VYTAUTO DIDŽIOJO UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS INSTITUTAS

Tatjana Jevsikova

INTERNETINĖS PROGRAMINĖS ĮRANGOS LOKALIZAVIMAS

Daktaro disertacija

Fiziniai mokslai (P 000)
Informatika (09 P)
Informatika, sistemų teorija (P 175)

Vilnius, 2009

Disertacija rengta 2005–2009 metais Matematikos ir informatikos institute

Mokslinė vadovė:

Prof. dr. Valentina DAGIENĖ (Matematikos ir informatikos institutas, fiziniai mokslai, informatika 09 P)

PADĖKA

Nuoširdžiai dėkoju mokslinei vadovei prof. dr. Valentinai Dagienei už nuoseklų vadovavimą, vertingas mokslines konsultacijas, pasitikėjimą manimi, nuolatinį padrąsinimą, kantrybę, neįkainuojamus patarimus ir pagalbą.

Esu labai dėkinga darbo recenzentams doc. dr. Gintautui Grigui ir doc. dr. Vladui Tumasoniui už vertingas pastabas ir diskusijas, padėjusias tobulinti šį darbą.

Dėkoju Matematikos ir informatikos instituto direktoriui prof. habil. dr. Gintautui Dzemydai ir Informatikos metodologijos skyriaus darbuotojams už naudingus patarimus ir pagalbą.

Ačiū mano šeimai už moralinį palaikymą, šiltą aplinką ir supratimą.

Tatjana Jevsikova

TURINYS

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNĖLIS

atributinė gramatika

Atributinė gramatika AG = <G, A, R> sudaroma iš trijų komponentų [Knu68]:

1. Bekontekstė gramatika G.
2. Baigtinė atributų aibė *A*.
3. Baigtinė semantikos taisyklių aibė *R*.

Su kiekvienu gramatikos simboliu X susiejama baigtinė atributų aibė A(X). A(X) aibė dalijama į du poaibius: paveldėti atributai I(X) ir sintezuoti atributai S(X), taip kad $I(X) \cap S(X) = \varnothing$. Visų gramatikos atributų aibė $A = \cup A(X)$.

Semantikos taisyklių baigtinė aibė yra susieta su išvedimo taisykle P. Pateikiama po vieną taisyklę kiekvienam sintezuotam atributui $X_0.a$ ir po vieną taisyklę kiekvienam paveldėtam atributui $X_i.a$, $1 \le i \le n$.

bekontekstė gramatika

Formalioji gramatika, kurios išvedimo taisyklės išreiškiamos pavidalu $A \rightarrow w$, čia $A \in N$, $w \in (N \cup T)^*$, N – neterminalinių simbolių aibė, T – terminalinių simbolių aibė.

fleksinė kalba

Kalba, kuriai būdingos fleksijos (kaityba) ir afiksų daugiareikšmiškumas. Iš indoeuropiečių kalbų ypač gausingą daugiareikšmę fleksiją, arba galūnių sistemą, turi sanskritas, senoji graikų, lotynų, lietuvių, rusų kalbos (anglų, prancūzų kalbose fleksija jau yra gerokai apnykusi) [Pal85].

formalioji gramatika

Formaliąją gramatiką G = <N, T, P, S> sudaro keturios dalys:

1. Neterminalinių simbolių baigtinė aibė N.
2. Terminalinių simbolių baigtinė aibė (kalbos abėcėlė) T.
3. Gramatikos taisyklių baigtinė aibė P; bendru atveju taisyklė išreiškiama $u \rightarrow v$, čia $u \in (N \cup T)+$, $v \in (N \cup T)^*$;
4. Pradinis neterminalinis simbolis S ($S \in N$).

grafinė sąsaja,
grafinė naudotojo sąsaja

Grafikos priemonėmis pagrįsta ryšio ir sąveikos priemonių tarp žmogaus ir kompiuterinės programos visuma.
Komandoms parinkti, programoms paleisti, failų ir katalogų vardams, parinktims, parametrams stebėti ir parinkti, taip pat kitiems veiksmams atlikti naudojami ekrane rodomi dialogo langai, meniu punktai, mygtukai ir kt. elementai, kuriais manipuliuojama pele, klaviatūra ar kt. manipuliavimo įtaisu.

internacionalizacija

1. Programinės įrangos projektavimo proceso, pritaikant ją įvairioms kalboms ir kultūroms, rezultatas.
2. Programinės įrangos savybių, darančių ja lengvai adaptuojama įvairioms kalboms ir kultūroms, visuma.

internacionalizavimas	Programinės įrangos ir jos duomenų struktūrų projektavimas taip, kad visa tai būtų galima lengvai adaptuoti įvairioms kalboms ir kultūroms.
internacionalizavimo klaida	Programinės įrangos projektavimo klaida, kai tam tikra jos savybė arba parametras nenumatomas įvairioms lokalėms pritaikyti.
internetinė programinė įranga	Programinė įranga, skirta dirbti internete (interneto teikiamoms paslaugoms pasiekti), kurią galima perskirti į dvi pagrindines grupes: 1. Klientų kompiuteriuose veikianti programinė įranga, kuri paprastai priskiriama ir prie bendrosios paskirties programinės įrangos. 2. Internete veikiančios programos, realizuojančios interneto paslaugas ir veikiančios tinklo serverių kompiuteriuose. Naudotojo požiūriui pateikiamos kaip sudėtingesnės interneto svetainės, pasiekiamos naudojantis naršykle.
išteklių atskyrimo metodas	Veiksmų ir priemonių visuma programos ištekliams atskirti nuo vykdomosios dalies.
išteklių eilutė, eilutė	Programos tekstinių lokalizuojamųjų išteklių elementas – ženklų seka, skirta rodyti ekrane (pvz., meniu punkto, komandos, klaidos pranešimo tekstas) arba nurodyti tam tikrą lokalizuojamosios programos nuostatą (pvz., koduotę, šriftą).
išteklių sekcija	Programos modulio vykdomojo failo sekcija, į kurią sudėti programos ištekliai.
koduotė	Ženklų kodavimas, vienareikšmiškai apibrėžiantis tam tikro rinkinio ženklų kodus.
kontekstas	Žodį, frazę, įvykį, reiškinį arba kitokį objektą supanti aplinka, nuo kurios gali priklausyti to objekto prasmė ir interpretacija.
kultūros elementas	Kompiuteriui skirtas duomenų elementas, priklausomas nuo kalbos, valstybės, teritorijos ir kt. kultūros aplinkybių.
lokalė	Kompiuteryje ir jo programinėje įrangoje naudojamų elementų, priklausančių nuo kalbos ir kultūros normų, visuma.
lokalės elementas	Kultūros elementas, kuris yra įtrauktas į formalųjį kultūros elementų aprašą – lokalę.
lokalizacija	1. Programinės įrangos adaptavimo kultūrinei ir kalbinei aplinkai proceso rezultatas. 2. Programos lokalizuota atmaina.
lokalizacijos testavimas	Lokalizacijos patikrinimas, ar jos tekstai išversti teisingai ir taisyklingai, ar lokalizacija atitinka kultūrinius reikalavimus.
lokalizavimas	Programinės įrangos pritaikymas tam tikrai kalbinei ir kultūrinei aplinkai.

lokalizavimo klaida	Programinės įrangos kurios nors savybės neatitikimas lokalei, kai ta savybė yra internacionalizuota.
lokalizuojamieji ištekliai	Programos ištekliai, į kuriuos sudėti duomenys, priklausomi nuo kalbos ir kultūros.
lokalizuojamųjų išteklių metainformacija	Informacija, skirta apibūdinti lokalizuojamųjų išteklių duomenis: jų kontekstą, paskirtį, tipą ir pan.
lokalizuojamųjų išteklių pateikimo formatas	Lokalizuojamuose ištekliuose laikomų duomenų apipavidalinimo būdas.
MO	Angl. *Machine Object*. PO failo sukompiliuotas dvejetainis pavidalas, skirtas platinti kartu su programa.
parametrizuota eilutė	Išteklių eilutė, kurioje yra pavartotas parametras – reikšmė, įterpiama į eilutę programą vykdant.
PO	GNU „Gettext“ lokalizuojamųjų išteklių atskyrimo metodo naudojamas tekstinių lokalizuojamųjų išteklių apipavidalinimo būdas, kai lokalizavimui skirtame faile laikomos išteklių eilutės ir jų vertimai. Angl. *Portable Object*.
segmentas	Išteklių eilutė arba jos dalis.
standartas	Dokumentas, kuriuo nustatomi vienodi reikalavimai gaminiams, veiklai, dokumentų formoms ir kt., priimtas ir patvirtintas autoritetingos vyriausybinės arba nekomercinės organizacijos.
tekstiniai lokalizuojamieji ištekliai	Programos lokalizuojamieji ištekliai, pateikti tekstiniu pavidalu.
valdiklis	Bet koks valdymo elementas grafinės sąsajos ekrane, pvz., mygtukas, meniu juosta, išskleidžiamasis sąrašas. Angl. *control*.

1. ĮVADAS

1.1. Darbo aktualumas

Viena iš svarbių kultūriniu ir ekonominiu požiūriais programinės įrangos savybių yra jos sąsajos su naudotoju pateikimas naudotojo kalba. Jei programa lokalizuojama, tai ji turi atrodyti naudotojui kuo natūraliau, tarsi būtų sukurta jo kultūrinėje terpėje. Programinės įrangos lokalizavimo poreikis atsirado tada, kai prasidėjo masinis jos eksportas į įvairias valstybes. Šiandien, augant kompiuterių ir interneto naudotojų skaičiui, šis poreikis vis didėja. Todėl lokalizavimo tyrinėjimai ir jo spartinimo bei kokybės gerinimo metodų paieška – aktuali problema.

Programinės įrangos lokalizavimo pradžioje (XX a. 9-asis deš.) pagrindinis dėmesys buvo skiriamas galimybei apdoroti lokalės tekstus, tik vėliau buvo imta nagrinėti pritaikymą įvairioms kultūrinėms ir kalbinėms normos, galiojančioms konkrečioje kalboje ar teritorijoje, dar vėliau – visapusiškam programinės įrangos adaptavimui lokalei, tarsi programa būtų specialiai suprojektuota konkrečios vietovės bendruomenei [Gri98]. Tačiau net ir dabar tenka pastebėti, kad programinės įrangos lokalizavimas kartais suprantamas kaip vertimas. Nors iš tikrųjų net jei turėtume omenyje tik programinės įrangos tekstų pateikimą kita kalba, šis darbas būtų labiau panašus ne į įprastą, bet į gerokai adaptuotą vertimą.

Daugelis visuotinai naudojamų programų dar nėra pakankamai lokalizuotos, pasitaiko įvairių klaidų ir neatitikimų kalbos normoms. Ypač daug jų yra internetinėje programinėje įrangoje, kuri internetui paplitus yra visuotinai naudojama.

Darbe tiriamas programinės įrangos lokalizavimas. Pagrindinis dėmesys skiriamas internetinei programinei įrangai, tačiau daugelis gautų rezultatų tiks ir bendrosios paskirties programinei įrangai lokalizuoti.

Nagrinėjamą internetinę programinę įrangą sudaro dvi darbui internete skirtos programų grupės:

1. Klientų kompiuteriuose veikianti programinė įranga, kuri paprastai priskiriama ir prie bendrosios paskirties programinės įrangos (pvz., naršyklė, elektroninio pašto programa).
2. Internete veikiančios programos, realizuojančios interneto paslaugas ir veikiančios tinklo serverių kompiuteriuose. Naudotojo požiūriui jos pateikiamos kaip sudėtingesnės interneto svetainės, pasiekiamos naudojantis naršykle (pvz., virtualioji mokymosi aplinka, turinio valdymo sistema).

Lokalizuojant internetinę programinę įrangą ir siekiant užtikrinti lokalizavimo tęstinumą, susiduriama su šiomis pagrindinėmis problemomis, darančiomis neigiamą įtaką dabartinių lokalizacijų kokybei:

1. Dažnai išleidžiamos naujos programų versijos (kur kas dažniau lyginant su autonomiškai veikiančia programine įranga).
2. Versijų atnaujinimai vyksta internetu, dažnai automatiškai, o lokalizuotų versijų atnaujinimai platinami vienu metu su originalių versijų atnaujinimais.

Daugelis lokalizavimo problemas tiriančių mokslininkų pabrėžia, kad programinės įrangos lokalizavimo darbus galima suskirstyti į dvi pagrindines dalis [Ess00; Sul01; DGJ04; Yan07 ir kt.]:

- Programos adaptavimas konkrečiai kalbinei ir kultūrinei terpei (koduotės, skaičių formatai, datų ir laiko formatai, dokumentų formos ir kt.);

- Dialogo tekstų (meniu komandų, mygtukų užrašų ir kt., įskaitant elektroninius žinynus, naudotojo vadovus) vertimas ir adaptavimas.

Pirmosios dalies problemos dažniausiai sprendžiamos naudojantis lokalės formaliais aprašais.

Gerokai didesnės problemos kyla imantis spręsti antrosios dalies – dialogo tekstų – lokalizavimą. Tekstų daug, programos dažnai atnaujinamos, reikia nuolatos būti pasirengus išversti po keletą eilučių, kurios ištrauktos iš konteksto neretai esti sunkiai suprantamos. Dėl šios priežasties pirminis dialogo tekstų vertimas gali būti laikomas tik juodraštiniu ir turi būti tobulinamas kartojant testavimo ir vertimo derinimo ciklą, kurio metu ieškomas dialogo tekstų kontekstas veikiančioje programoje ir atsižvelgiant į jį koreguojamas vertimas. Dažnai lokalizavimo klaidos atrandamos jau išleidus platinti skirtą programos versiją.

Kita problema – tai nepakankamas programinės įrangos internacionalizavimo lygis. Lokalizavimo metu paprastai atrandama internacionalizavimo klaidų, kurias privalo ištaisyti programos autoriai. Tačiau dėl minėto versijų išleidimo dažnumo klaidų taisymas nukeliamas į vėlesnes versijas.

Todėl paieška būdų, kaip pagerinti lokalizuotų dialogo tekstų kokybę, sumažinti lokalizacijos testavimo sąnaudas yra aktuali problema.

1.2. Darbo objektas

Programinės įrangos lokalizavimo procesas, internetinių programų lokalizuojamieji ištekliai ir formaliųjų gramatikų tyrimas.

1.3. Darbo tikslas

Pateikti internetinės programinės įrangos tekstinių lokalizuojamųjų išteklių formalizavimo metodą lokalizavimo darbams sisteminti ir lokalizacijų kokybei gerinti, remiantis internetinių programų lokalizavime iškylančių problemų analize.

1.4. Darbo uždaviniai

1. Atlikti internetinės programinės įrangos lokalizavimo ir internacionalizavimo mokslinių ir eksperimentinių pasiekimų analizę.
2. Ištirti internetinių programų lokalizavimo eigą, problemas, lokalizuojamųjų išteklių formatus, išskirti ir susisteminti šių programų internacionalizavimo klaidas.
3. Pasiūlyti internetinės programinės įrangos tekstinių lokalizuojamųjų išteklių metainformacijos formalizavimo metodą, kuriuo remiantis būtų galima pagerinti lokalizacijų kokybę.
4. Atlikti pasiūlyto lokalizuojamųjų išteklių metainformacijos formalizavimo metodo ekspertinį vertinimą.

1.5. Tyrimų metodai

Darbe naudojamos tyrimų metodikos pagrindą sudaro analitiniai, apibendrinamieji, konstruktyvieji ir vertinamieji metodai.

Analizuojant mokslinius ir eksperimentinius pasiekimus programinės įrangos lokalizavimo ir internacionalizavimo srityse naudoti informacijos paieškos, sisteminimo, analizės, lyginamosios analizės ir apibendrinimo metodai.

Tiriant lokalizavimo procesą ir sisteminant internetinės programinės įrangos kultūros elementus naudoti analizės, klasifikavimo ir apibendrinimo metodai.

Kuriant lokalizuojamųjų išteklių metainformacijos formalizavimo metodą naudoti kontekstinės metainformacijos analizės, atributinių gramatikų konstravimo metodai.

Atliekant pasiūlyto metaduomenų formalizavimo metodo eksperimentinį vertinimą naudoti eksperimento, ekspertų apklausos ir apibendrinimo metodai.

1.6. Mokslinis naujumas

Darbe ištirtas programinės įrangos lokalizavimo procesas, išskirti ir suklasifikuoti pagrindiniai internetinės programinės įrangos kultūros elementai, pasiūlytas formaliųjų atributinių gramatikų metodas, skirtas tekstinių lokalizuojamųjų išteklių metainformacijai formalizuoti.

Pagrindiniai šiame darbe pateikto metodo naujumo aspektai:

- Metodo pagrindą sudaro atributinės gramatikos, papildytos naujomis priemonėmis: a) atributai skirstomi į išorinius ir vidinius, b) į gramatiką įtrauktos priemonės skaičiuojamiesiems medžio mazgų atributams papildyti iš išorės įvedamais atributais, c) valdoma konteksto galiojimo sritis. Atributinės gramatikos taikomos ne vertimui automatizuoti, bet tam, kad būtų galima formalizuoti tekstinių lokalizuojamųjų išteklių kontekstinę informaciją ir perteikti ją lokalizuotojui.
- Formalizuojama tekstinių lokalizuojamųjų išteklių eilučių informacija apie kontekstą grafinėje sąsajoje ir semantinė kalbinė informacija.
- Metodas skirtas ir statinių eilučių kontekstui grafinėje sąsajoje nusakyti, ir dinamiškai valdomoms eilutėms grupuoti ir priskirti prie atitinkamų grafinės sąsajos elementų.
- Įtrauktos formalios priemonės statiškai ir dinamiškai parametrizuojamų eilučių parametrų dermei patikrinti.

Metodas skirtas palengvinti lokalizuotojo darbą pateikiant kontekstinę informaciją ir sudaryti prielaidas tvarkingai pateikti programos originalo lokalizuojamuosius išteklius ir tokiu būdu išvengti internacionalizavimo ir lokalizavimo klaidų.

1.7. Ginamieji teiginiai

1. Esami lokalių modeliai ir lokalizuojamųjų išteklių formatai neleidžia perteikti pakankamai informacijos, kad būtų galima tuos išteklius teisingai ir taisyklingai išversti į kitą kalbą (dėl to nukenčia lokalizacijų kokybė arba kelis kartus išauga lokalizavimo darbų sąnaudos dėl testavimo ir klaidų taisymo).
2. Modifikuotomis atributinėmis gramatikomis paremtas metodas leidžia formalizuoti internetinių programų lokalizuojamųjų išteklių metainformaciją.
3. Papildomos sąnaudos, reikalingos lokalizuojamiems ištekliams papildyti formalizuota kontekstine informacija, atsiperka kai lokalizuojama trim ir daugiau kalbų.

