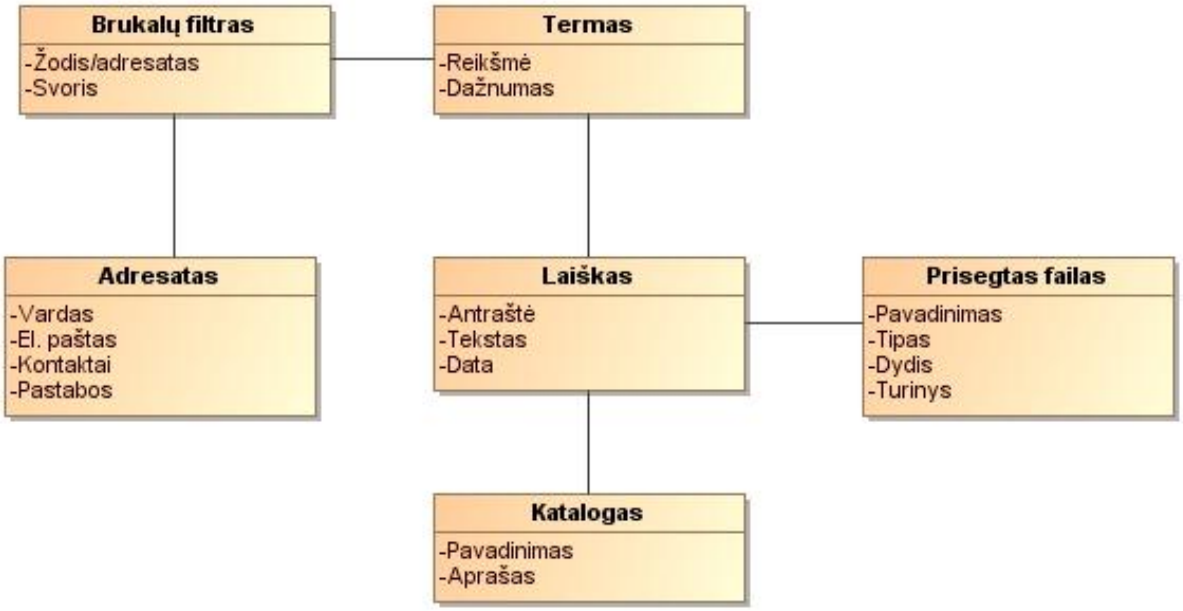
10 pav. El. laiško priskyrimo kategorijai proceso modelis

3.2 Duomenų modelis

Elektroninių laiškų saugojimo ir indeksavimo problemų sprendimui sistemoje numatoma visus duomenis saugoti nuo naudojamos el. pašto tvarkymo programos nepriklausomoje duomenų bazėje. Toks būdas išspręstų „failinės sistemos“ problemas, kai el. pašto programos reikalauja laiko ir resursų visų laiškų indekso sudarymui. Taip pat nepriklausoma duomenų bazė išsprendžia viso el. pašto dėžutės turinio perkėlimo į kitą

kompiuterį problemą, kai dalis laiškų ar prie jų prisegtų failų būna pametami. Prie laiškų prisegti failai taip pat saugomi duomenų bazėje, taip užtikrinant jų ryšį su laiškais net perkeliant visą duomenų bazę. Kiekvienas gautas naujas elektroninis laiškas išsaugomas duomenų bazėje ir yra bet kuriuo metu greitai pasiekiamas naudojant duomenų bazės indeksus.

Sistemoje išskiriamos atskiros duomenų esybės. Šiomis esybėmis grindžiamas duomenų bazės lentelių ir sąryšių tarp jų sudarymas. Esybės ir jų atributai parodyti sistemos esybių modelyje 11-tame paveiksle.



11 pav. Sistemos esybių modelis

3.3 Integracijos metodai

Elektroninio pašto automatizuotos procesų sistemos integracijai su Microsoft® Outlook® programine įranga pasirinkta kompanijos Microsoft® kartu su programa pateikiama išplėstinė pranešimų aplikacijos programavimo sąsaja (angl. Messaging Application Program Interface, MAPI). Net ir naudojant išplėstinę MAPI, programiniai kreipiniai gali būti naudojami ir netiesiogiai: per bazinę MAPI sąsają, naudojantis *dažnų pranešimų kreipinių*(angl. Common Messaging Calls, CMC) API klientine sąsaja, ar per objektinės CDO bibliotekos sąsają. Šiuos tris metodus naudoti paprasčiau ir jie yra skirti nesudėtingoms aplikacijoms, naudojančioms elektroninio pašto integraciją. (Bazinė MAPI ir CMC sąsajos buvo išimtos iš Exchange® nuo 2003 versijos).

3.4 Taisyklių sistema

Brukalų filtravimas. Pradininėje elektroninio laiško apdorojimo stadijoje, nustatymui ar laiškas yra brukalas naudojamas iš anksto apmokytas Bayes‘o filtras. Laikoma, kad nėra išankstinės tikimybės, jog laiškas yra brukalas. Tokiu atveju, tiek tikimybė kad laiškas yra brukalas, tiek tikimybė kad laiškas yra pageidaujamas yra lygios 50%, t.y. (1) formulės koeficientai Pr(S) ir Pr(H) yra lygūs 0.5. Tada (1) formulė supapaprastėja iki:

$$\Pr(S|W) = \frac{\Pr(W|S)}{\Pr(W|S) + \Pr(W|H)},$$

kur Pr(W|S) apskaičiuojama pagal nagrinėjamo žodžio pasitaikymo brukaluose dažnumą filtro apmokymo metu, o Pr(W|H) – pagal žodžio pasitaikymo dažnumą pageidaujamuose laiškuose.