1.8. Praktinė darbo reikšmė

Išanalizuoti ir palyginti pagrindiniai lokalių modeliai, kuriuos naudoja programinės įrangos projektuotojai rengdami programinę įrangą lokalizuoti. Išskirtos galimų lokalizavimo ir internacionalizavimo klaidų priežastys, kylančios iš esamų lokalių formaliųjų aprašų dėl

informacijos nepakankamumo, nustatyta, kokios informacijos lokalėse labiausiai trūksta. Išanalizuoti pagrindiniai šiuo metu naudojami programinės įrangos lokalizuojamųjų išteklių pateikimo būdai (formatai), atskyrimo metodai.

Ištirtas programinės įrangos lokalizavimo procesas, pateikta lokalizavimo eigos schema: nuo susipažinimo su programa, jos licencija iki internacionalizavimo problemų formulavimo. Sudarytas internetinės programinės įrangos kultūros svarbiausių problemas keliančių lokalizavimo metu kultūros elementų sąrašas, pasiūlyta kultūros elementų klasifikacija, kuri gali būti naudinga:

- tyrėjams, vertinant programinės įrangos originalo internacionalizavimo lygį arba tikrinant lokalizuotos programinės įrangos kokybę;
- programų projektuotojams, siekiant geriau internacionalizuoti programas;
- programų lokalizuotojams, adaptuojant daugiau kultūros elementų, nustatant internacionalizavimo klaidas.

Metodas, pasiūlytas darbe, skirtas dialogo tekstų vertimui spartinti ir lokalizacijų kokybei gerinti – ypač tai reikalinga lokalizuojant internetinę programinę įrangą, kai dažnai tenka rengti naujas programos versijas, daryti atnaujinimus. Siūloma remtis modifikuotomis formaliosiomis atributinėmis gramatikomis ir jomis aprašyti lokalizuojamuosius išteklius, per atributus įtraukiant lokalizavimo požiūriu naudingą kontekstinę informaciją. Kadangi lokalizavimui svarbūs atributai išreiškiami neprisirišant prie konkrečios kalbos savybių, tai metodas tiks ne tik lietuvių kalbai, bet ir daugeliui kitų fleksinių abėcėlinių kalbų.

Pagrindinė metodo praktinė reikšmė yra ta, kad lokalizuojamieji ištekliai papildomi trūkstama kontekstine informacija, dėl to išteklių pirminio vertimo kokybė pagerėja, o lokalizacijos testavimo ir vertimų koregavimo sąnaudos sumažėja. Internacionalizavimo klaidų aptikimas ir šalinimas perkeliamas į ankstyvąją programinės įrangos projektavimo stadiją.

Atliktas metodo ekspertinis vertinimas, kuriuo nustatyta, kad pritaikius pasiūlytą metodą, bendrosios lokalizavimo darbo sąnaudos pradeda žymiai mažėti, kai programa lokalizuojama ne mažiau kaip 3 kalboms.

1.9. Darbo struktūra

Darbą sudaro: terminų ir santrumpų žodynėlis, įvadas, keturios pagrindinės dalys, bendrosios išvados ir rezultatai, cituotos literatūros ir standartų sąrašas.

Pirmojoje dalyje – įvade – aprašomas darbo aktualumas, iškeliamas darbo tikslas ir uždaviniai, darbo mokslinis naujumas ir praktinė darbo reikšmė, pateikiamas autorės mokslinių publikacijų disertacijos tema ir mokslinėse konferencijose skaitytų pranešimų disertacijos tema sąrašas.

Antrojoje dalyje analizuojami moksliniai literatūros šaltiniai ir standartai, susiję su internetinės programinės įrangos lokalizavimo tema: analizuojamos pagrindinės koncepcijos bei priežastys, dėl kurių būna žema lokalizuotų programų kokybė. Nagrinėjamos svarbiausios programinės įrangos lokalizavimo sąvokos, lokalizavimo laipsniai, lokalių modeliai, mažiau akivaizdi kultūros įtaka lokalizavime – kultūrinės dimensijos, lokalizuojamųjų išteklių plačiau naudojami atskyrimo metodai ir pateikimo formatai bei apžvelgiami lokalizavimo dėsniai.

Trečiojoje dalyje tiriamas programinės įrangos lokalizavimo procesas, išskirti pagrindiniai lokalizavimo etapai, išanalizuoti ir klasifikuoti internetinės programinės įrangos kultūros elementai, išskirtos lokalizavimo problemas labiausiai keliančios priežastys.

Ketvirtojoje dalyje nagrinėjama tekstinių lokalizuojamųjų išteklių struktūra bei pasiūlytas tekstinių lokalizuojamųjų išteklių metainformacijos formalizavimo metodas, skirtas įtraukti kontekstinę informaciją, naudinga lokalizavimo metu. Metodas remiasi modifikuotomis atributinėmis gramatikomis, pateikiami lokalizuojamųjų išteklių gramatikų bendrieji sudarymo principai, išskiriami lokalizavimui svarbūs atributai bei sudarytos atributinės gramatikos dviems apibendrintiems internetinės programinės įrangos fragmentams.

Penktojoje dalyje pateiktas trečiojoje dalyje pasiūlyto metodo taikymo pavyzdys, atliktas metodo ekspertinis vertinimas ir apibendrinti jo rezultatai.

Bendra disertacijos apimtis yra 149 puslapiai.

1.10. Darbo publikavimas ir aprobavimas

Disertacijos rezultatai pateikti 12 mokslinių publikacijų: 6 recenzuojamuose periodiniuose leidiniuose, 3 leidiniuose, įtrauktuose į Mokslinės informacijos instituto konferencijos darbų sąrašą (ISI Proceedings), 3 mokslinių konferencijų darbų leidiniuose. Disertacijos rezultatai pateikti 14 pranešimų 12 tarptautinių ir nacionalinių konferencijų bei seminarų.

Straipsniai recenzuojamuose periodiniuose leidiniuose:

1. Dagienė, V., Jevsikova, T. Cultural Elements in Internet Software Localization. In: Lenca, P., Brézillon, P., Coppin, G. (guest eds.) Revue d'intelligence artificielle. Human-centered processes – Current trends. Volume 23 – no 4/2009. Hermes – Lavoisier, 2009, p. 485–501. ISSN 0992-499X.
2. Jevsikova, T., Dagienė, V. Lokalizuojamųjų programinės įrangos išteklių metainformacijos formalizavimo metodas. Informacijos mokslai. T. 50 (2009), p. 205–211. ISSN 1392-0561.
3. Dagienė, V., Grigas, G., Jevsikova, T. Programinės įrangos lietuvinimas: patirties analizė. Informacijos mokslai. T. 31, 2004, p. 171–184. ISSN 1392-0561.
4. Dagienė, V., Grigas, G., Jevsikova, T. Atvirųjų programų politika švietime. Informacijos mokslai. T. 34, 2005, p. 25–29. ISSN 1392-0561.
5. Dagienė, V., Jevsikova, T. Virtualiosios mokymosi aplinkos lokalizavimo požiūriu. Lietuvos matematikos rinkinys. T. 45, spec. Nr., 2005, p. 197–202. ISSN 0132-2818.
6. Grigas, G., Jevsikova, T. Interneto programų paketo „Mozilla" lokalizavimas ir panaudojimas mokykloje. Lietuvos matematikos rinkinys, T. 42, spec. nr., 2002, p. 241–248. ISSN 0132-2818.

Straipsniai Mokslinės informacijos instituto (ISI) duomenų bazėse referuojamuose konferencijų darbų leidiniuose (ISI Proceedings):

7. Jevsikova, T. Internationalization and Localization of Web-based Learning Environment. In: R. Mittermeir (Ed.) Informatics Education – the Bridge Between Using and Understanding Computers. Lecture Notes in Computer Science (LNCS), Springer Verlag, Berlin, Heilderlberg, Vol. 4226, 2006, p. 310–319. ISSN 0302-9743. [ISI Proceedings]
8. Jevsikova, T., Dagienė, V.. Education in a Networked Society: Virtual Learning Environments and Standards. In: V. Dagienė, R. Mittermeir (Eds.) Informatics in secondary schools: evolution and perspectives: November 7–11, 2006, Vilnius, Lithuania: selected papers, 2006, p. 176–187. ISBN 9955-680-47-4. [ISI Proceedings]

9. Jevsikova, T., Dagienė, V., Grigas, G. Mozilla Internet application suite: developing for education. In: T. Boyle, P. Oriogun, A. Pakstas (Eds.), 2nd International Conference Information Technology: Research and Education, London, 28 June – 1 July, 2004. London Metropolitan University, 2004, p. 96–100. ISBN 0-7803-8625-6. [ISI proceedings]

Straipsniai mokslinių konferencijų pranešimų leidiniuose:

10. Dagienė, V., Jevsikova, T. Cultural Elements in Internet Software Localization. In: Brezillon, P., Coppin, G., Lenca, P. (Eds.) HCP-2008 – Third International Conference on Human Centered Processes. Delft, Netherlands, June 8–12, 2008, p. 275–288. ISBN 978-2-908849-22-6.
11. Jevsikova, T. Understanding Cultural Aspects of ICT for Schools: Naming Issues. In: D. Benzie, M. Iding (eds.) Informatics, mathematics, and ICT: a 'golden triangle' [Elektroninis išteklius]: working joint IFIP conference: WG3.1 secondary education, WG3.5 primary education: Boston, Massachusetts USA, June 27–29, 2007. Boston: Northeastern university, 2007, p. 1–5. ISBN 978-0-615-14623-2.
12. Jevsikova, T. Programų adaptavimas lietuviškai lokalei. Informacinės technologijos 2003, Konferencijos pranešimų medžiaga, ISBN 9955-09-335-8, Kaunas: Technologija, 2003, p. I-(8–14).

Pranešimai tarptautinėse mokslinėse konferencijose:

1. Cultural Elements in Internet Software Localization. 3d International Conference on Human Centered Processes (HCP 2008). Delft (Nyderlandai), 2008 m. birželio 8–12 d.
2. Internationalization and Localization of Web-based Learning Environment. Konferencijoje ISSEP 2006 (Informatics in Secondary Schools: Evolution and Perspectives), Vilnius, 2006 m. lapkričio 7–11 d.
3. Understanding Cultural Aspects of ICT for Schools: Naming Issues. IFIP konferencija „Informatics, Mathematics and ICT: a Golden Triangle", Bostonas (JAV), 2007 m. birželio 27–29 d.
4. An Approach to Combine Learning Entities to Support Mathematics Curriculum in Schools. IFIP konferencija „Informatics, Mathematics and ICT: a Golden Triangle", Bostonas (JAV), 2007 m. birželio 27–29 d.
5. Education in a Networked Society: Virtual Learning Environments and Standards. Konferencijoje ISSEP 2006 (Informatics in Secondary Schools: Evolution and Perspectives), Vilnius, 2006 m. lapkričio 7–11 d.
6. Mozilla Internet application suite: developing for education. Konferencijoje: 2nd International Conference: Information Technology: Research and Education, Londonas, birželio 28 d. – liepos 1 d., 2004.

Pranešimai Lietuvos mokslinėse konferencijose ir seminaruose:

7. Vardų formos lokalizuotose programose. Lietuvos matematikų draugijos XLVIII konferencija. 2007 m. birželio 27–28 d., Vilnius, Vilniaus Gedimino technikos universitetas.
8. Virtualiosios mokymosi aplinkos lokalizavimo požiūriu. Lietuvos matematikų draugijos XLVI konferencija. 2005 m. birželio 15–16 d., Vilnius.
9. Interneto programų paketo „Mozilla" lokalizavimas ir panaudojimas mokykloje. Lietuvos matematikų draugijos XLIII konferencija 2002 m. birželio 17–18 d., Vilnius.
10. Programų adaptavimas lietuviškai lokalei. Konferencijoje: Informacinės technologijos 2003, 2003 m. sausio 28–29 d., Kaunas, KTU.

11. Atvirųjų programų politika švietime. Konferencijoje: Kompiuterininkų dienos 2005. 2005 m. rugsėjo 15–17 d., Klaipėda, KU.
12. „Mozillos“ šeimos programų lokalizavimo ypatumai. Matematikos ir informatikos instituto seminaras. 2007 m. sausio 24 d., Vilnius.
13. „Mozillos“ šeimos programų internacionalizuotumas ir lokalizavimo ypatumai. Matematikos ir informatikos instituto seminaras. 2006 m. vasario 22 d., Vilnius.
14. „Mozilla Firefox“ ir „OpenOffice.org“ lokalizacijų atnaujinimai. Informacinės visuomenės plėtros komiteto prie LR Vyriausybės organizuotame seminare „Lietuvių kalbos vartojimo informacinėje visuomenėje svarba“. 2007 m. gruodžio 11 d., Vilnius.

2. PROGRAMINĖS ĮRANGOS LOKALIZAVIMO ANALIZĖ

Šio skyriaus tikslas – pateikti mokslinės literatūros ir norminių dokumentų, susijusių su lokalizavimu ir internetinės programinės įrangos lokalizavimo ypatumais, analizės rezultatus. Informacijos ieškoma naudojantis tarptautinėmis mokslinių šaltinių duomenų bazėmis, tarptautinių standartų organizacijų ir kt. ištekliais.

Įvairių autorių tyrimai patvirtina, kad lokalizuotų programų kokybė nepakankama ir dėl klaidų, daromų programinės įrangos lokalizavimo metu, ir dėl nepakankamo programų internacionalizavimo [Ca98; Co02; Gri03; DL04; DG06; BB07 ir kt.]. Šiame skyriuje analizuojamos pagrindinės koncepcijos bei priežastys, dėl kurių būna maža lokalizuotų programų kokybė. Nagrinėjamos svarbiausios programinės įrangos lokalizavimo sąvokos, lokalizavimo laipsniai, lokalių modeliai, mažiau akivaizdūs kultūriniai dalykai lokalizavime – kultūrinės dimensijos, lokalizuojamųjų išteklių plačiau naudojami atskyrimo metodai ir pateikimo būdai bei apžvelgiami lokalizavimo dėsniai.

Naujausius lokalizavimo tyrimus, susijusius su internetine programine įranga, galima suskirstyti į šias pagrindines grupes:

1. Žmogaus ir kompiuterio sąveika: kultūrinių skirtumų įtaka projektuojant programinės įrangos sąsają su naudotoju.
2. Bendrieji lokalizavimo ir internacionalizavimo principai.
3. Lokalizavimo patirtis ir konkrečių programų lokalizavimo problemos.
4. Lokalizavimo ir vertimo automatizavimas ar dalinis automatizavimas.

Analizuodami pagrindinį dėmesį skirsime pirmųjų trijų tyrimų grupių rezultatams. Taip pat nagrinėsime su lokalizavimu ir internacionalizavimu susijusius standartus ir specifikacijas.

2.1. Lokalizavimas ir internacionalizavimas

Lokalizavimo ir internacionalizavimo tyrimai prasidėjo XX amžiaus paskutinio dešimtmečio pradžioje. Nuo to laiko paskelbta nemažai mokslinių publikacijų iš programinės įrangos lokalizavimo ir internacionalizavimo sričių, skirtų bendros paskirties programinės įrangos ar interneto svetainių lokalizavimo ir internacionalizavimo įvairiems aspektams tirti. Abi minėtos tyrimų grupės yra glaudžiai susijusios su šiame darbe tiriamu internetinės programinės įrangos lokalizavimu.

Pradinius lokalizavimo tyrimus vykdė stambiosios programinės įrangos projektavimo kompanijos: „Sun Microsystems“, „Microsoft“ ir „Apple Computers“ [Nie90; Mic90; App92]. Vėliau ši sritis tapo konkurencine, dėl to susijusi informacija paprastai buvo prieinama tik tokių kompanijų viduje, neskalbiant jos viešai. Lygiagrečiai su pramonine veikla vyko akademiniai tyrimai, skirti programinei įrangai projektuoti taip, kad ji tiktų įvairioms kalboms ir kultūroms [pvz., RB93; TLB94; ED97]. Programinės įrangos gamintojų publikuojama medžiaga dažniausiai demonstravo konkretaus produkto projektavimo ir lokalizavimo gerąją patirtį. Vėliau, paplitus įvairiems mobiliesiems įrenginiams ir internetui, domėjimasis lokalizavimo ir internacionalizavimo darbais padidėjo. Nemažai dėmesio skiriama atvirajai internetinei programinei įrangai lokalizuoti [DG06; DL04; NWT05; Lan05; LD03; FH05 ir kt.]. Iš programinės įrangos lokalizavimo ir internacionalizavimo sričių parašyta keletas daktaro disertacijų [Atk01; Sul01; Ev01a; Lau07; Gas07], nuo 1995 metų Airijos Limeriko universiteto Lokalizavimo tyrimų centras kasmet rengia tarptautines lokalizavimo konferencijas [pvz., Moo06], su lokalizavimu ir internacionalizavimu susiję

tyrimai pristatomi žmogaus ir kompiuterio sąveikos konferencijų darbuose [pvz., RB07; SJ07; WM07; FG03].

Programinės įrangos *lokalizavimas* įvairių autorių apibrėžiamas panašiai: „programinės įrangos pritaikymas tam tikrai kalbinei ir kultūrinei terpei“ [UHP93; DGJ08; Es00 ir kt.]. Tačiau, kaip rodo konkrečių lokalizuotų programų analizė, ši sąvoka suprantama nevienareikšmiškai. Įvairūs autoriai įdeda skirtingą prasmę į žodį „pritaikymas“ ir frazes „kalbinė ir kultūrinė terpė“. Vieni (autoriai, lokalizuotojai ar programuotojai) tai supranta tik kaip programos tekstų ir žinynų vertimą, kiti – tik kaip vertimą ir adaptavimą kai kurioms kultūrinėms normoms (pvz., datos ir laiko formatai, dešimtainės trupmenos ir tūkstančių skirtukai, koduotės ir pan.). Visa tai – tik dalinis lokalizavimas. Visas šias dalis sujungę gauname visišką programos pritaikymą, kad jos forma ir turinys būtų toks, tarsi ji būtų specialiai sukurta tam tikrai kalbinei ir kultūrinei terpei, kuriai lokalizuojama [Sch02].

Literatūroje lietuvių kalba galima pastebėti vartojamus terminus: *lokalizavimas* ir *lokalizacija*. Šiame darbe naudosime terminą *lokalizavimas*, kai turimas omenyje programinės įrangos adaptavimo kultūrinei terpei procesas, o terminą *lokalizacija* – kai kalbama apie adaptavimo kultūrinei terpei proceso rezultatą arba kai įvardijama programinės įrangos lokalizuota atmaina.

Daugelis autorių [Sul01; Ess00; Sha02; DGJ04; Ya07 ir kt.] sutinka, kad programinės įrangos lokalizavimo darbus galima perskirti į dvi stambias dalis:

1. Programos adaptavimas, kad ji teisingai veiktų konkrečioje kalbinėje ir kultūrinėje terpėje (teisingų kodučių, skaičių ir pinigų sumų formatų, datų ir laiko formatų, pirmosios savaitės dienos ir kt. vartojimas).
2. Dialogo tekstų (meniu, dialogo langų užrašų, įvairių pranešimų, žinyno ir naudotojo vadovo ir kt. susijusių dokumentų) vertimas ir adaptavimas.

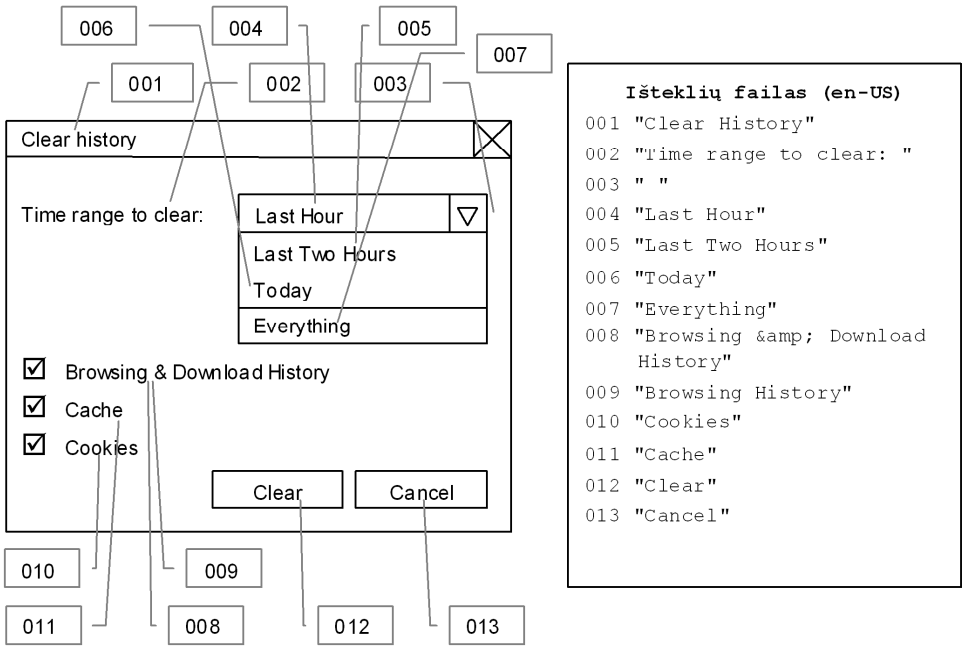
Lokalizavimo įgyvendinimo kokybę įtakoja tai, kiek programinė įranga yra pritaikyta lokalizavimui. *Internacionalizavimas* daugelio autorių apibrėžiamas kaip programinės įrangos projektavimo dalis, pritaikant programinę įrangą tarptautinei rinkai [DGJ08, HHP04 ir kt.]. Kuo geriau programinė įranga yra internacionalizuota, tuo paprasčiau ją lokalizuoti. Šios dvi sąvokos yra glaudžiai susijusios ir dažnai mokslinėje literatūroje vartojamos greta. Internacionalizavimas ir lokalizavimas yra laikomi programinės įrangos produkto elementų atsikyrimu į kultūriniu požiūriu neutralią ir nuo kultūros priklausomas dalis [Ca98]. Todėl programinės įrangos analizė lokalizavimo požiūriu yra kartu ir jos internacionalizacijos tyrimas, kai lokalė nefiksuojama, arba dalinis tyrimas, kai fiksuojama viena ar daugiau lokalių.

Kai kurie autoriai, pavyzdžiui [Tay92], laiko, kad lokalizavimas reikalauja gerokai daugiau pastangų, negu internacionalizavimas, kadangi lokalizuotojų komandos atranda problemas, paliktas internacionalizuotojų komandos.

Galima pastebėti vartojamus terminus: *internacionalizavimas* ir *internacionalizacija*. Šiame darbe naudosime terminą internacionalizavimas, kai turimas omenyje programinės įrangos projektavimo procesas, o terminą internacionalizacija – kai 1) kalbama apie programinės įrangos projektavimo proceso rezultatą arba 2) programinės įrangos savybių, darančių ja lengvai adaptuojama įvairioms kalboms ir kultūroms, visumą [Lau07].

Žemiau pailiustruosime programinės įrangos lokalizavimo principą, remdamiesi paprastu naršyklės fragmento pavyzdžiu. Be abejo, šis pavyzdys apima tik mažą dalį visų lokalizavimo aspektų, tačiau leidžia vaizdžiai pateikti kai kurias problemas, su kuriomis nuolat susiduriama lokalizavimo metu. 1 paveikslo kairėje schematiškai pavaizduotas lokalizuojamos programos langas, dešinėje – tame lange pavartotų tekstų (eilučių) failas, vadinamasis tekstinių lokalizuojamųjų išteklių failas. Kiekviena eilutė turi vardą (šiame

pavyzdyje 001–013) ir tekstą, skirtą rodyti ekrane (tarp stačiųjų kabučių). Etiketės su eilučių vardais iš išteklių failo rodo, kur atitinkama eilutė pasirodo dialogo lange programos veikimo metu. Pastebėsime, kad išteklių failai pateikiami atskirai, lokalizuotojas paprastai dirba su dideliu kiekiu išteklių failų ir eilučių ir paprastai nežino, kur atitinkama eilutė pasirodys programinės įrangos grafinėje sąsajoje. Kokį vertimą parinkti, lokalizuotojas turi spręsti iš eilutės teksto. Be to, dauguma programų lokalizuojamos iki galutinės versijos išleidimo, kai visos funkcijos dar nėra realizuotos, o numatomas funkcijas įvardijantys tekstai jau yra iškelti į lokalizuojamuosius išteklius.

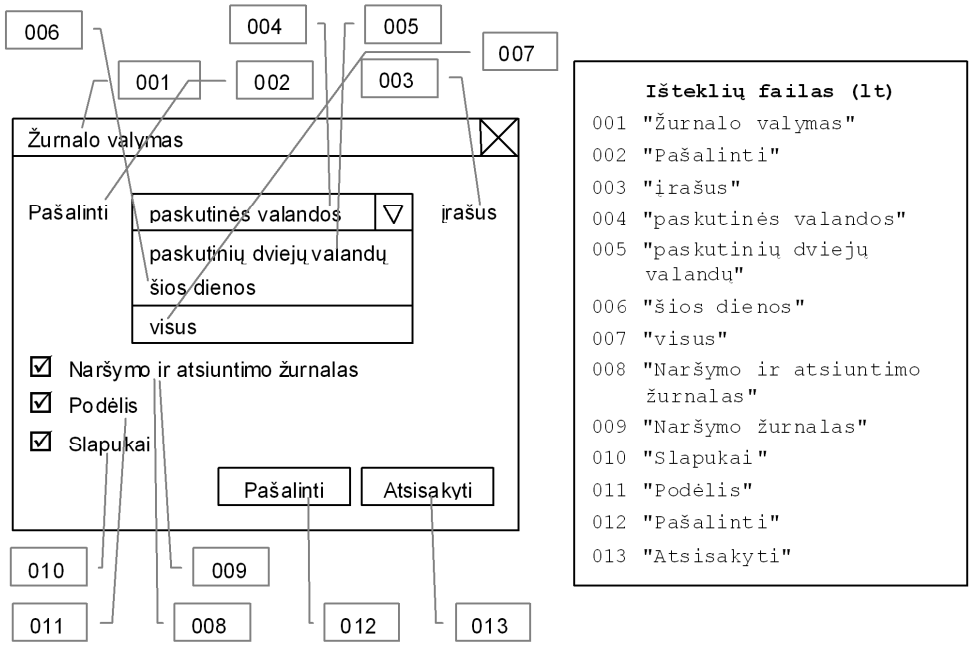


1 pav. Programos grafinės sąsajos fragmento ir atitinkamo išteklių failo pavyzdys

2 paveiksle pavaizduotas to paties lango lokalizuotas lietuvių kalbai (sulietuvintas) variantas (kairėje) ir išteklių failas (dešinėje).

Palyginus anglų (JAV) ir lietuvių išteklių failus, galima pastebėti, kad vartojamos skirtingos frazių formos, pvz., *Today = šios dienos*, skirtingai vartojamos didžiosios ir mažosios raidės, keičiasi net tekstų semantika (plg. 1 pav. išteklių failo 002–003 eilutes ir 2 pav. išteklių failo atitinkamas 002–003 eilutes). Be to, dažnai pasitaiko tekstų alternatyvų, kurios skirtos rodyti ekrane priklausomai nuo konteksto (pvz., 008 ir 009 eilutės rodomos tame pačiame valdiklyje, tačiau kuri bus rodoma priklauso nuo programos naudojimo sąlygų: jei atsiuntimų žurnalas nėra tuščias, tai būtų rodoma 008 eilutė, priešingu atveju – eilutė 009).

Nežinant konteksto, tinkamas lietuvių kalbos frazių formas parinkti yra labai sudėtinga, reikalingas papildomas juodraštinio vertimo testavimo ir vertimo koregavimo darbas, tačiau ne visas eilutes įmanoma pastebėti programos testavimo metu, dalis tekstų pasirodo tik retais atvejais, esant tam tikroms, sunkiai modeliuojamoms situacijoms.



2 pav. Lokalizuotos programos grafinės sąsajos fragmento ir atitinkamo išteklių failo pavyzdys

Šiame darbe dažnai vartosime sąvokas *internacionalizavimo klaida* ir *lokalizavimo klaida*.

Internacionalizavimo klaida laikysime programinės įrangos projektavimo klaidą, kai tam tikra jos savybė arba parametras nenumatomas įvairioms lokalėms pritaikyti. Internacionalizavimo klaidų turinti programinė įranga nėra neutrali kultūriniu ir kalbiniu požiūriais, todėl ji negali būti laikoma visiškai internacionalizuota. Ją sudėtinga ar net neįmanoma (priklausomai nuo klaidų skaičiaus ir sudėtingumo) lokalizuoti. Internacionalizavimo klaidų pavyzdžiai: programos grafinės sąsajos tekstai ir pranešimai neatskirti nuo pirminio programos teksto, trumpmenos skirtukas, laiko formatas, pirmoji savaitės diena ir kt. parametrai nesikeičia, kai programa diegiama įvairių lokalių operacinėse sistemose.

Lokalizavimo klaida – tai programinės įrangos kurios nors savybės neatitikimas lokalei, kai ta savybė yra internacionalizuota. Pavyzdžiui, lokalizavimo klaida gali būti laikoma netinkamai parinkta lokalizuotos programos numatytoji koduotė, kai ta koduotė yra įtraukta į programą, netinkamas datos arba laiko formatas, kai programoje yra galimybė jį pasirinkti, netinkami dokumentų šablonai ir kt. Netaisyklingas programinės įrangos tekstų ir pranešimų, matomų ekrane, vertimas ir komponavimas, netinkamų terminų parinkimas taip pat gali būti priskirtas prie lokalizavimo klaidų.

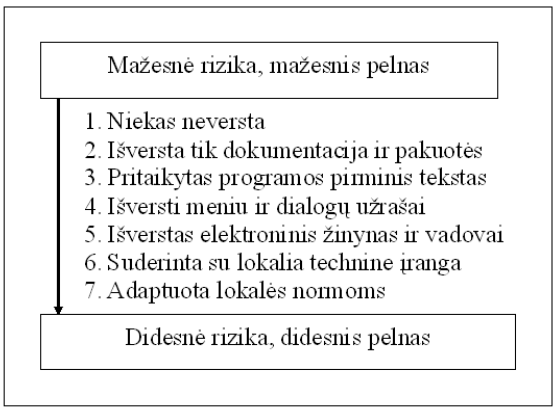
Tiksliau apibrėžti lokalizavimo sąvoką gali padėti jos istorinis vystimasis, aprašytas tolesniame skyrelyje.

2.2. Programinės įrangos lokalizavimo laipsniai

Programinė įranga pradėta lokalizuoti nuo XX a. devintojo dešimtmečio. Didesnis poreikis lokalizuoti programinę įrangą atsirado tada, kai prasidėjo masinis jos eksportas į įvairias valstybes. Augant kompiuterių ir interneto naudotojų skaičiui, šis poreikis vis didėja.

Lokalizuoti programas pradėta palaipsniui. Todėl įvairių lokalizavimo tematikos straipsnių autoriai išskiria kelis lokalizavimo laipsnius. Kai kurie autoriai, pavyzdžiui, [Kan95; Ca98] pagal atliktus lokalizavimo darbus skiria septynis lokalizavimo laipsnius ir sieja juos su rizikos lygiu ir pelnu (3 pav.).

Kuo daugiau išvardytų darbų atlikta, tuo geresnis lokalizavimas. Autorius pažymi, kad tam tikrais atvejais gali būti naudinga lokalizuoti tik iš dalies (pvz., jei programa yra skirta inžinieriams, programuotojams, serverių administratoriams, tai nebūtina lokalizuoti visą programos dokumentaciją).



3 pav. Lokalizavimo laipsniai (adaptuota remiantis [Ca98])

Kiti autoriai skiria lokalizavimo laipsnius pagal jų istorinį įgyvendinimą. Pavyzdžiui, lokalizavimo plitimo pradžioje pagrindinis dėmesys buvo ribojamas galimybe apdoroti lokalės tekstus, vėliau buvo pradėtas įgyvendinti pritaikymas prie įvairių kultūrinių ir kalbinių normų, galiojančių konkrečioje kalboje ar teritorijoje, dar vėliau – visapusiškas programinės įrangos adaptavimas lokalei, tarsi ji specialiai suprojektuota tai lokalei. Grigas skiria tris pagrindinius lokalizavimo laipsnius [Gri98].

Pirmasis laipsnis. Vakarų Europoje buvo įgyvendinamas iš karto, kai tik kompiuteriai buvo pradėti naudoti tekstinei informacija apdoroti, nes programa, negalinti apdoroti tos kalbos rašto ženklų, yra bevertė. Tuo tarpu Vidurio Europos ir Rytų Europos valstybėse dėl žemo jų ekonominio išsivystymo lygio ir nelegalios programinės įrangos paplitimo, procesas vėlavo. Užuot bent dalinai lokalizavus įrangą, ilgą laiką buvo taikstomasi su svetimų rašmenų naudojimu savos kalbos tekstams užrašyti. Pavyzdžiui, rusai rašė lotynų abėcėlės raidėmis, lietuviai raides su diakritiniais ženklais keitė kitomis raidėmis. Dabar šios problemos techniškai jau išspręstos. Tačiau techninių sprendimų vėlavimas pagimdė socialines ir kultūrines problemas. Daugelis kompiuterių naudotojų prisitaikė prie rašto ženklų ignoravimo ir anksčiau egzistavusio ribojimo vartoti kalbos visus ženklus kompiuteryje ir nebesinaudoja elementariomis šiuolaikinių kompiuterių ir programinės įrangos galimybėmis. Todėl reikalingas visuomenės švietimas.

Antrasis laipsnis – tai programinės įrangos pritaikymas prie įvairių kultūrinių ir kalbinių normų, galiojančių konkrečioje kalboje ar teritorijoje (valstybėje, įvairiakalbės valstybės autonominėje dalyje ir pan.): skaičių bei valiutos užrašymo formatai, laiko ir datos formatai, religinių ar kitokių simbolių vartojimo reglamentavimas, specifinių simbolių ir spalvų simbolinės reikšmės, tipinės mandagumo frazės. Kai kurių kultūrinių reikalavimų nepaisymas gali būti rimtų klaidų priežastimi (pvz., neteisingi dešimtainės trupmenos ir tūkstančių skirtukai) arba programą daryti svetimą susiklosčiusioms kultūrinėms tradicijoms. Tokių reikalavimų gana daug. Svarbiausi jų pateikiami kalbų lokalėse (žr. 2.3 sk.). Reikia

pripažinti, kad dar nemažai programų gamintojų savo gaminiuose nenumato tokių reikalavimų ir jų reguliavimo arba lokalizavimo galimybių. Todėl programų lokalizuotojams tikslinga veikti dviem kryptimis: rasti būdus, kaip lokalizuoti šiuos programų elementus, ir programinės įrangos gamintojus informuoti apie tai, kad jie naujas programų laidas geriau pritaikytų lokalizavimui.

Trečiasis laipsnis. Šiuolaikinį kompiuterį galima laikyti intelektualiu prietaisu, atliekančiu aptarnavimo funkcijas. Pavyzdžiui, tekstų rengyklė pakeičia ikikompiuterinių laikų mašininkę, elektroninio pašto programa – laiškanešį ir t. t. Akivaizdu, kad aptarnaujantis personalas privalo mokėti kliento kalbą. Kai kompiuteris pakeičia žmogų, jis privalo „mokėti“ jo aptarnaujamo kliento kalbą – priimti kliento kalba pateikiamas komandas ir klientui teikti informaciją kliento kalba, t. y. visi tekstai, matomi kompiuterio ekrane, turi būti rodomi kliento kalba. Tokių tekstų yra gana daug. Todėl visų jų vertimas ir adaptavimas reikalauja nemažai sąnaudų ir istoriškai buvo pradėtas vėliau. Dabartiniu metu šis laipsnis užima didžiausią lokalizavimo darbų dalį ir visame pasaulyje vyksta labai intensyviai. Šį laipsnį galima laikyti galutiniu lokalizavimo etapu, nes norint jį įgyvendinti reikalingi pirmieji du. Tai pats reikšmingiausias etapas. Todėl toliau kalbėdami apie lokalizavimą turėsime omeny trečiąjį laipsnį kaip galutinį ir problematiškiausią.

Kalbant apie internetinę programinę įrangą, ypač apie veikiančią serveryje, galima siekti trečiojo laipsnio (giluminio) lokalizavimo. Yra įprasta, kad lokalizuojant interneto svetainę, lokalizuojamas tik tas turinys, kuris matomas svetainės lankytojui, pvz., tinklalapio vaizdas naršyklės lange, bet yra pamirštama apie svetainės tinklalapių pirminį tekstą, išteklių universaliuosius adresus, rodomus naršyklės adresų juostoje. Pavyzdžiui, Yijun Yu, Jianguo Lu ir kt. pateikia metodą, kaip lokalizuoti XML dokumentų gaires, ne tik turinį [YJM05]. Naudojant SGML standartą [ISO 8879], kuriame naudojamas turinio ir žymėjimo gairių kodavimas Unikodu [Uni03], kuriami gairių vertimų žodynai ir XSLT stiliai gairėms konvertuoti bei „XPath“ reiškiniai žodyno užklausoms formuoti. Visa tai leidžia sukurti atitinkamus priedus internetinei programinei įrangai, pavyzdžiui, naršyklei, tinklalapių rengyklei, kurie leistų matyti tinklalapio pirminį tekstą lokalės kalba.