Laiškų priskyrimas kategorijoms. Automatiniam laiškui katalogizavimui sistemoje pasirinktas panašumo kategorijų metodas. Šio metodo įgyvendinimui naudojami *tf-idf* metodu apskaičiuojami laiškų panašumo koeficientai. Kiekvienas naujas laiškas yra apdorojamas taikant šiuos metodus ir priskiriamas tinkamiausiai kategorijai.

Tinkamumo kategorijų metodo įgyvendinimas grindžiamas elektroninių laiškų indeksavimu susiejant laiškus su *tf-idf* reikšmėmis. Indeksai leidžia greitai pasiekti galimai tinkamus laiškus, o *tf-idf* metodas įgyvendina logiką, reikalingą laiškų tinkamumo kategorijai nustatymui.

Pirmiausia reikia laišką išnagrinėti gramatiškai, tada atmesti nereikšmingus žodžius ir išskirti likusių reikšmingų žodžių kamienus. Tada kiekvienas laiškas laikomas „žodžių krepšeliu“ ir suskaičiuojama kiek kartų kiekvienas „krepšelio“ žodis (ar jo kamienas) pasikartoja laiške. Galiausiai, atrenkama *N* dažniausiai pasitaikiusių žodžių (termų), ir jie toliau naudojami procese.

Tada randamas kiekvieno termo pasikartojimų nagrinėjamame dokumente (šiuo atveju elektroniniame laiške) kiekis (nustatytas reikšminių termų kiekis yra išskiriamas pradinio laiško apdorojimo metu). Šis skaičius yra normalizuojamas, kad būtų išvengta netikslumų, kurie gali atsirasti didesnės apimties dokumentuose, kai termas pasikartoja daug kartų, nors ir nėra svarbus tame dokumente. Normalizuotas kiekis laikomas termo dažnumu ir yra apskaičiuojamas pagal formulę:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}},$$

kur $n_{i,j}$ yra nagrinėjamo termo (t_i) pasikartojimų dokumente d_j kiekis, o vardiklyje yra visų dokumente d_j esančių termų pasikartojimų suma, t.y. dokumento žodžių kiekis.

Atvirkštinis dokumento dažnumas (angl. Inverse document frequency, idf) yra termo svarbumo dokumentų visumoje matas. Jis gaunamas padalinant visų dokumentų kiekį iš kiekio dokumentų, kuriuose sutinkamas nagrinėjamas ir apskaičiuojant gauto rezultato dešimtainį logaritmą:

$$idf_i = \log \frac{|D|}{1 + |\{j : t_i \in d_j\}|},$$

kur D yra visų dokumentų (el. laiškų) kiekis el. pašto dėžutėje, o $\{j : t_i \in d_j\}$ – dokumentų, kuriuose rastas termas t_i kiekis. Jei termo nėra nei viename dokumente, tokia išraiška reikštų dalybą iš nulio, todėl prie daliklio reikšmės pridedamas vienetas.

Tada nagrinėjamo termo *tf-idf* koeficientas gaunamas sudauginus *tf* ir *idf* reikšmes:

$$(tf - idf)_{i,j} = tf_{i,j} \times idf_i \qquad (2)$$

Didelis termo svoris (aukštas *tf-idf* koeficientas) pasiekiamas, kai termo dažnumas dokumente yra aukštas, o dokumentų, turinčių nagrinėjamą termą, dažnumas visoje dokumentų kolekcijoje – žemas. Tokiu būdu nufiltruojami žodžiai, dažnai sutinkami daugumoje dokumentų. Termo *tf-idf* koeficientas bus didesnis už nulį tada ir tik tada, kai pologaritminės funkcijos reikšmė reikšmė bus didesnė už vienetą. Kai (2) formulės vardiklyje pridedamas vienetas, termas, sutinkamas visuose dokumentuose turės neigiamą *idf* reikšmę, o termas sutinkamas visuose dokumentuose, išskyrus vieną, *idf* reikšmę lygią nuliui.

Kitas žingsnis yra laiškų suindeksavimas. Kiekvieno termo dažnumas laiškuose yra saugomas bendroje termų dažnumo lentelėje. Kiekvienas įrašas turi ryšį su laiškais, kuriuose buvo rastas termas. Tokiu būdu greitai pasiekiami visi laiškai, kuriuose sutinkamas reikalingas žodis. Šalia termų susiejimo su laiškais, taip pat yra ir pranešimų identifikatorių lentelė, kurioje saugomi kiekvieno laiško termai ir jų dažnumas laiške. Šios reikšmės gali būti apjungiamos su globaliomis reikšmėmis ir taip gaunami *tf-idf* koeficientai. Tokios duomenų struktūros sudarymui pakanka O(*n*) laiko, kur *n* yra iš laiško atrenkamų reikšminių termų skaičius.

Laiškai atrenkami iš duomenų struktūros naudojant užklausą. Užklausa susideda iš termų rinkinio. Užklausa suformuojama iš nustatyto laiško termų skaičiaus *N*. Iš duomenų struktūros atrenkami dokumentai, kuriuose sutinkamas bent vienas termas naudotas užklausoje. Užklausos rezultatus galima filtruoti, atrenkant tik labiausiai panašius laiškus ir tokiu būdu išvengti visų atrinktų laiškų apdorojimo. Tada, kiekvienas atrinktas laiškas lyginamas su užklausos termais, naudojant panašumo logiką. Laiško panašumui su užklausa nustatymui, tyrime pasirinktas Dice koeficientas:

$$Pnš(Užk, Dok) = \frac{2\sum_{i=1}^{\#Termai} TfIdf(Užk(i)) \times TfIdf(Dok(i))}{\sum_{i=1}^{\#Termai} TfIdf(Užk(i))^2 + \sum_{i=1}^{\#Termai} TfIdf(Dok(i))^2}$$

Šis koeficientas grąžina normalizuotą reikšmę nuo nulio iki vieneto, kur vienetas reiškia visišką atitikimą, o nulis – jokio atitikimo.