Skirtingi lokalizavimo laipsniai gali būti nevienareikšmiško *lokalizavimo* termino supratimo priežastimi [UHP93]. Pavyzdžiui, žemas įvairių programų lokalizacijų lygis paskatino kai kuriuos autorius netgi taikyti alternatyvųjį metodą, analogišką kino filmų subtitrams [LW03]. Programos kiekvienos komandos, su kuria susiduria naudotojas, vertimas pateikiamas dinamiškai (etiketėje). Tai gali turėti taikymų daugiakalbėje kultūrinėje aplinkoje, tačiau neišsprendžia problemų, su kuriomis susiduria lokalizuotojai, o tik atitolina nuo jų sprendimo.

Pirmajam, antrajam ir iš dalies trečiajam lokalizavimo laipsniams įgyvendinti reikalinga informacija – programinėje įrangoje naudojami elementai, kurie yra skirtingi įvairioms kalboms ir kultūroms. Ši informacija yra struktūrizuojama, formalizuojama ir kaupiama naudojantis lokalių modeliais, kurie yra analizuojami šio darbo 2.3 skyrelyje.

Trečiajam lokalizavimo laipsniui realizuoti reikalinga informacija, kuri priklauso nuo gilesnių kultūrinių tradicijų, mąstymo ir suvokimo ypatumų, pateikiama remiantis suformuluotomis kultūrinėmis dimensijomis (2.4 sk.).

Trečiojo lokalizavimo laipsnio negalima pasiekti, jeigu programos ištekliai, kurie nėra neutralūs kalbiniu ir kultūriniu požiūriu (pvz., visi dialogų tekstai, grafika, garsai ir kt.), nebus atskirti nuo programos pirminio teksto. Todėl 2.5 skyrelyje yra analizuojami egzistuojantis tokių išteklių atskyrimo metodai bei lokalizuojamųjų išteklių pateikimo formatai.

2.3. Lokalių modelių analizė

Lokalė – tai pagrindinė koncepcija programinės įrangos lokalizavime. Tarptautiniame ir Lietuvos kultūrinių nuostatų registravimo procedūrų standarte LST ISO/IEC 15897 lokalė apibrėžiama kaip „naudotojo aplinkos poaibio, priklausančio nuo kalbos ar kultūrinių normų, apibrėžimas“. Tačiau ne visas kultūrines normas galima vienareikšmiškai apibrėžti, o tokių normų svarba yra neabejotina kuriant ar lokalizuojant programinę įrangą. Todėl dažnai lokalė yra suprantama plačiau, kaip visų nuo konkrečios vietovės (ar tai būtų valstybė, ar jos dalis) it kalbos priklausančių dialogo su naudotoju aspektų visuma [Jev03]. Enciklopediniame kompiuterijos žodyne [DGJ08] pabrėžiamas kultūrinių normų naudojimas kompiuteryje ir programinėje įrangoje: „kompiuteryje ir jo programinėje įrangoje vartojamų elementų, priklausančių nuo kalbos ir kultūros normų, visuma“.

Lokalėje apibrėžiama kalbos abėcėlė, ženklų koduotės, ženklų rikiavimas, datos ir laiko žymėjimas, ženklų išdėstymas klaviatūroje, pašto adresų forma, laiškų forma, kreipiniai į asmenis ir daugelis kitų konkrečiai kalbai ir kultūrai būdingų dalykų. Kiekviena kalba ir valstybė gali turėti savo lokalę. Viena valstybė, kurioje yra vartojamos įvairios kalbos, gali turėti kelias lokales (pvz., Šveicarijos vokiečių ir Šveicarijos prancūzų). Ta pati kalba, vartojama keliose valstybėse, taip pat gali turėti atskiras lokales (pvz., Prancūzijos prancūzų ir Kanados prancūzų) [DGJ08].

Lokalių elementai, su kuriais dažniausiai susiduriama programinėje įrangoje, yra kaupiami, ieškoma formalių būdų jiems pateikti, standartizuoti jų pateikimą ir naudojimą [Lau03]. Be abejo, visų lokalės elementų neįmanoma aprašyti formaliai, pavyzdžiui, kultūrai būdingo mąstymo stiliaus įtakojamų programinės įrangos funkcijų, grafikos elementų, spalvų vartojimo, tam tikros kalbos eilučių visų komponavimo atvejų, visų žodžių formų, todėl formalizuojami tik pagrindiniai lokalių elementai.

Lokalė paprastai yra identifikuojama nurodant kalbą ir teritoriją. Dažniausiai naudojami dviraidžiai kalbų kodai, apibrėžti standarte ISO 639-1, ir dviraidžiai valstybių ar vietovių kodai, apibrėžti standarte ISO 3166-1. Pavyzdžiui, Lietuvos lokalė paprastai nurodoma eilute „lt_LT“, JAV anglų – „en_US“, D. Britanijos anglų – „en_GB“.

Lokalėse apibrėžiami kultūriniai elementai bei pateikiamos priemonės šiems elementams formaliai aprašyti. Šiame skyrelyje analizuojami svarbiausi dokumentai ir projektai, skirti lokalės duomenų aprašams apibrėžti. Dalis šių dokumentų yra tarptautiniai standartai arba specifikacijos. Nagrinėjami pagrindiniai lokalių modeliai, kuriais naudojamasi projektuojant internacionalizuotą programinę įrangą: nuo programavimo kalbos nepriklausomos POSIX ir FDCC lokalės, Javos lokalė (kadangi nemažai internetinių programų projektuojama remiantis Javos technologijomis), tarptautinio standarto ISO/IEC 15897:1999 apibrėžiamas lokalių modelis (kultūrinių elementų registravimo procedūra) ir CLDR lokalė kaip pagrindas didžiausiai šiuo metu egzistuojančiai lokalių duomenų bazei kurti. Esama ir kitų lokalių modelių – tai įvairių programavimo kalbų ir programavimo priemonių naudojamos lokalės, kurių atskirai nenagrinėsime dėl jų struktūros panašumo į čia analizuojamas pagrindines lokales.

2.3.1. POSIX lokalė

POSIX (angl. *Portable Operating System Interface for Computer Environments*) standartas vienas pirmųjų pateikė priemones lokalės formaliam aprašui suvienodinti. Jos buvo įtrauktos į POSIX standartų grupę, kuriamą Tarptautinės standartų organizacijos (ISO) ir tarptautinės elektrotechnikos komisija (IEC) bei Elektrotechnikos ir elektronikos inžinierių instituto (IEEE). Pirmoji standarto versija pasirodė 1988 m. Su lokale yra susijusios pirmos dvi dalys: POSIX.1 (atitinka ISO/IEC 9945-1) ir POSIX.2 (ISO/IEC 9945-2), apibrėžiančios

portatyvių programų sąsają, komandas ir priemones. Naujausios šių standartų versijos buvo priimtos 2003 metais: ISO/IEC 9945-1:2003 ir ISO/IEC 9945-2:2003.

POSIX standarte pateiktas lokalės modelis sudaro programavimo kalbos C lokalės pagrindą, kuris yra apibrėžtas standarte ISO/IEC 9899:1999, todėl naudojamas įvairiose sistemose veikiančių C kompiliatorių. POSIX lokalės modelis taip pat naudojamas UNIX sistemose, „Microsoft Windows“ operacinės sistemos turi su POSIX lokale suderinamą posistemį.

POSIX lokalėje naudojamos šios kategorijos (1 lentelė).

1 lentelė. POSIX lokalės kategorijos

Lokalės kategorija	Aprašas
LC_CTYPE	Ženklų klasifikacija (mažosios ir didžiosios raidės, dešimtainiai ir šešioliktainiai skaitmenys, skyrybos ženklai, tarpai, spausdinami ženklai ir kt.), ženklų transformacijos ir kiti ženklų požymiai
LC_COLLATE	Ženklų rikiavimo taisyklės (ženklų sekų rikiavimas, rikiavimo svoriai, ženklų susiejimas, ženklų ekvivalenčių klasių apibrėžimas ir kt.)
LC_MONETARY	Pinigų sumų užrašymo formatai (valiutų simboliai), dešimtainės trupmenos ir tūkstančių skirtukai pinigų sumose, skaitmenų grupavimas, teigiamos ir neigiamos reikšmės, valiutos ženklo padėtis (prieš arba po sumos) ir kt.
LC_NUMERIC	Skaičių rašymo formatai (dešimtainės trupmenos sveikosios ir trupmeninės dalies skirtukas, tūkstančių skirtukas, skaitmenų grupavimo taisyklės)
LC_TIME	Datos ir laiko formatai (savaitės ir mėnesių sutrumpinti ir nesutrumpinti pavadinimai, datos ir laiko komponentų tvarka, 12 ar 24 val. laiko formatas, eros pavadinimas ir kt.)
LC_MESSAGES	Informatyvių pranešimų formatai (teigiami ir neigiami atsakymai)

Standarte pateiktos formalios taisyklės kiekvienam kategorijos elementui aprašyti.

POSIX lokalių modelio privalumai:

- Tai tarptautinis standartas, kuris pirmasis formaliai apibrėžė svarbiausius lokalės elementus ir su kuriuo suderinama nemažai projektuojamos programinės įrangos.
- Portatyvumas: nepriklausomumas nuo operacinės sistemos ar kt. platformos.

Tačiau yra ir trūkumų:

- Kategorijos negali būti išplėstos (arba kopijuojamos, arba aprašomos iš naujo).
- Galimas nesuderinamumas su koduotėmis (nėra natūralios galimybės nurodyti Unikodo kodus).
- Per mažai kategorijų. Jos neapima pagrindinių su kalba ir vietove susijusių kultūrinių normų.

2.3.2. FDCC lokalė

Šis lokalės modelis pateiktas tarptautiniame standarte ISO/IEC 14652: kultūrinių susitarimų formalių apibrėžčių rinkinys (Set of Formal Definitions of Cultural Conventions). Tai POSIX ir C lokalės, apžvelgtos aukščiau, viršaibis, skirtas panaikinti POSIX lokalės modelio trūkumus. Pirmosios šešios kategorijos yra POSIX lokalės kategorijos, kitos – naujos kategorijos (2 lentelė).

Be to, POSIX kategorijos standarte ISO/IEC 14652 yra išplėstos, pavyzdžiui, pinigų sumų kategorijoje yra galimybė nurodyti, ar yra tarpas tarp sumos ir valiutos ženklo, ar nėra, ar naudojami kiti ženklai valiutos ženklu atskirti. Skaičių formatų kategorija papildyta

galimybe užrašyti sudėtingesnes skaitmenų grupavimo taisykles. Datos ir laiko kategorijoje įvesta galimybė aprašyti savaites, kurios yra ne septynios dienos. Ženklų rikiavimo kategorijoje įtraukti efektyvesni rikiavimo būdai, įvestos ženklų transliteracijos taisyklės. Ženklai gali būti nurodomi Unikodo kodų kaitos sekomis, <Uxxxx> arba <Uxxxxxxxx>.

2 lentelė. FDCC rinkinio kategorijos

Kategorija	Aprašas
<POSIX kategorijos>	Žr. 1 lentelę
LC_IDENTIFICATION	Metainformacija apie lokalę: kūrėjas, lokalės kodas
LC_ADDRESS	Pašto adresų užrašymo taisyklės, valstybių pavadinimai, priimtos valstybių pavadinimų santrumpos, naudojamos įvairiose srityse, pvz., ant transporto priemonių, ISBN numerių, kalbų pavadinimai, kalbų pavadinimų santrumpos ir kt.
LC_MEASUREMENT	Informacija apie naudojamą matavimo sistemą
LC_NAME	Asmenvardžių ir titulų užrašymo taisyklės, kreipiniai į vyrus ir moteris, į vyrą, netekėjusias ir ištekėjusias moteris, universalus kreipinys į moteris
LC_PAPER	Dokumentams naudojamas numatytasis popieriaus lapo dydis
LC_TELEPHONE	Telefonijos paslaugų naudojami formatai: tarptautinis formatas, lokalus formatas, kodų prefiksai

Standartas patobulina POSIX lokalės modelį, atsirado svarbių naujų kategorijų, tačiau jis nėra visiškai suderinamas su POSIX programomis.

2.3.3. Javos lokalė

Javos programavimo kalba ir programavimo terpė naudoja savą objektinį lokalės modelį, apibrėžtą Javos kalbos specifikacijoje. Lokalės kategorijas atitinka Javos klasės, turinčios įvairių metodų darbui su lokalės elementais. Naudojami atvirieji, laisvai platinami lokalių aprašai (aprašyta daugiau kaip 100 lokalių), kurių duomenys pasiekiami naudojantis Javos lokalės klasėmis ir metodais, įtrauktais į *java.text* ir *java.utils* pakuotes.

Javos lokalės modelį sudaro tokie lokalės elementai [DC01]: skaičių formatai (dešimtainės trupmenos skirtukai ir tūkstančių skirtukai), datos ir laiko formatai (trumpasis ir ilgasis), valiutų simboliai, kalendoriaus elementai, laiko juostos, rikiavimas ir kt. veiksmai su tekstu, grafinių komponentų išdėstymas (pvz., iš kairės į dešinė ar iš dešinės į kairę), lokalės išteklių (teksto, grafikos, garso, vaizdo įrašų ir kt.) atskyrimas. Klasės, skirtos išvardytoms lokalių kategorijoms, nurodytos 3 lentelėje.

3 lentelė. Javos lokalės kategorijos

Kategorija	Klasės ir metodai
Lokalės informacija	Klasė: `Locale`
Skaičių ir pinigų sumų formatai	Klasė `NumberFormat`, Metodai: `getNumberInstance`, `getPercentInstance`, `getCurrencyInstance`, `getIntegerInstance`, `Format`
Datos ir laiko formatai	Klasės: `DateFormat`, `SimpleDateFormat`, Metodai: `getDateInstance`, `getTimeInstance`, `getDateTimeInstance`, `Format`
Kalendorius	Klasės: `Calendar`, `GregorianCalendar`
Laiko juostos	Klasė: `TimeZone`
Rikiavimas ir kt. veiksmai su tekstu	Klasės: `Collator`, `RuleBasedCollator`, `BreakIterator` Metodai: `GetInstance`, `Compare`, `getWordIterator`, `getLineIterator`, `getSentenceIterator`, `getCharacterIterator` ir kt.

Kategorija	Klasės ir metodai
Grafinių elementų išdėstymas	Klasė: `ComponentOrientation`, Metodas: `GetOrientation`
Lokalės išteklių atskyrimas	Klasės `ResourceBundle`, `ListResourceBundle`

Lokalizuojamiesiems ištekliams atskirti Javos modelyje naudojami išteklių rinkiniai. Tekstiniai ištekliai atskiriami įtraukiant juos į PROPERTIES failus, o nuo lokalės priklausomi multimedijos ištekliai atskiriami naudojant klasę *ListResourceBundle*.

Javos lokalės modelyje naudojama daugiau elementų negu POSIX, tačiau mažiau, lyginant su FDCC, bet pateikiamos konkrečios darbo su lokalės elementais kalbos konstrukcijos, nuo pat pradžios įvestas Unikodas. Kai kurias FDCC kategorijas, kurių nėra Javos modelyje, galima realizuoti kitomis priemonėmis, pvz., asmenvardžio užrašymo taisykles – per parametrizuotas eilutes.

2.3.4. ISO/IEC 15897:1999 lokalė

Tarptautinis standartas ISO/IEC 15897:1999 „Kultūros elementų registravimo procedūra“ apibrėžia kultūros elementų, pateiktų tiek nusakomuoju tekstu, tiek formaliai, registravimo procedūras. 2001 m. šis standartas buvo priimtas Lietuvos standartu: LST ISO/IEC 15897:2001. Standartas yra suderinamas su ISO/IEC 9945-2 (POSIX) aprašytomis kategorijomis, tačiau leidžia registruoti daugiau lokalės elementų. Registravimo rezultatai yra laisvai prieinami, pvz., programinės įrangos gamintojai gali laisvai jais pasinaudoti.

Pagal šį standartą gali būti registruojami keturi kultūros specifikacijų tipai:

1. Nusakomoji kultūros specifikacija.
2. POSIX lokalė.
3. POSIX ženklų koduotė.
4. POSIX ženklynas.

Nusakomoji kultūros specifikacija apibrėžia kultūros normas pasakojamąja anglų kalba, kartu ji gali pateikti ir ekvivalenčius apibrėžimus kitomis kalbomis. Ji taikytina tada, kai trūksta formaliais kultūros normų specifikavimo metodais paremtos kodifikacijos. Jei kai kurie nusakomosios kultūros specifikacijos elementai taip pat apibrėžiami POSIX lokalėje ar POSIX ženklų koduotėje, tai specifikacija turi nurodyti į tą lokalę ar ženklų koduotę.

2, 3 ir 4 tipai yra aprašomi naudojant kultūros elementų POSIX specifikacijas, apibrėžtas ISO/IEC 9945-2 standarte.

Nusakomosios kultūros specifikacijos pirmieji šeši skyriai sutampa su POSIX lokalės kategorijomis, kiti skyriai skirti išsamesniam atitinkamuose pirmuose šešiuose POSIX skirsniuose išvardytų kultūros aspektų aprašymui (4 lentelė). Aprašymai pateikiami neformaliai, tačiau formatu, kurį galėtų apdoroti kompiuteris.

4 lentelė. Nusakomosios kultūros specifikacijos elementai

Specifikacijos skyrius	Aprašas
POSIX lokalės kategorijų skirsniai	Žr. 1 lentelę
Nacionalinė ir regioninė informacinių technologijų terminija	Gali būti išvardyta terminai bendrine kalba ir regioniniai jų variantai, pavyzdžiui, informacinių technologijų ISO terminų vertimai
Nacionaliniai ir regioniniai standartų profiliai	Gali būti išvardyti standartų profiliai, pavyzdžiui, atvirųjų sistemų sujungimo (OSI) nacionaliniai profiliai arba POSIX standartų profiliai

Specifikacijos skyrius	Aprašas
Ženklų rinkinio aptarimas	Aprašoma rašto ženklų vartosena, pavyzdžiui, kokie rašto ženklai būtinai reikalingi konkrečiai kalbai; kokie kiti ženklai vartojami kalbai niuansuoti; kokiais ženklais įprasta laikraščiuose ir knygose rašyti asmenvardžius bei vietovardžius; kokie ženklai vartojami senuosiuose raštuose; kitiems tikslams vartojami ženklai
Rikiavimo ir paieškos taisyklės	Taisyklės, kaip skaidyti įrašą į rikiavimo laukus, kurie specialūs žodžiai ignoruojami atliekant palyginimo ar paieškos veiksmus. Čia taip pat gali būti aprašytos palyginimo taisyklės, kuriose remiamasi tarimu
Ženklų transformacijos	Ženklų transliteracijos bei transformacijos, pavyzdžiui, lotynų, graikų ir kirilicos rašmenų transliteracijos, arba kai kurių raidžių pakaitalai
Ženklų savybės	Papildoma informacija apie ženklų savybes, pvz., kaip mažosios raidės, neturinčios didžiųjų atitikmenų, turi būti rašomos ten, kur įrasta rašyti didžiąsias raides
Specialiųjų ženklų vartojimas	Kabučių, santrumpų ženklų bei skirtukų ir kt. ženklų vartosena
Ženklų vaizdų formavimas	Kokios ženklų vaizdavimo alternatyvos laikomos adekvačiomis ir kokie ženklų grafiniai vaizdai priimtini
Ženklų įvedimas	Įvedimas klaviatūra ir kiti ženklų įvedimo metodai
Asmenvardžių sudarymo taisyklės	Taisyklės, kas laikoma pavarde, kaip tituluojama, ar leistina pavardes rašyti vien didžiosiomis raidėmis, ar rašomi pilni vardai, ar inicialai. Čia taip pat galima pateikti taisykles, nurodančias, kaip vaikai paveldi tėvo ir motinos pavardę, kokios būna sutuoktinių pavardės
Kaityba	Pateikiamos kalbos kaitybos taisyklės arba nuorodos, kur jas rasti
Žodžių kėlimas	Žodžių kėlimo taisyklės arba kur jas galima rasti
Rašyba	Rašybos taisyklės ir rašybos žodynai bei nuorodos į ortografijos šaltinius
Numeravimas, kelintiniai skaitvardžiai bei matų sistemos	Matų sistemos (paprastai tai yra ISO SI sistema)
Pinigų sumos	Papildoma POSIX kategorija, pvz., senoviški piniginiai vienetai
Data ir laikas	Papildoma POSIX atitinkama kategorija joje nenumatytais datų rašymo formatais, laiko juostų pavadinimais, vasaros laiko įvedimo taisyklėmis bei kitomis rašytinėje kalboje vartojamomis išraiškomis
Informacija apie valstybę	Įvairios žinios apie valstybę, pavyzdžiui: pašto indeksai, administracinio suskirstymo padalinių kodai, policijos skyrių kodai, teritorijų pavadinimų santrumpos
Telefonų numeriai	Vietinių ir tarptautinių telefonų numerių rašymo formatai
Pašto adresai	Pašto adresų rašymo forma: kur rašomas adresatas, gatvės pavadinimas ir pašto indeksas, pastato aukštų pavadinimai ir panaši informacija
Asmenų ir organizacijų identifikavimas	Identifikavimo sistemos, pavyzdžiui, socialinio draudimo numeriai, įmonių kodai ir kt.
Elektroninio pašto adresai	El. pašto adresų reglamentacija
Sąskaitų numeriai	Valstybei būdinga banko sąskaitų numerių sandara
Ženklų išdėstymas klaviatūroje	Aprašomas ženklų išdėstymas klaviatūroje
Žmogaus ir kompiuterio dialogas	Gali būti aprašyta, kaip lokalizuoti programas
Popieriaus lapo formatai	Popieriaus lapo formatai (paprastai ISO standartai), vokų su langeliais naudojimas ir pan.
Maketavimo dalykai	Aprašoma, kaip maketuoti, pavyzdžiui, verslo laišką arba faksogramą

Šio standarto privalumai:

- Suderinamas su POSIX lokalės modeliu.
- Teikia priemonių aprašyti nemažai svarbių papildomų kultūros elementų, kurių nėra nei POSIX, nei FDCC, nei Javos lokalėse.

Nepaisant šių privalumų, lokalių duomenys nėra aktyviai registruojami, taigi programinės įrangos projektuotojai negali realiai jais pasinaudoti.

2.3.5. CLDR lokalė

CLDR (Common Locale Data Repository) projektas prasidėjo 2004 metais, tęsiant 2003 metais „OpenI18N“ grupės pradėtą kurti XML lokalės duomenų saugyklą. Projektą vykdo Unikodo konsorciumas [Uni07]. CLDR tikslai: pateikti priemones bendriems programinėje įrangoje naudojamiems įvairių pasaulio lokalių duomenims specifikuoti, sukaupti kuo daugiau lokalių duomenų. Lokalių duomenų mainams naudojamas XML formatas (lokalės duomenų žymėjimo kalba LDML) [Dav07], o duomenys laisvai prieinami internete. Naudojamas duomenų kodavimo būdas – UTF-8.

5 lentelė. LDML lokalės kategorijos

LDML kategorijos	Lokalės elementas
<identity>	Informacija apie lokalę ir lokalių identifikatoriai
<localeDisplayNames>	Kalbų, kalbų grupių, valstybių, teritorinių vienetų, kalendorių, laiko juostų, valiutų pavadinimų vertimas į lokalės kalbą
<layout>	Teksto kryptis, programų grafinės sąsajos elementų išdėstymas
<characters>	Lokalės kalboje vartojami ženklai ir jų klasifikacija (sudaro kalbos pagrindiniai ir papildomi ženklai)
<delimiters>	Kabutės, įdėtinės kabutės
<measurement>	Matavimo sistema, dokumentų lapų dydžiai
<dates>	Datos ir laiko ilgieji ir trumpieji formatai (įskaitant savaitės dienų pavadinimus, šablonus). Naudojamas kalendorius, erų pavadinimai, mėnesių pavadinimai ir jų santrumpos, metų ketvirčių ar jų ekvivalentų pavadinimai, savaitės dienų pavadinimai ir jų santrumpos, savaitės pirmoji diena, laiko juostos. Datos formatai remiasi Javos lokalės modelyje naudojamais formatais
<numbers>	Skaičių ir pinigų sumų formatai (skaitmenų grupavimas, skirtukai, teigiamos ir neigiamos sumos, valiutos simbolis ir jo padėtis ir kt.)
<posix>	Ši kategorija įtraukta suderinamumui su POSIX lokale užtikrinti, naudojamos POSIX lokalės kategorijos (1 lentelė)
<collations>	Ženklų rikiavimas, paieška, lyginimas ir transformacijos

Tai didžiausia šiuo metu egzistuojanti lokalės duomenų bazė (2009 m. pateikta daugiau kaip 100 lokalių formalių aprašų). CLDR projektas taip pat teikia priemones duomenims eksportuoti į su POSIX suderinamą formatą, Javos naudojamus išteklių rinkinius, raštinės paketo „OpenOffice.org“ naudojamą lokalės formatą. Nuo 2006 m. CLDR lokalė naudojama „MacOS X“, „Solaris“, „AIX“ operacinėse sistemose, „Acrobat“, „ModernBill“ programinėje įrangoje, „OpenOffice.org“ raštinės programų pakete [Moo06].

2.3.6. Lokalių modelių kultūrinių elementų palyginimas

Pateiksime ankstesniuose skyreliuose analizuotų lokalių modelių lyginamąją lentelę (6 lentelė). Lokales lyginsime pagal kairiajame lentelės stulpelyje pateiktų lokalių elementų buvimą (žymimą +) ar nebuvimą (žymimą –) atitinkamame lokalės modelyje. Naudosime iš visų lokalių generalizuotą elementų grupių sąrašą.

6 lentelė. Analizuotų lokalių modelių palyginimas

Lokalės elementai	POSIX	FDCC	Java	ISO/IEC 15897:1999	CLDR
1. Ženklai, jų klasifikacija ir transformacijos	+	+	+	+	+
2. Ženklų rikiavimas ir paieška	+	+	+	+	+
3. Pinigų sumų užrašymo formatai	+	+	+	+	+
4. Skaičių rašymo formatai	+	+	+	+	+
5. Datos ir laiko formatai, savaičių, mėnesių, erų pavadinimai	+	+	+	+	+
6. Laiko juostos ir jų pavadinimai	–	–	+	+	+
7. Teigiami ir neigiami atsakymai	+	+	+	+	+
8. Pašto adresai, valstybių ir kalbų pavadinimai	–	+	+	+	+
9. Matavimo vienetai	–	+	+	+	+
10. Asmenvardžių, titulų, kreipinių užrašymo taisyklės	–	–	–	+	+
11. Popieriaus formatai	–	+	–	+	+
12. Telefonų numerių ir kodų formatai	–	+	–	+	+
13. Grafinių elementų išdėstymo kryptis	–	–	+	+	+
14. Informacinių technologijų terminija	–	–	–	+	–
15. Nacionaliniai standartai	–	–	–	+	–
16. Specialiųjų ženklų ir skirtukų vartojimas	–	–	–	+	+
17. Ženklų vaizdų formavimas	–	–	–	+	–
18. Ženklų įvedimas, klaviatūros, išdėstymas klaviatūroje	–	–	–	+	–
19. Kaityba ir rašyba (nuorodos į taisykles)	–	–	–	+	–
20. Informacija apie valstybę	–	–	–	+	–
21. Elektroninio pašto adresai	–	–	–	+	–
22. Banko sąskaitų numeriai	–	–	–	+	–
23. Maketavimo ypatumai	–	–	–	+	–
24. Lokalės išteklių atskyrimo priemonės	–	–	+	–	–
Iš viso	6	10	10	23	14

ISO/IEC 15897:1999 standartas numato priemones didžiausiam lokalių elementų grupių skaičiui apibrėžti lokalėje, tačiau nėra plačiai vartojamas. CLDR lokalė numato priemonės pateikti 14 iš lentelėje išskirtų 24 lokalės elementų grupių informacijai ir turi sukauptų ir laisvai prieinamų lokalių aprašų.

2.4. Kultūrinės dimensijos lokalizavime

Kultūros elementų, įtrauktų į lokalių modelius, nepakanka norint visiškai lokalizuoti programinę įrangą. Nemažai tyrimų [Cha07; RB07; FG03; CT07; BB07 ir kt.] patvirtina sunkiau formalizuojamų, gilesnių kultūrinių dimensijų svarbą projektuojant ir lokalizuojant programinę įrangą, ypač skirtą internetui.

Hofstede pagal įvairių kultūrų žmonių mąstymo, pojūčių ir veiklos šablonus sukūrė penkių kultūrinių dimensijų modelį, išnagrinėjo 53 valstybes ir apskaičiavo šių dimensijų lygius, atspindinčius kultūrinių skirtumų šaknis [Hof91; HH05]. Šias kultūrines dimensijas sudaro:

1. Galios atstumas (nurodo, kiek svarbi ir aiški yra hierarchija visuomenėje, ar aukšta žmonių diferenciacija). Ši dimensija daugiau išreikšta Lotynų Amerikos, Azijos ir Afrikos valstybėse, mažiau – germanų valstybėse. Aukščiausią galios atstumo lygį turi Malaizija.
2. Individualizmas (lyginant su kolektyvizmu). Individualių arba kolektyvinių pasiekimų vertinimas. Individualistinės kultūros vertina asmeninį laiką, laisvę, privatumą, tuo tarpu kolektyvizmas nusako stiprų priklausymą grupėms nuo pat gimimo, aukštai vertinama grupių gerovė, grupės nuomonė. Individualizmas labiau pasireiškia išsivysčiusiose ir Vakarų valstybėse, o kolektyvizmas – besivystančiose ir Rytų valstybėse. Pavyzdžiui, Japonija turi vidutinį individualizmo lygį, JAV – aukščiausią, o žemiausią – Lotynų Amerikos šiaurės-vakarų valstybės, taip pat Pakistanas, Indonezija.
3. Vyriškumas (lyginant su moteriškumu). Atspindi lyčių vaidmenis visuomenėje. Aukštesnį vyriškumo lygį turinčiose kultūrose, moterys ir vyrai turi aiškiai paskirstytus vaidmenis, tuo tarpu žemesnį vyriškumo lygį turinčiose kultūrose moterų ir vyrų vaidmenys yra panašūs. Pavyzdžiui, aukščiausią vyriškumo lygį turi Japonija, o žemiausią – Švedija. Gana aukštu vyriškumo lygiu pasižymi Vokietija, Austrija.
4. Nežinomybės (neapibrėžtumo) vengimas. Atspindi, kiek visuomenė linkusi vengti nežinomybės, dviprasmybės. Aukštesnio nežinomybės vengimo lygio kultūrose stengiamasi aiškiai apibrėžti elgesio taisykles, įstatymus. Žemesnio nežinomybės vengimo lygio kultūrose labiau toleruojama įvairovė, pokyčiai ir eksperimentavimas. Nežinomybės vengimo lygis yra aukštas Lotynų Amerikos valstybėse, Japonijoje ir vokiškai kalbančiose valstybėse. Žemesnis – Šiaurės šalių, britų, Kinijos kultūrose.
5. Ilgalaikė perspektyva (lyginant su trumpalaike). Atspindi visuomenės „laiko horizontą“, praeities, dabarties ir ateities svarbą. Ilgalaikės perspektyvos kultūrose labiau vertinamas taupumas, ištvermingumas, atkaklumas, pragmatizmas (aukščiausius įvertinimus gavo Kinija, Japonija, Taivanis, Šiaurės Korėja). Žemiausias ilgalaikės perspektyvos lygis buvo nustatytas besivystančiose valstybėse (Pakistanas) ir Vakarų valstybėse (JAV).

Tai reiškia, kad lokalizuojant programą, reikia kreipti dėmesį ne tik į pagrindinius (formaliai ar neformaliai išreikštus) lokalės elementus, bet ir į gilesnius kultūrinius skirtumus. Pavyzdžiui, lokalizuojant interneto svetainę, galbūt tektų parinkti kitokius paveikslus, sukurti kitokią naršymo struktūrą, labiau priimtiną paskirties kultūroje. Tam tikrais atvejais reikėtų net iš naujo suprojektuoti kai kurias programinės įrangos funkcionalumo dalis, grafinę sąsają. Tai patvirtina, kad turi vykti glaudus lokalizuotojų ir programų projektuotojų ryšys.

Marcus ir Gould [MG01] analizuoja kultūrinių dimensijų įtaką svetainėms ir portalams bei siūlo klausimų sąrašą, į kuriuos vertėtų atsakyti prieš projektuojant arba lokalizuojant produktą tam tikrai kultūrai.

Trompernaar [Tro97] ir Hall [HH90] pasiūlė atitinkamai septynių ir keturių dimensijų kultūros modelius. Mohd Isa ir Md Noor apibendrino Hofstede, Hall ir Trompenaar

kultūrines dimensijas ir, vadovaudamiesi jomis, sukūrė rekomendacijas interneto svetainių naršymo schemai bei turiniui projektuoti [WM07]. Pavyzdžiui, svetainėse, skirtose kultūroms, kurių aukštas nežinomybės vengimo lygis, reikėtų naudoti ribotą spalvų skaičių, nepiktnaudžiauti garso įrašais, informaciją dėstyti dalimis pagal temas, įtraukti su šalies tradicijomis susijusias temas, klientų konsultavimo paslaugą, platinamų produktų demonstracinių versijų parsisiuntimą bandymui, mažinti naudotojų galimų „klaidų“ skaičių. Tuo tarpu, jei svetainė skirta žemo nežinomybės vengimo lygio kultūros atstovams, tai informaciją derėtų dėstyti pagal užduotis, informaciją perteikti įvairiais būdais: naudojant spalvas, garso įrašus, animaciją.

Lanier apžvelgia Rytų ir Vakarų kultūrinius skirtumus, partikuliarizmo ir universalizmo, individualizmo ir kolektyvizmo (pagal [HH90; Tro97]), skirtingų kultūrinių vertybių poveikį programinės įrangos projektavimui ir naudojimui [Lan05]. Iškelia idėją, kad programinė įranga ne visada ir ne visoms kultūroms įmanoma kokybiškai lokalizuoti, dėl to kartais verta kurti atskiras programų versijas specialiai skirtas toms kultūroms. Pabrėžiama, kad dabartinių lokalizacijų kokybę mažina tai, kad šiuo metu dauguma lokalizavimo sprendimų priima ne konkrečios kultūros (kuriai lokalizuojama) atstovai, o programinės įrangos projektuotojai (pvz., dirbantys Jungtinėse Amerikos Valstijose). Dėl šių priežasčių atvirosios programos yra lankstesnės, nes lokalizuojant galima suprojektuoti iš naujo arba pakoreguoti neatitinkančias lokalei dalis. Pvz., iliustruojama, kad „Microsoft“ korporacijos operacinės sistemos nėra skirtos kolektyvinėms nuostatoms realizuoti (naudotojų profiliai pabrėžia individualumą). Autorius siūlo teikti nebaigtus kurti produktus, kad juos būtų galima baigti projektuoti ir lokalizuoti konkrečiose valstybėse.

Reinecke ir Bernsteinas [RB07] savo straipsnyje siūlo idėją sukurti tokią programinę įrangą, kuri savaime prisitaikytų prie naudotojo kultūrinių normų. Pasinaudojant aukščiau minėtomis kultūrinėmis dimensijomis, sukuriamos kultūrinės žymės – programinės įrangos sąsajos elementai, pageidautini tam tikroje kultūroje, naudojami natotojo modeliavimo modeliai ir sąsajos pritaikymo pasiekimai iš dirbtinio intelekto sričių. Kaip pagrindas imami naudotojo valstybė ir kalba, o visi kiti elementai modeliuojami naudojant stereotipus, bendruomenes, analizuojant naudotojų sąveiką su sistema. Straipsnyje pateikiama tik idėja, tačiau tokios realizacijos dar nėra sukurta.

Hoffman [Hof07] pastebi, kad programinės įrangos gamintojai, norėdami sumažinti programinės įrangos, skirtos lokalizavimui, eilučių žodžių skaičių, dažnai pakeičia juos grafine (vaizdine) informacija (piktogramomis, paveikslais). Tačiau klaidinga nuomonė, jog grafiškai pateiktos informacijos nereikia lokalizuoti (net jeigu joje ir nėra panaudota teksto). Pabrėžiama, kad lokalizuoti reikia ne tik lingvistinėms naudotojų grupėms (kaip jau yra įprasta), bet ir grupėms, kurių narius jungia kiti požymiai, pvz., amžius. Amžiaus grupėms yra ypač svarbu parinkti tinkamas piktogramas ir kt. sąsajos elementus.

Auer ir Dick atliktame tyrime [AD07] dalyvavo respondentai iš JAV, Vokietijos ir Kinijos. Kiekvienam dalyviui buvo pateikti tradicinių piktogramų rinkiniai, iš kurių jis turėjo parinkti jo manymu tinkamą piktogramą nurodytai funkcijai atlikti. Eksperimentui buvo parinktos 28 funkcijos. Tyrimu nustatyta, kad esama piktogramų interpretavimo skirtumų, priklausančių nuo respondentų valstybės (respondentai iš Kinijos atpažino interpretavo pateiktas programas kitaip ir atpažino mažiau piktogramų). Buvo naudojamos ir nuo kultūros priklausomos piktogramos, ir nepriklausomos. Tai reiškia, kad JAV kuriamos piktogramos nėra vienodai interpretuojamos Kinijos naudotojų, ir tai reikia numatyti projektuojant programinę įrangą.

Evers daktaro disertacijos nuotolinio mokymo internetinės aplinkos naudotojų tyrimu [Ev01a] nustatyta, kad metaforiškai pateikiami programinės įrangos sąsajos elementai skirtingų kultūrų atstovams kelia skirtingas asociacijas, o tą patį tikslą pasiekti dažnai

naudojami skirtingi veiksmai. Tą patį patvirtina ir elektroninio mokymosi svetainių lokalizavimo tyrimas [MT05].