Atrinkus labiausiai į užklausą panašų laišką/us, nagrinėjamas laiškas (kurio termais pagrįsta užklausa) priskiriamas kategorijai, kuriai priklauso atrinktas panašiausias laiškas.

4 EKSPERIMENTINĖ DALIS

Eksperimentinėje dalyje realizuojama ankstesniame skyriuje aprašyta sistema. Suprogramuojami sistemos moduliai bei sukuriama suprojektuota duomenų bazė. Tiriama ir tobulinama sukurtos sistemos integracija su esamomis elektroninio pašto programomis. Taip pat nustatomi sistemos įvertinimo kriterijai ir matuojami sistemos veikimo parametrai.

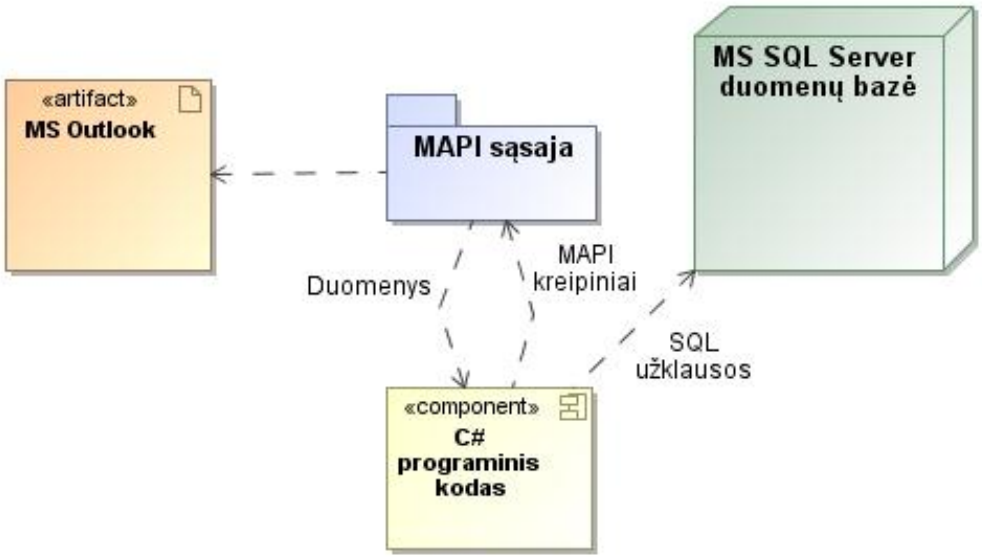
4.1 Praktinė sistemos realizacija ir integracijos tyrimai

Eksperimento metu buvo programiškai realizuota aprašytos sistemos dalis, reikalinga integracijos ir efektyvumo tyrimams atlikti. Kadangi vienas iš darbo tikslų buvo integruoti elektroninio pašto procesų automatizavimo sistemą su Microsoft® Outlook® programine įranga, eksperimento atlikimui buvo pasirinktos būtent kompanijos Microsoft® pateikiamos technologijos: programavimo aplinka MS Visual Studio® ir duomenų bazių valdymo sistema MS SQL Server®. Sistemos programinei realizacijai naudotos programinės įrangos versijos pateiktos 1-oje lentelėje:

Lentelė 1. Naudota programinė įranga

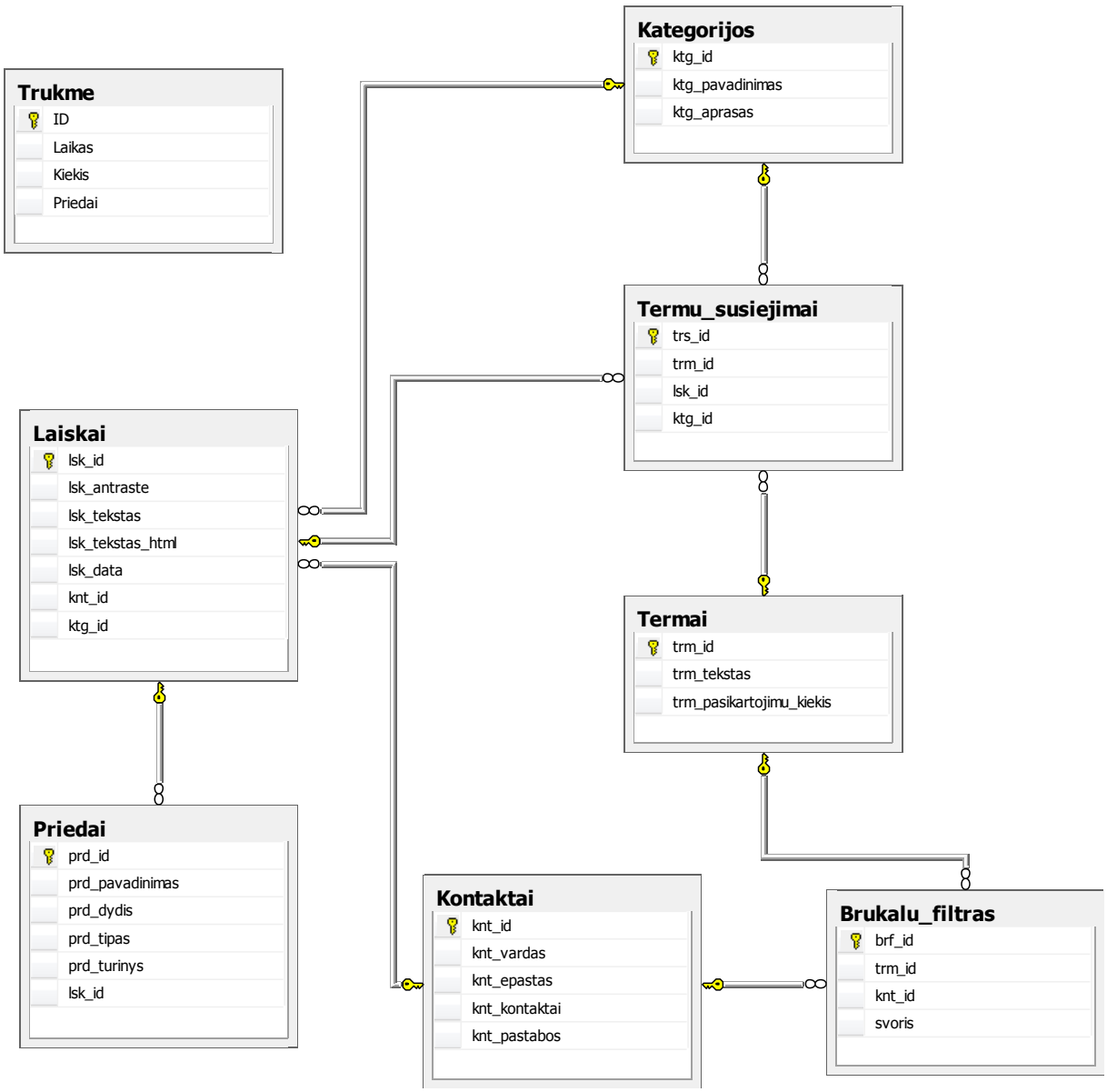
	Pavadinimas	**Versija**
Programavimo aplinka	Microsoft Visual Studio®	9.0.30729.1
Duomenų bazių valdymo sistema	Microsoft SQL Server®	10.0.2531.0
Elektroninio pašto programa	Microsoft Outlook®	14.0.4734.1000

Sistemos programinis kodas buvo parašytas programavimo kalba C#. Sistemos integracijai su Microsoft Outlook® programa panaudota aprašyta MAPI sąsaja. Naudojant programinius kreipinius į MAPI biblioteką, iš Microsoft Outlook® pašto dėžutės gaunami elektroniniai laiškai ir jų priedai, kurie apdorojami programiniame kode. Kiekvienas elektroninis laiškas yra išskaidomas į dalis (antraštę, siuntėją, tekstą, priedus ir kita) ir panaudojant SQL užklausas įterpiamas į duomenų bazę. Principinė sistemos realizacijos architektūra pateikta 12-tame paveiksle.



12 pav. Principinė eksperimentinės sistemos architektūra

Elektroninių laiškų duomenų, jų priedų ir eksperimento rezultatų saugojimui buvo sukurta aprašyta reliacinė duomenų bazė. Duomenų bazėje realizuotos sistemos modeliavimo dalyje aprašytos duomenų esybės ir sąryšiai tarp jų. Taip pat sukurta lentelė eksperimento duomenis kaupti ir saugoti. Eksperimento metu realizuotos duomenų bazės schema pateikiama 13-ame paveiksle.



13 pav. Eksperimentinės duomenų bazės schema

Pagrindinis sistemos uždavinys yra elektroninių laiškų ir jų priedų saugojimas duomenų bazėje. Eksperimento metu paaiškėjo, kad net ir naudojant MAPI sąsają, Outlook® programa dėl saugumo sumetimų nesuteikia tiesioginės prieigos prie savo duomenų saugykloje saugomų elektroninių laiškų priedų. Vienintelis būdas gauti priedus yra išsaugoti juos kompiuterio kietajame diske naudojant MAPI kreipinius. Dėl šios priežasties priedų išsaugojimas eksperimentinės sistemos duomenų bazėje reikalauja papildomo laiko ir resursų: pirmiausia reikia priedus išsaugoti kietojo disko kataloge ir tik tada galima juos nuskaityti į bitų masyvą iš kietojo disko ir įkelti į duomenų bazę. Toks perteklinis procesas mažina sistemos efektyvumą greičio atžvilgiu.

4.2 Sistemos kriterijų įvertinimas ir matavimas

Esminis tiriamos eksperimentinės elektroninio pašto procesų automatizavimo sistemos kriterijus yra jos našumas. Šio kriterijaus įvertinimui buvo atliktas eksperimentas importuojant elektroninius laiškus ir jų priedus iš Microsoft® Outlook® duomenų saugyklos į eksperimentinę duomenų bazę. Bandymai buvo atliekami naudojant asmeninį kompiuterį su 1.8 GHz taktinio dažnio AMD Mobile Sempron™ procesoriumi ir 2GB operatyvinės atminties. Kompiuteryje veikia operacinė sistema Microsoft® Windows 7 Professional®. Siekiant, kad eksperimento rezultatus kuo mažiau įtakotų operacinės sistemos ir kiti kompiuteryje veikiantys procesai, bandymai buvo atliekami kartojant juos po šimtą kartų.

Sistemos našumas buvo tiriamas matuojant laiką, kuris reikalingas nuskaityti skirtingą kiekį elektroninių laiškų iš Outlook® duomenų saugyklos ir išsaugoti juos eksperimentinėje duomenų bazėje. Programiniame kode, prieš metodo, kuris nuskaito laiškus iš saugyklos ir išsaugo duomenų bazėje, kreipinį buvo paleidžiamas laiko skaitiklis, o po metodo įvykdymo – sustabdomas. Šis procesas buvo kartojamas po šimtą kartų išsaugant 100, 200, 400 ir 800 laiškų duomenų bazėje, bei saugant laiškų priedus duomenų bazėje, kompiuterio kietojo disko kataloge ir visai jų neišsaugant. Kiekvieno matavimo rezultatai buvo atskira užklausa išsaugomi tam skirtoje duomenų bazės lentelėje. Matavimų rezultatai laiko atžvilgiu su skirtingais duomenų kiekiais ir skirtingais priedų išsaugojimo būdais pateikti 2-oje lentelėje:

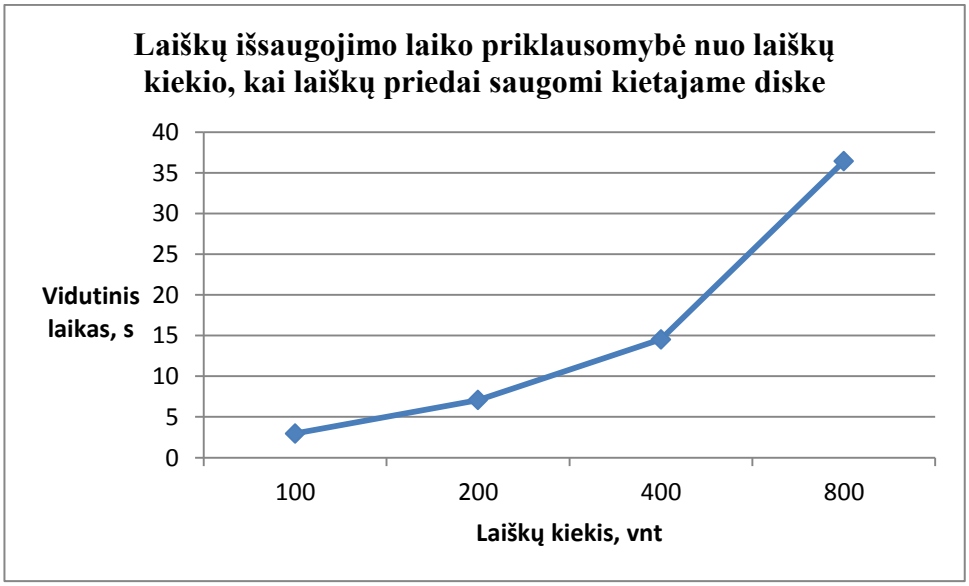
Lentelė 2. Laiškų įkėlimo į duomenų bazę trukmė

Priedai saugomi\Laiškų kiekis, vnt		**100**	**200**	**400**	**800**
Duomenų bazėje	**Min. laikas, s (iš 100 bandymų)**	3,6533203	8,7587891	18,9306641	51,0400391
	Maks. laikas, s (iš 100 bandymų)	4,8076172	13,9316406	22,34375	66,4619141
	Vid. Laikas, s (iš 100 bandymų)	3,947597659	9,573632817	20,031123045	55,325136724
Kietajame diske	**Min. laikas, s (iš 100 bandymų)**	2,7382813	6,3466796	13,5166015	34,1132812
	Maks. laikas, s (iš 100 bandymų)	3,5195313	7,8583984	20,0146484	43,3105469
	Vid. Laikas, s (iš 100 bandymų)	2,959863284	7,074228506	14,525097651	36,413535157

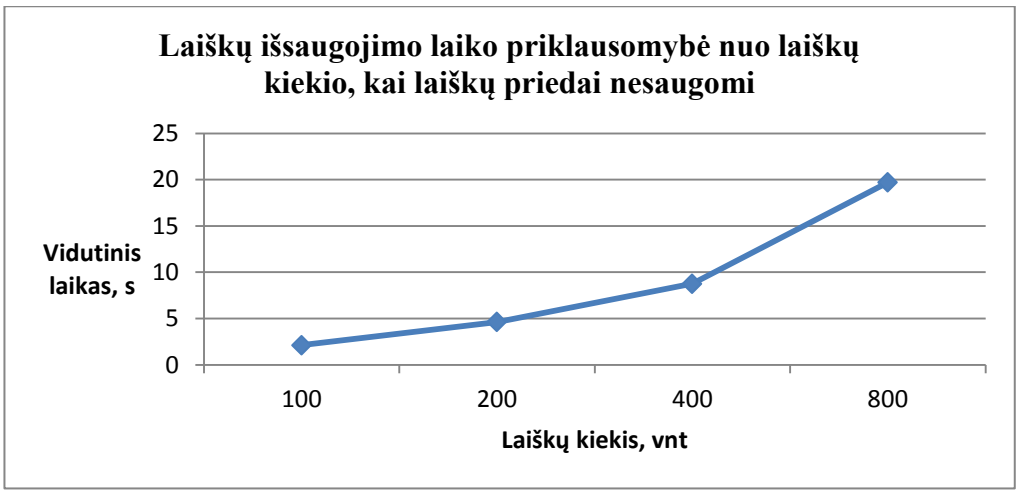
Nesaugomi	**Min. laikas, s (iš 100 bandymų)**	2,0341797	4,4326172	8,3789063	19,4091797
	Maks. laikas, s (iš 100 bandymų)	2,6298828	6,9394531	9,9667968	22,8515625
	Vid. Laikas, s (iš 100 bandymų)	2,133320318	4,636376961	8,758027342	19,704052735

Elektroninių laiškų išsaugojimo duomenų bazėje laiko priklausomybės nuo laiškų kiekio grafikai, kai laiškų priedai saugomi duomenų bazėje, kietojo disko kataloge ar nesaugomi visai, pateikiami atitinkamai 14-ame, 15-ame ir 16-ame paveiksluose.