2.5. Lokalizuojamieji ištekliai ir jų pateikimo formatai

Planuojamos lokalizuoti programinės įrangos kūrėjai turi ją internacionalizuoti. Vienas svarbiausių šio darbo etapų – lokalizuojamųjų išteklių atskyrimas nuo pirminių programos tekstų. Tai visų tekstų, grafikos, garsų, lokalės elementų, pagalbinių parametrų ir kt. elementų, pateikiamų naudotojui veikiant programai, atskyrimas nuo programos vykdomosios dalies – iškėlimas į atskirus failus arba atskiras vykdomųjų failų sekcijas (išteklių sekcijas).

Lokalizuojamieji programos ištekliai gali būti tekstiniai ir dvejetainiai. Jų pateikimas priklauso nuo programavimo kalbos, kuria parašyta programinė įranga, naudojamo kompiliatoriaus, platformos, kuriai kuriama programinė įranga, taip pat naudoto išteklių atskyrimo metodo. Kai kurie lokalizuojamųjų išteklių atskyrimo metodai leidžia realizuoti tik tekstinių išteklių pateikimą. O dvejetainiai objektai (pvz., grafika, vaizdo, garso įrašai) gali būti pateikiami tekstiniu pavidalu: jų universaliuoju identifikatoriumi vidinėje arba tinklo išorinėje lokalizuojamųjų išteklių sistemoje (URI).

Šiame skyrelyje vartojamos sąvokos *lokalizuojamųjų išteklių atskyrimo metodas* ir *lokalizuojamųjų išteklių pateikimo formatas*. Išteklių atskyrimo metodas teikia veiksmus ir priemones lokalizuojamiems ištekliams atskirti ir pateikti (įskaitant ir išteklių pateikimo formatą). Formatas yra daugiareikšmė sąvoka. Čia išteklių pateikimo formatu laikysime lokalizuojamuose ištekliuose laikomų duomenų apipavidalinimo būdą. Šis būdas gali priklausyti nuo išteklių atskyrimo metodo (t. y. formatą apibrėžia išteklių atskyrimo metodas), tačiau yra lokalizuojamųjų išteklių pateikimo formatų, kurie gali būti naudojami su įvairiais atskyrimo metodais.

2.5.1. Lokalizuojamųjų išteklių atskyrimo metodai ir pateikimo formatai

Yra įvairių lokalizuojamųjų išteklių pateikimo formatų, kurie priklauso nuo programavimo kalbos, kuria parašyta programa, naudojamo kompiliatoriaus, platformos, kuriai kuriama programinė įranga, išteklių atskyrimo metodo. Prieš nagrinėjant lokalizuojamųjų išteklių formatus naudinga apžvelgti pagrindinius programinės įrangos internacionalizacijos tipus.

Laucius savo disertacijoje [Lau07] remdamasis Taylor [Tay92] skiria tris internacionalizacijos tipus (jie kartu gali būti laikomi ir programinės įrangos išteklių atskyrimo tipai):

1. Internacionalizacija kompiliavimo metu.
2. Internacionalizacija susaistymo metu.
3. Internacionalizacija vykdymo metu.

Iš jų pastarieji du tipai (saistymo ir vykdymo metu) nuo Tayloro knygos publikavimo metų (1992 m.) dėl programinės įrangos ir kompiliatorių vystimosi susiliejo į vieną.

Internacionalizacija kompiliavimo metu – tai programos projektavimo būdas, kai lokalizavimui skirti ištekliai neatskiriami nuo pirminio programos teksto: teksto eilutės, kurios bus matomos kompiuterio ekrane, įkompiliuojamos į pirminį programos tekstą. Tuomet lokalizuojant programą daromos atskiros kopijos kiekvienai lokalei, pirminiame tekste randamos lokalizuotinos eilutės, jos lokalizuojamos (verčiamos), ir programa perkompiliuojama. Lokalizuoti programą, neperkompiliuojant jos, neįmanoma. Lokalizavimo metu atsiranda pavojus pažeisti pirminį programos tekstą.

Tokį metodą vadinti internacionalizacija yra daugiau nei simboliška, kadangi šiuo metu programą, kurioje tekstiniai ir kt. nuo kalbos ir kultūros priklausantys ištekliai nėra atskirti nuo pirminio teksto, negalima vadinti internacionalizuota. Tačiau tenka pastebėti, kad net ir dabar pasitaiko tokių programų, kuriose lokalizuotinos eilutės tebėra paliktos pirminių programos tekstų lygyje [LD03]. Taip pat įvairių programų lokalizavimo patirtis [DJ05, Jev03 ir kt.] rodo, kad net jeigu lokalizuojamieji ištekliai atskiriami nuo pirminių tekstų, kai kurie tekstai būna pamirštami, ir dėl to norint juos lokalizuoti tenka modifikuoti pirminį programos tekstą. Pagrindinė priežastis, dėl kurios net dabar pasirenkamas toks programos projektavimo būdas, tai spartesnis pirmos versijos išleidimas: nesirūpinant išteklių atskyrimu ir projektavimu tarptautinei rinkai, programos pirmoji versija išleidžiama sparčiau. Tačiau vėliau, kai atsiranda lokalizavimo poreikis, autoriai bando iškelti lokalizuotinus išteklius, o dėl nemetodiško pradinio projektavimo dalis jų lieka pirminių tekstų lygyje.

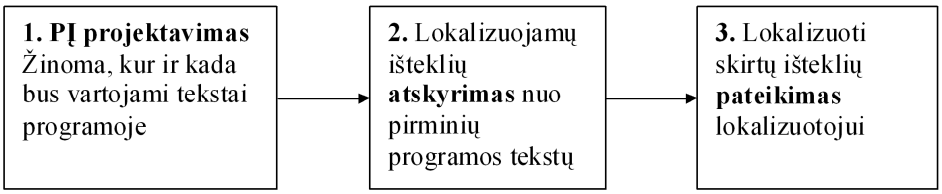
Internacionalizacija susaistymo ir vykdymo metu pasižymi tuo, kad lokalizavimui skirti ištekliai yra atskiriami nuo pirminio programos teksto. Norint įtraukti lokalizuotus išteklius, pirminio programos teksto nereikia perkompiliuoti. Lokalizavimui skirti ištekliai yra įkompiliuojami į vykdomąsias programas arba į vykdymo metu prie programų prijungiamas bibliotekas, išteklių paketus arba duomenų bazes.

Tokiu būdu parengtas programas galima suskirstyti į lokalizuotas ir daugiakalbes [KS97]. Lokalizuotos programos realizuojamos saistant atitinkamas išteklių bibliotekas, kurios pakeičia originalias programos funkcijas, neatitinkančias lokalę. Daugiakalbės programos naudoja išorinius pranešimų ir išteklių failus, kurie gali būti platinami atskirai ir pakeičiami ar prijungiami prie programos ją vykdant.

Kiekvienas stambesnis programinės įrangos gamintojas sukuria savo išteklių atskyrimo metodą ir išteklių pateikimo lokalizavimui formatą. Atvirųjų programų kūrėjai taip pat kuria ir naudoja savo formatus. Tačiau šiuo metu egzistuojantys formatai yra panašūs tuo, kad jie pateikia lokalizuotinus išteklius be konteksto arba tik su menkomis užuominomis apie kontekstą, kuriame tam tikras pranešimas bus naudojamas programoje.

4 paveiksle pavaizduotas programinės įrangos parengimo lokalizavimui procesas: projektavimo metu yra žinomas visų eilučių ir kt. nuo lokalės priklausomų elementų kontekstas programoje, tada lokalizuojamieji ištekliai yra atskiriami nuo pirminio programos teksto ir tam tikru formatu (kuriuos apžvelgsime toliau) pateikiami lokalizuotojams.

Tarp 2 ir 3 žingsnio prarandamas lokalizuojamųjų išteklių elementų kontekstas. Taip pat atskirti ištekliai lokalizavimui gali būti pateikiami išteklių atskyrimo metodo numatomu formatu arba transformavus į kitą formatą (pvz., paprasčiausią tekstinį). Pastaruoju atveju prarandama dar daugiau kontekstinės informacijos, kuri galėtų būti naudinga lokalizavimui.



4 pav. Programinės įrangos parengimo lokalizavimui proceso pagrindinių etapų schema

Autoriai [AFO08] taip pat tvirtina, kad pagrindinės problemos, su kuriomis susiduriama projektuojant tarptautinei rinkai skirtą programinę įrangą, – tai informacijos perdavimo problemos tarp įvairių projektavimo etapų ir darbuotojų komandų.

Toliau apžvelgsime pagrindinius lokalizuojamųjų išteklių atskyrimo metodus ir formatus. Santykis tarp išteklių atskyrimo metodo ir formato yra tas, kad vienas metodas gali turėti keletą jam būdingų lokalizuojamųjų išteklių pateikimo failų formatų. Paprastai išteklių atskyrimo metodas turi numatytąjį lokalizuojamųjų išteklių pateikimo formatą, tačiau projektuotojai gali pasirinkti pateikti išteklius lokalizavimui ne metodo numatytu formatu, o konvertuoti į kitą, pavyzdžiui, paprastesnį tekstinį formatą. Iš programos lokalizuojamųjų išteklių pateikimo formato ne visada galima vienareikšmiškai nustatyti išteklių atskyrimo metodą.

Pateiksime lentelę, kurioje išvardyti pagrindiniai lokalizuojamųjų išteklių atskyrimo metodai ir atitinkami metodo numatytieji lokalizuojamųjų išteklių pateikimo failų formatai, kurie analizuojami tolesniuose skyreliuose (7 lentelė).

7 lentelė. Lokalizuojamųjų išteklių atskyrimo metodai ir atitinkami failų formatai

Išteklių atskyrimo metodas	Pagrindiniai numatytieji išteklių failų formatai
RC	RC, Resources, EXE, DLL
RESX	RESX, Resources, EXE, DLL
GNU „gettext“	PO, POT, MO
Javos išteklių rinkiniai (resource bundles)	PROPERTIES
Mozilla	DTD, PROPERTIES
PHP	PHP
XLIFF	XLIFF

Lentelėje paminėtas XLIFF nėra priskiriamas prie išteklių atskyrimo metodų (2.5.7. sk.), jo pavadinime vartojamas žodis „formatas“ (angl. *XML Localization Interchange File Form*at), nes nenumato priemonių ištekliams atskirti, o nurodo taisykles ištekliams pateikti. Papildžius XLIFF atitinkamomis išteklių atskyrimo priemonėmis, jį galima būtų laikyti ir išteklių atskyrimo metodu.

2.5.2. RC

„RC“ lokalizuojamųjų išteklių atskyrimo metodas sukurtas „Microsoft“ bendrovės ir paprastai naudojamas „Microsoft“ programinėje įrangoje bei programinėje įrangoje, skirtoje „MS Windows“ operacinėms sistemoms. Metodo pavadinimas kilo iš lokalizuojamųjų išteklių scenarijaus failo prievardžio „.rc“. Tai tekstinis failas, kuriame gali būti naudojamos vienbaitė, dvibaitė koduotės arba Unikodas [Mic09]. Scenarijus apibrėžia teksto eilutes ir išorinius lokalizuojamus išteklius (piktogramas ir kt. grafinius išteklius, garso ir vaizdo išteklius, dialogo langus, šriftus ir kt. – iš viso 15 numatytųjų išteklių tipų). „RC“ scenarijus turi keletą sekcijų, iš jų tris pagrindines sekcijas, kuriose atsiranda lokalizuotinos eilutės: programos meniu ištekliai, statinių dialogo langų sekcija ir teksto eilučių (pranešimų) sekcija.

Išteklių scenarijai „RC“ yra kompiliuojami, ir ištekliai yra susiejami su vykdomaisiais failais ar dinaminėmis bibliotekomis (DLL). Tada lokalizuojamieji ištekliai pateikiami kaip atskiros dinaminių bibliotekų arba vykdomųjų failų sekcijos (išteklių sekcijos). Išteklius galima lokalizuoti arba iš pradinių tekstinių failų, arba iš dvejetainių failų pasitelkus specializuotas išteklių rengykles – tokių išteklių modifikavimo vizualias priemones.

Išteklių scenarijaus meniu sekcijoje (žr. pavyzdį 5 pav.) kiekvienas meniu elementas turi jį atitinkančią originalo eilutę, pvz., „&Save“, prieigos klavišą, kuris žymimas ampersando ženklu, parašytu prieš klavišą įvardijantį ženklą. Taip pat nurodomi komandų klavišai, pvz., „Ctrl+S“.

```
6 MENU
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
{
POPUP "&File"
{
  MENUITEM "&New...\tCtrl+N", 57600
  MENUITEM "&Open...\tCtrl+O", 57601
  MENUITEM "&Save\tCtrl+S", 57603
  MENUITEM "Save &As...", 57604
  MENUITEM SEPARATOR
  MENUITEM "&Print...\tCtrl+P", 57607
  MENUITEM "Print Pre&view", 57609
  MENUITEM "Page Set&up...", 32771
  MENUITEM SEPARATOR
  MENUITEM "Recent File", 57616, GRAYED
  MENUITEM SEPARATOR
  MENUITEM "Sen&d...", 57612
  MENUITEM SEPARATOR
  MENUITEM "E&xit", 57665
}
MENUITEM SEPARATOR
MENUITEM SEPARATOR
}
```

5 pav. „MS Windows“ RC išteklių failo meniu sekcijos pavyzdys

Dialogo lango sekcija (6 pav.) turi valdiklio pavadinimą, valdiklio koordinates ir dydžio parametrus bei tekstą, vaizduojamą tame valdiklyje. Yra numatyti 19 valdiklių tipų: 11 pagrindinių valdiklių tipų (mygtukas, išskleidžiamasis sąrašas, žymimasis langelis ir kt.) bei jų variacijos (pvz., centruotas teksto užrašas, lygiuotas pagal kairįjį kraštą teksto laukas, trijų būsenų valdiklis – 3 žymimųjų akučių rinkinys ir kt.).

```
143 DIALOG 0, 0, 185, 112
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUPWINDOW | WS_VISIBLE | WS_DLGFRAME
CAPTION "Paragraph"
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
FONT 8, "MS Shell Dlg"
{
  GROUPBOX   "Indentation", 1022, 7, 7, 114, 75
  LTEXT   "&Left:", -1, 13, 21, 29, 8
  EDITTEXT   1000, 52, 19, 60, 14, ES_AUTOHSCROLL
  LTEXT   "&Right:", -1, 13, 42, 29, 8, NOT WS_GROUP
  EDITTEXT   1001, 52, 40, 60, 14, ES_AUTOHSCROLL
  LTEXT   "&First line:", -1, 13, 63, 32, 8, NOT WS_GROUP
  EDITTEXT   1002, 52, 60, 60, 14, ES_AUTOHSCROLL
  LTEXT   "&Alignment:", 1017, 13, 91, 37, 8
  COMBOBOX   111, 52, 89, 60, 45, CBS_DROPDOWN | WS_VSCROLL
  DEFPUSHBUTTON   "OK", 1, 128, 10, 50, 14
  PUSHBUTTON   "Cancel", 2, 128, 27, 50, 14
}
```

6 pav. „MS Windows“ RC išteklių failo dialogo lango sekcijos pavyzdys

Eilučių sekcijoje pateikiamos eilutės, kurios paprastai nėra statiškai priskirtos prie atitinkamų statinių dialogo langų valdiklių (7 pav.). Į šią sekciją patenka programos klaidų pranešimo eilutės (176–182 eilutės pavyzdžio paveiksle), klausimai programos naudotojui, įvairių veiksenų pavadinimai (183–165), sąrašų elementai (189–191), parametrizuotos eilutės (179–182), dinamiškai dialogo languose atsirandančios eilutės (186–188).

```
STRINGTABLE
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
{
176, "There are too many tab stops set in this paragraph."
177, "An error occurred while sending the document."
178, "There is not enough memory. Quit one or more programs and then try again."
179, "Can not load %1 files."
180, "Unable to open %1.  There are too many files already open."
181, "Unable to create %1. This folder is full. Use another folder or delete some files
182, "The document %1 is in use by another application and cannot be accessed."
183, "Text"
184, "Rich Text"
185, "Word"
186, "Options"
187, "Write"
188, "Embedded"
189, "Text Document"
190, "Rich Text Document"
191, "Word 6 Document"
}
```

7 pav. „MS Windows“ RC išteklių failo teksto eilučių sekcijos pavyzdys

Eilučių sekcija yra pati sudėtingiausia lokalizavimo požiūriu [Ess00, p. 63], kadangi joje nebėra grafinės sąsajos konteksto, kuris iš dalies buvo pateikiamas .RC dialogo langų ir meniu sekcijose.

Pranešimams dažnai trūksta aiškumo, norint juos kokybiškai lokalizuoti reikia nemažai programavimo, konkrečios programos ir konteksto žinių. Vienas iš pavyzdžių galėtų būti vieno ar dviejų žodžių eilutės, pvz., „None“, kuri verčiama į kitas kalbas keliais skirtingais žodžiais arba frazėmis, atsižvelgiant į kontekstą. Toks žodis kaip „Copy“ gali būti verčiamas kaip veiksmažodis, daiktavardis arba sudaryti dalį kurios nors kitos frazės, skirtos rodyti ekrane. Todėl tenka nemažai laiko investuoti į lokalizuotos programos testavimą.

Pastebėsime, kad semantinio eilučių konteksto nėra pateikiama nei vienoje iš .RC sekcijų. Tačiau šio metodo privalumas, palyginus su kitais metodais – tai numatyta galimybė atskirti ne tik tekstinius išteklius, bet ir dvejetainius, ir naudoti egzistuojančias vizualias priemones statiniams dialogo langams lokalizuoti.

2.5.3. RESX

„RC“ metodo atmaina, kai išteklių apibrėžimo scenarijui aprašyti naudojama XML kalba [Hal07]. Formatas buvo sukurtas naudoti „.NET“ platformai kuriamų programų lokalizuojamiems ištekliams atskirti, tačiau gali būti naudojamas ir kitų programų.

Ištekliams pateikti naudojamas „.Resx“ failų formatas, kuriame objektai ir eilutės apibrėžiamos XML gairėmis. Į „.resx“ failą gali būti įterpta ir dvejetainių objektų (paveikslų, garsų, piktogramų ir pan.). Išteklių failą galima taisyti įprasta tekstų rengykle arba specialiomis priemonėmis, vizualizuojančiomis ir dvejetainius objektus.

Lokalizuojamieji ištekliai pateikiami vardų ir jų reikšmių poromis (8 pav.).

```
< ?xml version="1.0" encoding="utf-8" ? >
<root>
   <xsd:schema > …. </xsd:schema>
   …
  <data name="download">
    <value>Drop a link or file to download it</value>
  </data>
…
</root>
```

8 pav. RESX formato išteklių fragmento pavyzdys

Tekstinėms eilutėms nurodomas vardas ir eilutė, dvejetainiams objektams – vardas ir MIME tipas bei pateikiamas pats objektas kaip įterptasis dvejetainis kodas.