14 pav. Laiškų išsaugojimo laiko priklausomybė nuo laiškų kiekio, kai laiškų priedai saugomi duomenų bazėje

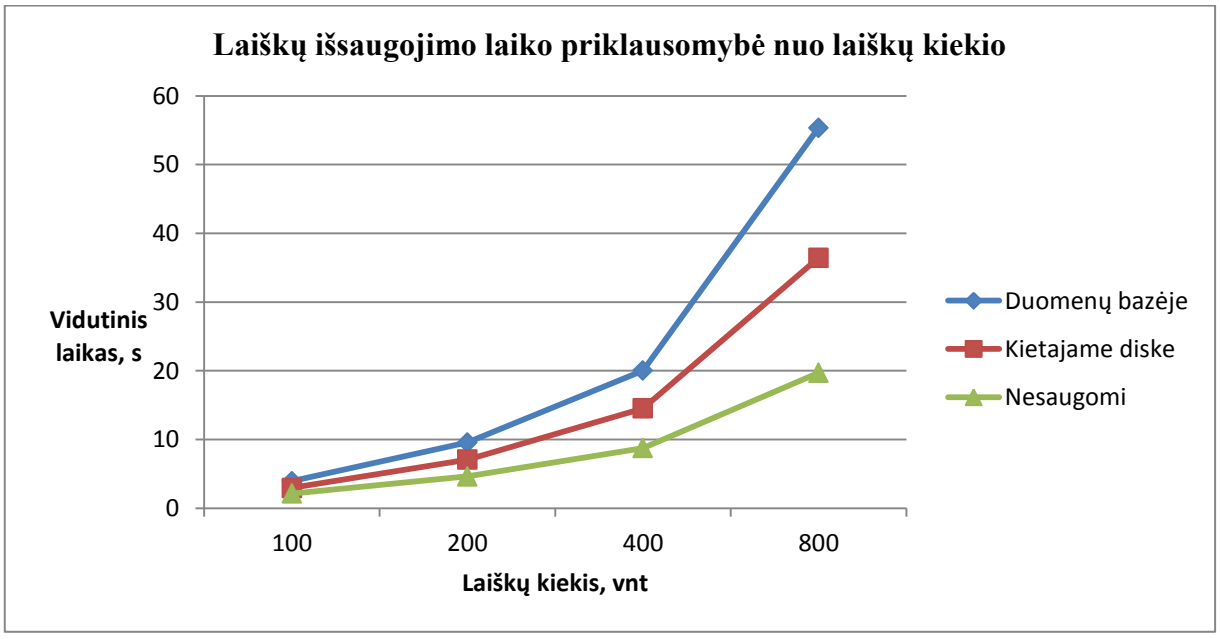


15 pav. Laiškų išsaugojimo laiko priklausomybė nuo laiškų kiekio, kai laiškų priedai saugomi kietajame diske



16 pav. Laiškų išsaugojimo laiko priklausomybė nuo laiškų kiekio, kai laiškų priedai neišsaugomi

Apjungtas elektroninių laiškų išsaugojimo duomenų bazėje laiko priklausomybės nuo laiškų kiekio grafikas pateikiamas 17-tame paveiksle.



17 pav. Laiškų išsaugojimo laiko priklausomybė nuo laiškų kiekio

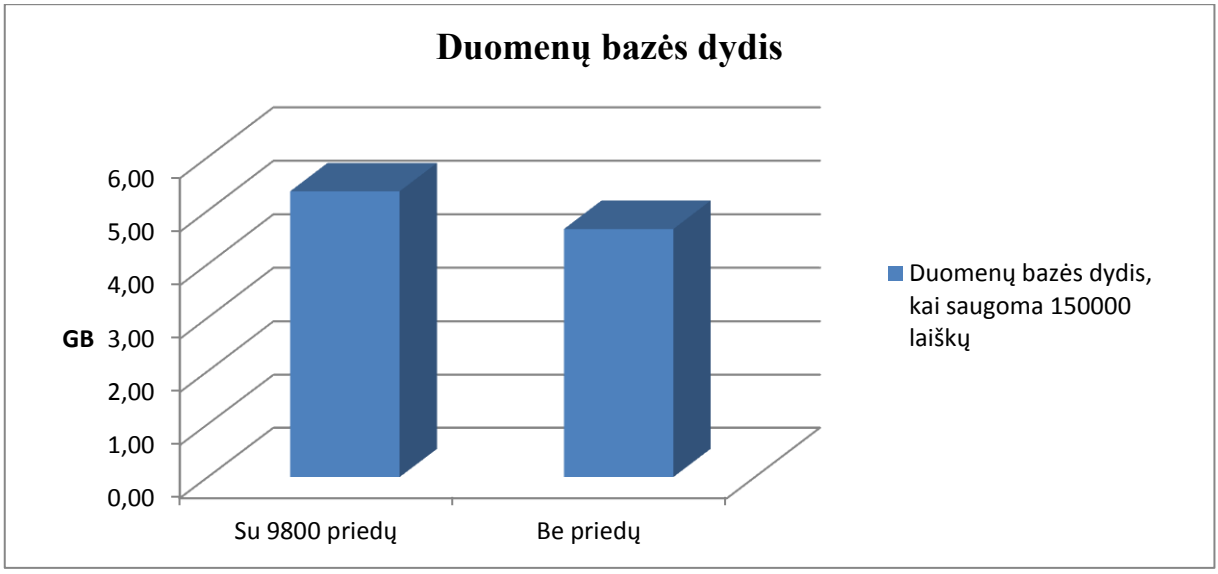
Iš eksperimento metu gautų rezultatų matyti, kad laiškų priedų saugojimo būdas gana ženkliai įtakoja laiškų išsaugojimo laiką duomenų bazėje, kuris didėjant laiškų kiekiui auga eksponentiškai. Priedų saugojimo būdo įtaka ypač išryškėja esant didesnėms duomenų apimtims: esant 800 laiškų duomenų imčiai, dėl laiškų saugojimo būdo, algoritmo greitis skiriasi net 200 – 300 procentų.

Kitas svarbus elektroninio pašto automatizuotos procesų sistemos veikimo kriterijus yra duomenų bazės dydis. Eksperimentinės duomenų bazės užimama vieta kompiuterio kietajame diske pateikiama 3-ioje lentelėje:

Lentelė 3. Duomenų bazės apimtis

Laiškų kiekis, vnt	Su 9800 įvairių priedų	Be priedų
150000	5.359,616 KB	4.654,080 KB

Prie elektroninių laiškų prisegtų failų išsaugojimo duomenų bazėje įtakos duomenų bazės apimčiai grafikas pateiktas 18-tame paveiksle.



18 pav. Duomenų bazės apimtis

Išmatavus duomenų bazės apimtį saugant laiškus su priedais ir be jų matoma, kad prisegtų failų saugojimas duomenų bazėje sąlyginai nedaug įtakoja duomenų bazės užimamą vietą kompiuterio kietajame diske.

4.3 SQL užklausų efektyvumo tyrimas

SQL (angl. Structured Query Language) tyrimas buvo atliekamas naudojantis kartu su MS SQL Server® pateikiamu įrankiu SQL Server Profiler®. Buvo matuojamas elektroninių laiškų bei jų priedų įkėlimo į duomenų bazę užklausų laikas, kompiuterio

procesoriaus atliekamos duomenų nuskaitymo ir įrašymo operacijos, reikalingos užklausos įvykdymui ir kita. Apibendrinti matavimų rezultatai pateikiami 4-oje lentelėje:

Lentelė 4. Laiškų įkėlimo į duomenų bazę užklausų parametrai

Priedai saugomi\Laiškų kiekis, vnt		**Trukmė, ms**	**Nuskaitymai**	**Įrašymai**
El. laiško įkėlimo užklausa	**Pirma užklausa**	4	126	3
	Sekančios užklausos (vid.)	1,7	2	0
Failo (220 kB) įkėlimo užklausa	**Pirma užklausa**	256	162	35
	Sekančios užklausos (vid.)	3,5	158	29

Atlikus bandymus matoma, kad pirma vykdoma užklausa yra vykdoma gerokai ilgiau ir reikalauja daugiau resursų nei po jos sekančios analogiškos užklausos. Taip yra todėl, kad duomenų bazės valdymo sistema gavusi pirmąją užklausą turi atlikti jos sintaksinę analizę ir „išrišimą“, o vėlesnėms analogiškoms užklausoms šie veiksmai nebeatliekami, nes duomenų bazės variklis remiasi prieš tai išnagrinėta analogiška užklausa. Tokiu būdu velėsnės užklausos įvykdomos gerokai greičiau nei pirmoji ir tai leidžia teigti, kad SQL užklausų naudojimas elektroninių laiškų įterpimui į duomenų yra efektyviausias tada, kai paeiliui vykdomos analogiškos užklausos, t.y. pirmiausia sukeliami visi elektroniniai laiškai, o tada jų priedai. Tačiau realizuotoje eksperimentinėje sistemoje toks metodas nebūtų efektyvus todėl, kad norint tokiu būdu vykdyti SQL užklausas ir išsaugoti elektroninių laiškų susiejimą su jų priedais, reikėtų du kartus atlikti ciklišką MS Outlook® duomenų saugyklos nuskaitymą, bei naudoti papildomas SQL užklausas, kas užimtų daugiau laiko, nei nenuoseklus SQL užklausų vykdymas.

5 IŠVADOS

- Atlikus panašaus pobūdžio elektroninio pašto procesų automatizavimo sistemų analizę nustatyta, kad sistemos “Xobni” ir “Lookeen” neturi savo duomenų saugyklų korespodencijai, o naudoja “MS Outlook®” programos iššsaugotus laiškus. Šios sistemos laiškų indeksavimui naudoja failinę sistemą, kuri augant elektroninio pašto apimčiai tampa neefektyvi funkcionavimo atžvilgiu, t.y. didėjant pašto apimčiai sulėtėja atliekamų operacijų greitis, kuris ženkliai (iki keleto minučių) jaučiamas vartotojui.
- Elektroninių laiškų rūšiavimui ir filtravimui naudojamų metodų ir algoritmų analizė parodė, kad brukalų atskyrimui plačiausiai naudojamas ir bene tiksliausiai „šlamštą“ nufiltruojantis yra Bajeso algoritmas, o laiškų rūšiavimui į atitinkamus katalogus TF-IDF metodas. Šie metodai yra savaime tobulėjantys (angl. self-learning), todėl reikalauja tik minimalaus vartotojo įsikišimo.
- Remiantis analitinėje dalyje apžvelgtais elektronių laiškų apdorojimo metodais sudarytas sistemos modelis, detaliai aprašyti sistemoje vykstantys procesai ir jų veikimo principai. Sistemos duomenų modelis pagrįstas elektroninių laiškų ir jų priedų saugojimu duomenų bazėje, nenaudojant „failinės sistemos“ ir su ja susijusių pašto dėžutės perkėlimo ir lėto laiškų indeksavimo funkcijų.
- Sistemos integracijai su Microsoft® Outlook® programine įranga panaudota kompanijos Microsoft® pateikiama MAPI (angl. Messaging Application Program Inteface) sąsaja, kuri sudarė aprašytai sistemai galimybę naudotis Outlook® programos duomenimis ir funkcijomis.
- Sukurta taisyklių sistema, kuri naudodama aprašytus TF-IDF bei tinkamumo kategorijų metodus leidžia automatiškai klasifikuoti elektroninę korespondenciją pagal vartotojo sukurtas kategorijas ir nereikalauja papildomų vartotojo veiksmų, norint priskirti laišką vienai ar kitai kategorijai.
- Realizuota eksperimentinė elektroninio pašto automatizuotos procesų sistemos dalis. Atlikus sistemos integraciją su Microsoft® Outlook® programine įranga paaiškėjo, kad elektroninių laiškų priedų eksportavimas iš Outlook® saugyklos ir įkėlimas į duomenų bazę nėra efektyvus, nes reikalauja perteklinių procesų, kurie viso proceso trukmę prailgina ~1,4 karto lyginant su laiškų priedų saugojimo kompiuterio kietajame diske.