Lokalizuotojams .RESX formato ištekliai gali būti eksportuojami į tekstinį failą, lentelę ar tiesiogiai. Pastaruoju atveju jie taisomi naudojant specializuotas tokių išteklių rengykles. .RESX failas taip pat gali būti kompiliuojamas į dvejetainį „resources“ tipo failą, įkompiliuojamas į vykdomuosius failus (EXE) arba dinamines bibliotekas (DLL).

2.5.4. GNU „Gettext“

GNU „Gettext“ metodas ir kartu priemonių paketas leidžia automatiškai atskirti programos grafinėje naudotojo sąsajoje naudojamas teksto eilutes nuo pirminio programos teksto. Paplitęs atvirųjų programų lokalizuojamiems tekstiniams ištekliams pateikti. Yra sukurta „Gettext“ priemonių paketų ir bibliotekų, pateikiančių API įvairioms programavimo kalboms (Javai, C, C++, Paskaliui, PHP, Phytonui, Perlui ir kt.) ir jų realizacijoms.

GNU „Gettext“ priemonių paketas teikia [DMP07]:

1. taisyklių rinkinį, kaip rašyti programas ir atskirti tekstinius išteklius į „gettext“ katalogus;
2. eilučių katalogų failų ir katalogų struktūros bei jų vardų sudarymo taisykles;
3. biblioteką kreipiniams vykdymo metu į išverstas eilutes;
4. keletą atskirų programinių priemonių eilučių vertimams tvarkyti;
5. biblioteką, skirta tvarkyti išverstų eilučių failų kūrimui tvarkyti;
6. „Emac“ priemones, skirtas eilutėms atnaujinti.

Eilučių sąrašas gaunamas automatiškai iš programos tekste panaudotų eilučių, ištraukiant jas į vieną ar daugiau pranešimų šablonų failų (vadinamųjų POT failų, angl. *Portable Object Template*). Eilučių atskyrimas yra automatizuotas: naudojamos specialios funkcijos „_“ arba „gettext“, pvz., _(„Eilutės tekstas“). Šios funkcijos leidžia ir iškelti teksto eilutes iš pirminio programos teksto, ir pasiekti eilutes bei lokalizuotas jų versijas, kai reikia jas parodyti ekrane. Tokiu būdu eilutės vardu išteklių sistemoje tampa visa eilutė. Jeigu nėra eilutės vertimo, tai naudojamas originalios eilutės tekstas.

Lokalizuotojai dirba su PO formato failais, kurie yra sukuriami iš POT šablono ir yra jam identiški pagal formatą. POT šablonas yra laikomas pagrindine eilučių saugykla, kurioje laikomos tik originalo eilutės (be jų vertimų), o PO failai – tai šio šablono kopijos, su kuriomis dirba įvairių kalbų lokalizuotojai ir kurios papildomos eilučių vertimais. Kai reikia atnaujinti lokalizuojamųjų eilučių failą, atnaujinamas POT šablonas, o iš jo naujų eilučių originalo kalba tekstas importuojamas į PO failus.

PO formato failas – tai tekstinis failas, kuriame kiekvieną įrašą atitinka keletas failo teksto eilučių – laukų. Vienoje eilutėje rašomas tekstas originalo kalba, kitoje – teksto vertimas (9 pav.).

Papildomi laukai žymimi specialiaisiais simboliais. Pavyzdžiui, eilutėje, prasidedančioje ženklu „#“ rašomi vertėjo komentarai, po ženklų poros „#.“ rašomi automatiniai komentarai, kuriuos sukuria PO failą tvarkanti programa, po ženklų poros „#,“ rašomos įrašo gairelės, pavyzdžiui, gairelė „fuzzy“ reiškia tikslintiną eilutę. Specialios programos (pvz., „Poedit“) gali interpretuoti PO failo įrašus ir vaizdžiai juos pateikti lentele. Jos turi priemonių įrašams ieškoti, tvarkyti ir kt.

Iš PO tekstinių failų yra kompiliuojami dvejetainiai MO tipo failai, kurie skirti platinti su programa.

```
#: ./skins/plone_scripts/folder_cut.cpy:32
msgid "${count} item(s) cut."
msgstr "Iškirpta elementų: ${count}."

#: ./skins/plone_scripts/folder_rename.cpy:58
msgid "${count} item(s) renamed."
msgstr "Pervardyta elementų: ${count}."

#: ./skins/plone_scripts/folder_delete.cpy:49
msgid "${items} could not be deleted."
msgstr "${items} negalima pašalinti."

#. Default: "${monthname} ${year}"
#: ./skins/plone_portlets/portlet_calendar.pt:56
msgid "${monthname} ${year}"
msgstr "${year} ${monthname}"

#: ./skins/plone_scripts/check_id.py:62
msgid "${name} is not a legal name. The following characters are invalid:
${characters}"
msgstr "„${name}“ varde yra neleistinų ženklų: ${characters}"

#: ./skins/plone_scripts/check_id.py:70
msgid "${name} is reserved."
msgstr "„${name}“ yra rezervuotas vardas."
```

9 pav. Gettext PO formato pavyzdys (turinio valdymo sistemos *Plone* eilučių failo fragmentas)

Tai, kad pati eilutė laikoma ir jos vardu PO faile, palengvina darbą programuotojams, tačiau yra ir „gettext“ metodo trūkumas, kadangi ta pati anglų kalbos eilutė, naudojama skirtingose programos vietose, dažnai būna verčiama į kitas kalbas įvairiais skirtingais būdais, pvz., „Open file“ – „Atverti failą“ (meniu komanda), „Failo atvėrimas“ (langa pavadinimas). Norint įtraukti keletą vienodų eilučių, reikia įdėti papildomų programavimo pastangų.

Kitas „gettext“ metodo trūkumas yra tas, kad jis numato tik tekstinių lokalizuojamųjų išteklių atskyrimą. Kiti nuo lokalės priklausomi ištekliai (paveikslai, piktogramos, garsai ir kt.) turi būti atskiriami kitais metodais, kuriuos pasirenka programuotojas. Taip pat šis metodas nenumato lokalės svarbiausių elementų (datos ir laiko formatai, piniginiai vienetai ir kt., žr. 2.3 sk.), todėl dažniausiai kartu su juo naudojamas POSIX lokalės modelis.

„Gettext“ metodo privalumas yra tas, kad jame galima naudoti skirtingas formas žodžių, vartojamų su skaitvardžiu (pvz., 1 objektas, 2 objektai, 10 objektų). Kalbos formų kaitymas aprašomas scenarijumi, kurį galima įterpti į PO failo antraštės sekciją ir naudoti programoje. Pavyzdžiui, scenarijus lietuvių kalbai atrodytų taip:

```
Plural-Forms: nplurals=3; plural=n%10==1 && n%100!=11 ? 0 :
n%10>=2 && (n%100<10 || n%100>=20) ? 1 : 2.
```

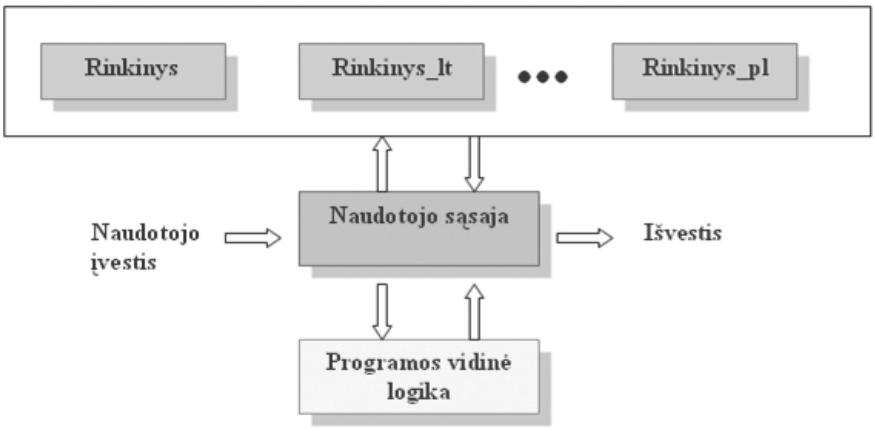
2.5.5. Javos išteklių rinkiniai

Javos programavime naudojamas lokalizuojamų išteklių atskyrimo metodas (angl. *Resource Bundles*), kuris skirtas atskirti tekstui ir dvejetainiams nuo lokalės priklausomiems objektams, pvz., paveikslams, piktogramos, garso ir vaizdo failams.

Pagrindinės išteklių rinkinio savybės [DC01]:

- Priemonės lokalės informacijai laikyti ir užklausti;
- Leidžia programoje naudoti daugelį lokalių;
- Leidžia pridėtų naujų lokalių.

Javos internacionalizuotos programos išteklių struktūra pavaizduota 10 paveiksle.



10 pav. Javos lokalizuojamųjų išteklių struktūra programoje (adaptuota remiantis [DC01])

Ištekliams atskirti naudojama Javos abstrakčioji klasė *ResourceBundle*. Ištekliai gali būti pateikiami dvejopai: kaip Javos klasės arba kaip tekstiniai failai su prievardžiu *.properties*. Tekstinių išteklių failų formatas – eilučių vardų ir eilučių tekstų porų sąrašas (11 pav.). Papildomai toks failas gali turėti komentarų. Šiuo metu tokie failai paprastai koduojami UTF-8 koduote, tačiau anksčiau buvo naudojamas ženklų, kurių nėra ASCII koduotėje, keitimas Unikodo kaitos sekomis *\u{ženklo kodas Unikode}*, pvz., vietoje lietuviškos raidės *ė* būtų rašoma *\u0117*. Toks kodavimas tebenaudojamas kai kurių programų.

```
nv_done=Done
nv_timeout=Timed Out
nv_stopped=Stopped
openFile=Open File

droponbookmarksbutton=Drop a link to bookmark it
dropondownloadsbutton=Drop a link or file to download it
droponnewtabbutton=Drop a link or file to open it in a new tab
droponnewwindowbutton=Drop a link or file to open it in a new window
droponhomebutton=Drop a link or file to make it your home page
droponhometitle=Set Home Page
droponhomemsg=Do you want this document to be your new home page?
```

11 pav. Javos PROPERTIES failo formato pavyzdys

Lokalizuotiems ištekliams įkelti programos vykdymo metu naudojami tam skirti specialieji Javos klasės metodai. Esant reikalui, klasę *ResourceBundle* galima praplėsti pritaikant konkretiems išteklių atskyrimo poreikiams.

Išteklių rinkinių metodas pastaraisiais metais taikomas ne tik Javos programavime. Panašus principas gali būti naudojamas ir programuojant C bei C++ kalbomis, taip pat yra panašumų „Mozilla“ išteklių atskyrimo metode.

2.5.6. Mozilla

„Mozilla“ tarptautinės bendrijos kuriamose atvirosiose internetinėse programose (naršyklė, el. pašto programos, hiperteksto rengyklė, pokalbių programa, kalendorius ir nedidelės programos-priedai, kurie papildo pagrindines programas tam tikromis funkcijomis) naudojamas tas pats „Mozilla“ išteklių atskyrimo metodą ir pateikimo formatus. Šiame darbe tikslinga apžvelgti šį metodą, kadangi internetinių „Mozilla“ programų skaičius, jų lokalizacijų ir naudotojų skaičius nuolat didėja.

Tekstiniai lokalizuojamieji ištekliai yra iškeliami į dviejų pagrindinių tipų grynojo teksto failus: DTD ir PROPERTIES formato.

DTD failuose pateikiamos statinės išteklių eilutės, t. y. tokios eilutės, kurios panaudotos grafinės naudotojo sąsajos aprašo faile: pastovios meniu komandos, dialogo langų elementų – mygtukų, teksto užrašų, žymimųjų langelių ir kt. pavadinimai. DTD failas koduojamas UTF-8 koduote ir turi daugeliui lokalizuojamųjų išteklių atskyrimo metodų būdingą eilučių vardų ir tekstų porų struktūrą, tik apsupta XML gairėmis (12 pav.). Tokiame faile galima pateikti ir komentarus.

```
<!ENTITY  colorsDialog.title              "Spalvos">
<!ENTITY window.width                     "42em">

<!ENTITY  color                           "Tekstas ir fonas">
<!ENTITY  textColor.label                 "Tekstas:">
<!ENTITY  textColor.accesskey             "T">
<!ENTITY  backgroundColor.label           "Fonas:">
<!ENTITY  backgroundColor.accesskey       "F">
<!ENTITY  useSystemColors.label           "Spalvas imti iš operacinės sistemos">
<!ENTITY  useSystemColors.accesskey       "s">

<!ENTITY  underlineLinks.label            "Saitus pabraukti">
<!ENTITY  underlineLinks.accesskey        "b">
<!ENTITY  links                           "Saitų spalvos">
<!ENTITY  linkColor.label                 "nelankytų:">
<!ENTITY  linkColor.accesskey             "n">
<!ENTITY  visitedLinkColor.label          "aplankytų:">
<!ENTITY  visitedLinkColor.accesskey      "a">
```

12 pav. DTD failo fragmentas („Mozilla Firefox“ naršyklės lietuviška lokalizacija)

PROPERTIES failai niekuo nesiskiria nuo aukščiau aprašyto Javos išteklių rinkinių PROPERTIES failo formato. Tik „Mozilla“ programose laikomasi susitarimo: PROPERTIES failuose pateikiamos dinaminės naudotojo sąsajos eilutės, kuriomis operuoja *JavaScript* scenarijai: programos pranešimai, klaidų pranešimai, sąrašų elementai, besikeičiantys valdiklių pavadinimai. Koduojamas toks failas UTF-8 koduote (anksčiau, kaip ir Javos ištekliu atveju buvo naudojamos Unikodo kaitos ženklų sekos vietoje ne ASCII ženklų).

„Mozilla“ metodas numato ne tik tekstinių išteklių atskyrimą, bet ir dvejetainių objektų, grafinio apipavidalinimo stilių, nuostatų failų, RDF failų ir pan. Visi šie nuo lokalės priklausomi ištekliai skirstomi į katalogus ir pakuojami į nepriklausomus nuo platformos XPI paketus, skirtus platinti.

2.5.7. XLIFF

Tai XML pagrindu sukurtas lokalizavimo mainų failų formatas (angl. *XML Localization Interchange File Format*), skirtas lokalizavimo duomenų mainams tarp programinės įrangos kūrėjų ir lokalizuotojų arba tarp įvairių lokalizavimo (dalinio) automatizavimo programinių priemonių.

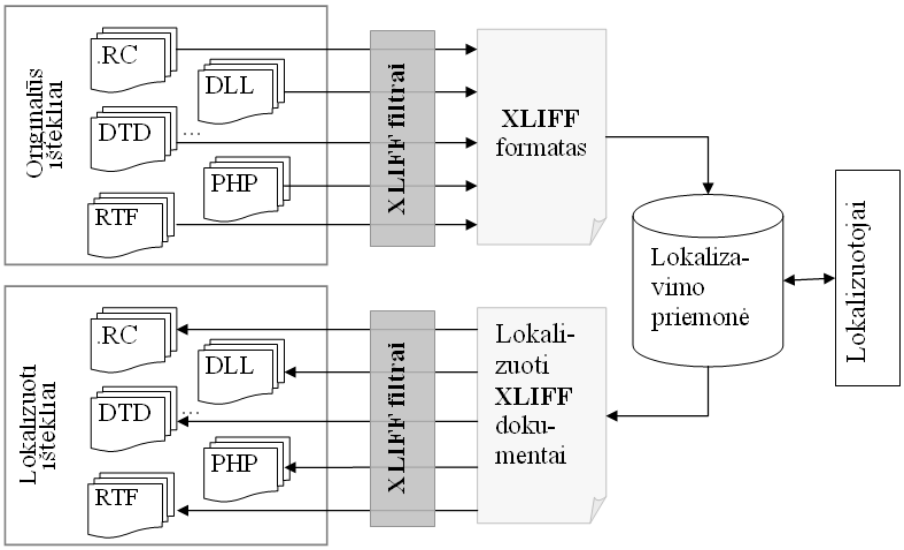
Šis formatas yra skirtas programinės įrangos naudotojo sąsajos lokalizuojamiems ištekliams atskirti, taip pat ir rišliam tekstui (pvz., interneto svetainių, dokumentacijos, įprastų tekstinių dokumentų) versti.

Pagrindinės XLIFF formato funkcijos:

- Atskirti lokalizuotiną arba verstiną tekstą nuo jo formatavimo informacijos.
- Pritaikyti darbui su įvairiomis programinėmis lokalizavimo ir vertimo priemonėms.
- Fiksuoti ir kaupti naudingą informaciją apie lokalizavimo procesą.

Kuriant XLIFF standartą buvo turimas omenyje pramoninis lokalizavimo procesas, kai vertėjai nebūtinai turi pakankamai techninių darbo su įvairiais formatais žinių, lokalizavimas pavedamas kelioms įvairioms kompanijoms, kad tas pats formatas apimtų ir rišlaus teksto dokumentų vertimus, kad būtų galima mainytis duomenimis apie lokalizavimo proceso eigą ir užbaigtumą.

XLIFF šiuo metu naudojamas tik kaip tarpinis formatas: programos lokalizuojamieji ištekliai yra atskiriami naudojant vieną iš aukščiau aprašytų metodų ir pateikiami tą metodą atitinkančiais formatais (pvz., PO, RC, RESX, EXE, DLL, PROPERTIES, DTD ir kt.), tada jie yra konvertuojami į XLIFF formatą naudojant specialias priemones – XLIFF filtrus, tada XLIFF failai pateikiami lokalizuotojams, kurie lokalizuoja (verčia) šį failą pasitelkę specializuotas lokalizavimo priemones, o lokalizavus XLIFF ištekliai konvertuojami atgal į originalų formatą naudojant atgalinius filtrus (13 pav., paveikslas sudarytas nagrinėjant XLIFF temą).



13 pav. XLIFF formato naudojimas lokalizavimo procese

XLIFF specifikacijoje daug dėmesio skiriama verstino ar lokalizuojamo failo formatavimo informacijai, kuri gali būti pateikiama HTML ar XML gairėmis, RTF ar WORD formatavimo direktyvomis ir pan. Tai labiau tinka rišlių tekstų vertimų darbams (programinės įrangos lokalizavimo gali būti taikoma elektroniniams žinynams lokalizuoti).