- Eksperimentiškai įvertinus elektroninio pašto automatizuotos procesų sistemos greitį paaiškėjo, kad laiškų saugojimas duomenų bazėje yra efektyvus greičio ir sistemos mobilumo požiūriu, tačiau laiškų priedų saugojimo sprendimą dar reikėtų tobulinti, nes priklausomai nuo laiškų saugojimo būdo, elektroninių laiškų išsaugojimo greitis skiriasi 1,3 – 1,5 karto: įkeldama 800 laiškų į duomenų bazę, o jų priedus saugodama kietajame kompiuterio diske sistema vidutiniškai užtrunka ~36,4 sekundės, kai tuo tarpu tokio pat kiekio laiškų ir jų priedų saugojimui duomenų bazėje sistemai vidutiniškai prireikia 55,3 sekundės.

6 LITERATŪRA

[1] Laclavik, M.; Maynard, D.; Motivating Intelligent E-mail in Business: An Investigation into Current Trends for E-mail Processing and Communication Research; Commerce and Enterprise Computing, 2009. Psl: 476 – 482

[2] Fast and easy email management: Xobni Review [interaktyvus] Žiūrėta: 2010-06-16 Priega per internetą: <http://blog.taragana.com/index.php/archive/fast-and-easy-email-management-xonbi-review/>

[3] Indexing and Searching emails faster in Outlook-Xobni review [interaktyvus] Žiūrėta: 2010-06-16 Priega per internetą:<http://helpdeskgeek.com/free-tools-review/indexing-and-searching-emails-faster-in-outlook-xobni-review/>

[4] Lookeen 2010 3.0 - Outlook Search Add-On [interaktyvus] Žiūrėta: 2010-06-16 Priega per internetą: <http://email.about.com/od/outlookaddons/gr/lookeen.htm

[5] Lookeen review – better search for Outlook [interaktyvus] Žiūrėta: 2010-06-16 Priega per internetą: <http://blog.jtbworld.com/2009/06/lookeen-review-better-search-for.html>

[6] Xiao-Lin Wang; Cloete, I.; Learning to classify email: a survey; Machine Learning and Cybernetics, Proceedings of 2005 International Conference ;2005 , Psl: 5716 - 5719

[7] M. Porter, "An algorithm for suffix stripping," Program, vol. 14, no. 3, Psl: 130–137.

[8] J. Provost, "Naieve-bayes vs. rule-learning in classification of email." [interaktyvus]. Žiūrėta: 2010-06-16 Prieiga per internetą:

<http://citeseer.ist.psu.edu/provost99naiumlvebayes.html>

[9] Bajeso teorema [interaktyvus]. Žiūrėta: 2010-06-16 Prieiga per internetą: <http://lt.wikipedia.org/wiki/Bajeso_teorema>

[10] J. D. Brutlag and C.Meek, "Challenges of the email domain for text classification," ICML, 2000, Psl: 103–110.

[11] Ayodele, T.; Shikun Zhou;Applying Machine learning Algorithms for EmailManagement; Pervasive Computing and Applications, ICPCA 2008, Psl:339 – 344

[12] Ayodele, T.; Shikun Zhou; Khusainov,R.; Evolving email clustering method for email grouping:A machine learning approach; Applications of Digital Information and Web Technologies, 2009. Psl: 357 – 362

[13] B. Klimt, Y. Yang; The enron corpus: A new dataset for email classification research; *Proceedings of ECML'04, 15th European Conference on Machine Learning*, 2004. Psl: 217-226

[14] R. Bekkerm, A. McCallum, G. Huang; Automatic Categorization of Email into Folders:

Benchmark Experiments on Enron and SRI Corpora, 2008.

[15] Huijie Yang, Junyong Luo, Meijuan Yin, Yan Liu; Automatically Detecting Personal Topics by Clustering Emails; Education Technology and Computer Science (ETCS), 2010 Second International Workshop, 2010. Psl: 91-94

[16] Qing Yang, Fang-Min Li; Support Vector Machine For Customized Email Filtering Based On Improving Latent Semantic Indexing; Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 2005.

[17] Tao Liu, Zheng Chen, Benyu Zhang, Weiying Ma and Gongyi Wu; "Improving Text Classification using Local Latent Semantic Indexing"; Proceeding of ICDM2004, pp. 162-169, 2004.

[18] Kenrick Mock; " Dynamic Email Organization via Relevance Categories"; Tools with Artificial Intelligence, 1999. Proceedings. 11th IEEE International Conference on , 1999.

[19] MAPI (Messaging Application Program Interface) [interaktyvus]. Žiūrėta: 2011-05-05 Prieiga per internetą: < http://searchexchange.techtarget.com/definition/MAPI>.

[20] Messaging Application Programming Interface [interaktyvus]. Žiūrėta: 2011-05-05 Prieiga per internetą: <http://en.wikipedia.org/wiki/Messaging_Application_Programming_Interface>.

Development and research of automated e-mail process system

Summary

Email in nowadays is an important part of business and personal life. Many people use email everyday and it is a significant tool. There are lots of programs created to be used for reading, sending and managing email, but not all of these are able to deal with huge amounts of email. The main objectives of this work were to analyze methods and algorithms of quick and effective email management and to design a system, which could implement the best methods and achieve good results in automated email processing. After doing analysis of commonly used approaches to solve described problems, several algorithms of email processing were chosen. The idea to achieve fast automated email processing was to store all received emails and attachments into the database, which ensures data integrity and fast indexing. To find out how such a system would perform in real circumstances an experiment was made. It involved creating a database and program code, which processes the email data and uploads it to the database. After test running the system, the test data and measurement results were stored and evaluated. Results exposed, that chosen approach for email processing was effective, but storing email attachments in the database caused some problems and probably is not the best way to deal with attached files.