Kai pradinių lokalizuojamųjų išteklių failas konvertuojamas į XLIFF formatą, struktūrinis formatavimas (pvz., kaip gairės <body> HTML dokumente) yra ištraukiamas ir išsaugomas karkaso (angl. *skeleton*) faile. Šiame pagalbiniame faile nurodoma, kur vertimo vienetas (angl. *unit*) turės būti patalpintas, konvertuojant iš XLIFF atgal į originalų formatą. Struktūrinio formatavimo elementai (gairės) nėra pasiekiami lokalizuotojams (vertėjams) ir lokalizavimo priemonių kūrėjams, jie dirba tik su tekstu ir XLIFF apibrėžtais elementais. Tokiu būdu apsisaugoma nuo dokumento pirminio teksto struktūros ir formatavimo informacijos pažeidimo.

Įterptinis formatavimas valdomas dviem būdais: 1) ištraukiamas į karkaso failą, o vietoj originalios formato gairės įterpiama viena iš dviejų XLIFF specifikacijoje apibrėžtų gairių (*g* gairę poriniams formatavimo elementams, pvz., elementams , o *x* gairė – neporiniams, pvz., <img>), arba 2) paliekant formatavimą XLIFF faile, tik įdedant XLIFF

gaires formatavimo pradžiai ir pabaigai pažymėti (pvz., <it id="a1" pos="1"><img src="pav.gif"></it>).

XLIFF turi priemonių lokalizavimo procesui žymėti. Specialūs lokalizavimo etapų elementai turi atributus, skirtus informacijai apie lokalizavimo laikotarpį, naudotą priemonę, atsakingą asmenį, lokalizavimo etapo pavadinimo (pvz., vertimas, recenzavimas), viso lokalizavimo proceso pavadinimas. Pastabų elementai skirti lokalizavimo etapui pakomentuoti laisva forma. Kiekvieno elemento vertimas gali turėti lokalizavimo etapo pavadinimo atributą, kuris leidžia matyti, kurio etapo metu buvo atliktas tam tikras vertimas (tai gali būti svarbu recenzavimui, peržiūrai).

Pagrindinis XLIFF failo elementas yra <trans-unit> (vertimo vienetas), kuris turi įdėtinius elementus <source> ir <target>. Failas yra dvikalbis: <source> elemente laikoma originali eilutė arba jos segmentas (kai eilutė yra ilga, pvz., rišlaus teksto dokumente), o <target> – eilutės arba jos segmento vertimas. Kiekvienam originaliam lokalizuotinam elementui galima priskirti keletą vertimo (lokalizavimo) variantų, pvz., neaiškų vertimą, vertimo atminties priemonės įterptą atitikmenį, automatinio vertimo atitikmenį, iš kurių vertėjas gali suformuoti galutinį to elemento vertimą.

Kontekstui nurodyti XLIFF turi elementą <context>, kuriuo galima apibūdinti pradinio teksto <source> arba alternatyvaus vertimo kontekstą. Šio elemento įvedimo tikslas – leisti tam tikroms teksto dalims turėti skirtingus vertimus, priklausomai nuo to, iš kur jie ateina, pvz., skirtingai išversti tą patį tekstą, kuris naudojamas žiniatinklio formoje, dialogo lange, duomenų bazės formoje ir pan. Konteksto elementu pažymėta informacija yra skirta vertėjui arba lokalizavimo priemonei, jei ji suprojektuota taip, kad galėtų ji interpretuoti ir automatiškai pasiūlyti tinkamą vertimą.

Tos pačios rūšies lokalizuojamiems ištekliams grupuoti yra numatytas elementas <group>.

Prie teksto fragmento gali būti priskirtas atributas, nurodantis valdiklio tipą, kuriame bus pavartota eilutė (pvz., mygtukas, žymimoji akutė, meniu elementas ir pan.).

Į XLIFF failą galima sudėti ne tik tekstinius lokalizuojamuosius išteklius, yra numatytas ir dvejetainiams objektams žymėti skirtas elementas <bin-uint> (objektas įterpiamas tiesiogiai, panašiai, kaip .RESX formate).

Konvertuojant lokalizuotą XLIFF failą atgal į originalų formatą, jis yra suliejamas su karkaso failu ir pateikiamas pageidaujamu formatu. Konvertavimo filtras skaito žymes ir įdeda visus vertimus, kurie turi patvirtintų statusą. Jei nėra patvirtinto vertimo, įdedamas originalus tekstas.

XLIFF formatas turi privalumų palyginus su aukščiau nagrinėtais formatais, kadangi tai vieningas lokalizavimo išteklių mainų formatas, leidžia pateikti daugiau išteklių ir lokalizavimo proceso metainformacijos. Tačiau kadangi šiuo metu jis yra naudojamas tik kaip tarpinis lokalizavimo formatas, tai realybėje metainformacija beveik nenaudojama (nes konvertuojama automatiškai iš esamų formatų, kuriuose šios informacijos nėra). Darbui su XLIFF skirtos priemonės taip pat šiuo metu realizuoja tik nedaugelį iš specifikacijoje siūlomų elementų. Be to, formato vieningumo realybė yra sąlyginė, kadangi atsiranda tarpinės priemonės-filtrai, kurie konvertuoja išteklius iš esamų formatų. Daugumą lokalizuotų XLIFF formatu failų po atgalinio konvertavimo tenka taisyti (išdėstymas ir formatas gali būti šiek tiek ne toks, koks originalaus). Lengviausiai konvertuojami ir nereikalauja taisymų XML, HTML, ir RTF formatai.

XLIFF oficiali specifikacija yra įteisinta kaip OASIS standartas [Oas08].

2.5.8. Kiti lokalizuojamųjų išteklių pateikimo formatai

Aukščiau apžvelgėme populiariausius lokalizuojamųjų išteklių atskyrimo metodus ir atitinkamus formatus. Yra ir kitų lokalizuojamųjų išteklių pateikimo failų formatų, tačiau jie visi turi vardų ir reikšmių porų sąrašo struktūrą, skirtumas yra tik tas, kad vardai ir reikšmės skiriamos naudojant skirtingus simbolius.

Galima pastebėti kalbinių išteklių, pateikiamų .LNG arba .LANG formatu (naršyklė „Opera“, failų ir katalogų tvarkymo programa „Total Commander“, pokalbių programa „Skype“ ir kt.), CSV formato failų, kuriuose eilutės vardas nuo eilutės teksto skiriami kableliais, TAB formato failus, kuriuose eilutės vardas nuo eilutės teksto skiriami tabuliacijos ženklais.

Lokalizuojant internete veikiančias sistemas tenka pastebėti, kad daugelis jų naudoja PHP formatą lokalizuojamiems ištekliams pateikti (pvz., „Moodle“ virtualioji mokymosi aplinka, „MediaWiki“ vikio sistema ir kt.). Tai paprasčiausias tekstinis failas, į kurį yra įrašytas PHP eilučių masyvas, taigi taip pat turintis vardų ir reikšmių porų struktūrą (14 pav.).

```
$string['activity'] = 'Activity';
$string['activityweighted'] = 'Activity per user';
$string['activityclipboard'] = 'Moving this activity: <b>$a</b>';
$string['activityiscurrentlyhidden'] = 'Sorry, this activity is currently
hidden';
$string['activitymodule'] = 'Activity module';
$string['activityreport'] = 'Activity report';
$string['activityreports'] = 'Activity reports';
$string['activityselect'] = 'Select this activity to be moved elsewhere';
$string['activitysince'] = 'Activity since $a';
$string['add'] = 'Add';
$string['addactivity'] = 'Add an activity...';
$string['addadmin'] = 'Add admin';
$string['addcreator'] = 'Add course creator';
$string['added'] = 'Added $a';
$string['addedrecip'] = 'Added $a new recipient';
$string['addedrecips'] = 'Added $a new recipients';
$string['addedtogroup'] = 'Added to group $a';
$string['addedtogroupnot'] = 'Not added to group $a';
```

14 pav. PHP tekstinių lokalizuojamųjų išteklių pateikimo formatas

PHP funkcija *get_string* (*'eilutės vardas'*, *'failo vardas'*) naudojama atitinkamai teksto eilutei iš atskirto PHP failo gauti.

Reikia pastebėti, kad programos gali naudoti vieną lokalizuojamųjų išteklių atskyrimo metodą, tačiau lokalizuotojams platinti tekstinius išteklius supaprastintu tekstiniu formatu, tokiu būdu kartais susiaurindami ir šiaip negausią metainformacija.

Pastebėsime, kad RC ir RESX metodai rečiau naudojami internetinėje programinėje įrangoje (8 lentelė).

8 lentelė. Internetinės programos ir jų lokalizuojamųjų išteklių formatai

Internetinė programinė įranga	Išteklių pateikimo formatas
Drupal	Gettext PO
Internet Explorer	.RC, .RESX
Lemill	Gettext PO
MediaWiki	PHP
Moodle	PHP
Mozilla Firefox	Mozilla DTD, PROPERTIES
Mozilla Thunderbird	Mozilla DTD, PROPERTIES
NVU	Mozilla DTD, PROPERTIES
Opera	Lng

Internetinė programinė įranga	Išteklių pateikimo formatas
Outlook Express	.RC, .RESX
Pidgin	Gettext PO
Plone	Gettext PO
SeaMonkey	Mozilla DTD, PROPERTIES
Skype	Lang

2.6. Lokalizavimo dėsniai

O'Sullivan [Sul01] suformulavo programinės įrangos lokalizavimo šešis dėsnius. Jis teigia, kad šie dėsniai turi būti taikomi programinei įrangai, turinčiai tą pačią pirminių programos tekstų struktūrą ir originalo versijoje, ir lokalizuotose versijose, bei nenumatančioje funkcinio programos adaptavimo (tų elementų, kurių nėra lokalių modeliuose).

1 dėsnis. Pirminis programos tekstas ir naudotojo sąsaja turi būti visiškai atskirti vienas nuo kito. Pavyzdžiui, operacinės sistemos „Microsoft Windows" programose naudotojo sąsajos tekstai yra pateikiami programos vykdomojo failo duomenų sekcijoje, tai laikoma, kad šis dėsnis pažeidžiamas.

2 dėsnis. Naudotojo sąsajos eilučių valdymas programiniu būdu turi būti ribojamas. Programos funkcijos neturi priklausyti nuo lokalizuojamųjų tekstų reikšmių, kitaip gali įvykti klaida. Pavyzdžiui, jei programa lygina dvi sąsajos eilutes ir atsižvelgiant į palyginimo rezultatą, atlieka tam tikrą funkciją, tai lokalizuotoje versijoje ta pati funkcija gali būti atliekama nekorektiškai, nes dviejų eilučių (net ir vienodų originale) vertimas ne visada būna vienodas. Taip pat reikėtų vengi eilučių dalių manipuliavimo iš programų pirminių tekstų, ribojimų teksto ilgiui ir pan.

3 dėsnis. Pirminis programos tekstas ir sąsaja turi būti kompiliuojami atskirai. Naudotojo sąsaja turėtų būti pateikiama atskirame programos modulyje. Tada visi pakeitimai, padaryti naudotojo sąsajos viduje, neturėtų poveikio pirminiam programos tekstui, nereikėtų papildomo susaistymo arba perkompiliavimo. Ir atvirkščiai, pirminio teksto pakeitimai nereikalautų grafinės sąsajos keitimo.

4 dėsnis. Naudotojo sąsajos visi tekstai turi būti pateikiami su jų naudojimo kontekstu. Kiekviena eilutė turi būti laikoma ištekliuose ir manipuliuojama su visu jos naudojimo naudotojo sąsajoje kontekstu. Deja, čia autorius turėjo omenyje tik eilučių identifikatorius, siejant juos su naudotojo sąsajos elementų (pvz., meniu) identifikatoriais tam, kad būtų paprasčiau atnaujinti lokalizuojamuosius išteklius išėjus naujai programos versijai.

5 dėsnis. Eilučių vertimas ir tolesnis atnaujinimas turi būti atliekamas „ką matau, tą gaunu" aplinkoje: išvertus išteklių eilutes, be specialaus perkompiliavimo lokalizuotojai turėtų matyti lokalizavimo rezultatą.

6 dėsnis. Naudotojo grafinės sąsajos komponentų dydžių keitimas ir išdėstymas turi būti vykdomas automatiškai programai veikiant. Taip programos naudotojo sąsaja prisitaiko prie lokalizacijoje naudojamo rašymo būdo (iš kairės į dešinę, iš dešinės į kairę, iš viršaus į apačią) ir vertimų sutalpinimo grafinės sąsajos elementuose.

Minėti dėsniai buvo suformuluoti 2001 metais. Jie yra labai bendri ir galima pastebėti, kad jų nepakanka siekiant tam tikros programos visiško lokalizavimo, trūksta įvairių lokalių elementų numatymo.

1 dėsnis yra natūralus reikalavimas programinei įrangai, kurią ruošiamasi lokalizuoti. Tačiau iki šiol galima rasti daugeliui kalbų lokalizuojamų programų, kuriose šis dėsnis pažeidžiamas arba pažeidžiamas iš dalies: pamirštama atskirti dalį grafinėje sąsajoje vaizduojamų tekstų nuo programos pirminio teksto [LD03, Mmv09b].

2, 4, 5, 6 dėsniai nėra visuotinai realizuoti ir dabar. Tai rodo įvairių programų analizė lokalizavimo požiūriu ir lokalizavimo ekspertizės [Mmv09b].

Pastebėsime, kad 4 dėsnį galima praplėsti iki konteksto, teikiamo ne tik eilučių ir sąsajos identifikatoriais, bet ir teksto semantikos, grafinių elementų, greta esančių elementų. Tokio konteksto žinojimas padėtų lokalizuotojui tiksliau išversti tam tikrą lokalizuojamą eilutę ir išvengti lokalizavimo klaidų.

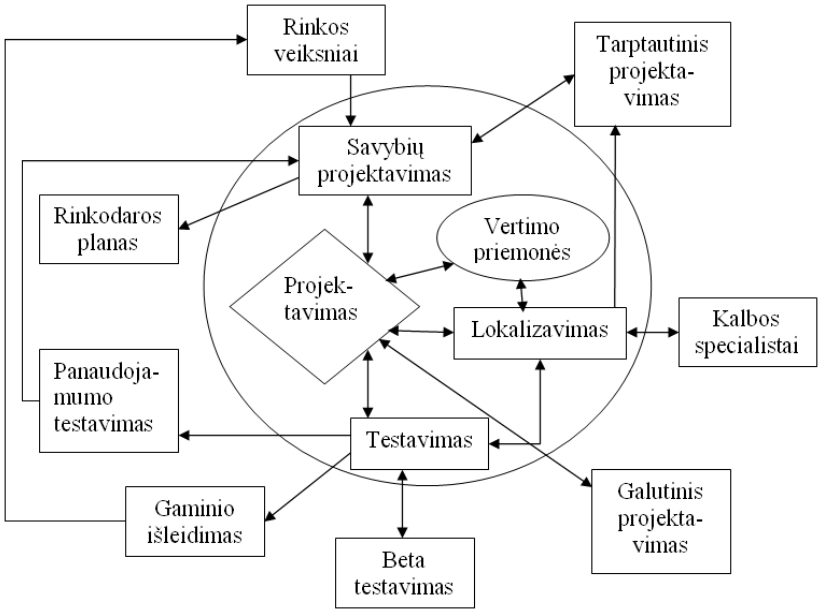
2.7. Išvados

1. Viena iš lokalizavimo problemų priežasčių yra ta, kad internacionalizuojant programinę įrangą naudojamasi esamais lokalių modeliais. Pagrindinių lokalių modelių analizė parodė, kad kiekvienas istoriškai vėliau atsirandantis lokalės modelis įtraukia daugiau lokalės elementų ir būna skirtas pagerinti ankstesnius žinomus modelius, tačiau:
   1.1. Skirtingi lokalių modeliai nėra tarpusavyje suderinami (paprastai išlaikomas tik naujo modelio suderinamumas su POSIX lokale, kurios kategorijų nepakanka norint tinkamai lokalizuoti programinę įrangą).
   1.2. Nei vienas lokalės modelis neapima pakankamai lokalių kultūrinių normų (pvz., giminių, linksnių, mažųjų ir didžiųjų raidžių derinimas ir frazių komponavimas), dėl to reikia ieškoti kitų būdų problemoms, susijusioms su minėtomis kultūrinėmis normomis spręsti.
2. Net pakankamai gerai internacionalizuotoje programinėje įrangoje pasitaiko tokių elementų, kuriuos galima aptikti ir pritaikyti tik lokalizavimo metu, pvz., grafika, naršymo schemos, grafinių elementų išdėstymas. Šias problemas galima spręsti pasitelkus esamus kultūrinių dimensijų modelius.
3. Apžvelgus pagrindinius egzistuojančius lokalizuojamųjų išteklių pateikimo formatus bei tokių išteklių atskyrimo nuo programinės įrangos pirminių tekstų metodus, galima padaryti išvadas, kad:
   3.1. Nepaisant to, kad egzistuoja keli pagrindiniai lokalizuojamųjų išteklių atskyrimo metodai ir nemažai formatų, dauguma internetinių programų lokalizuojamiems ištekliams pateikti naudoja tą patį lokalizuojamų eilučių pateikimo principą: tekstinis failas, kuriame pateikiamas eilučių vardų ir eilučių tekstų porų sąrašas. Papildomai tokiame faile yra numatomas tik komentarų rašymas. Išimtis – RC, RESX formatų dialogo sekcijos ir XLIFF formatas.
   3.2. .RC ir .RESX formatai yra rečiau naudojami internetinėje programinėje įrangoje, juos teko pastebėti tik „Microsoft“ klientinėse programose.
   3.3. Iš visų formatų GNU „gettext“ vienintelis numato daiktavardžių, vartojamų kartu su skaitvardžiu, formų derinimo mechanizmą.
   3.4. XLIFF turi privalumų, palyginus su kitais formatais, tačiau specifikacijoje numatytos galimybės nėra realizuotos dėl to, kad vyksta konvertavimas iš esamų formatų, kurie neturi metainformacijos pateikimo priemonių, t. y. XLIFF faktiškai paveldi ir kitų formatų trūkumus. Be to, XLIFF yra skirtas ne tik programinės įrangos sąsajos eilutėms lokalizuoti, bet pritaikytas ir rišliam tekstui, turinčiam kontekstinę informaciją, versti (bet kuriems dokumentams, kurie nebūtinai yra aiškiai struktūrizuoti, kaip grafinės naudotojo sąsajos eilutės), dėl to jame atsiranda daug papildomos formatavimo informacijos, kas nėra aktualu programų grafinės sąsajos tekstiniams ištekliams lokalizuoti.
   3.5. Būtina ieškoti būdų, kaip efektyviau pateikti lokalizuojamuosius išteklius, įtraukiant lokalizavimui svarbią kontekstinę informaciją.

3. LOKALIZAVIMO PROCESO TYRIMAS

Šiame skyriuje nagrinėjama lokalizavimo darbų struktūra ir jų eiga – lokalizavimo procesas, išskiriami pagrindiniai internetinės programinės įrangos kultūros elementai. Nagrinėjant lokalizavimo proceso etapus ir klasifikuojant kultūros elementus apibendrinama internetinės programinės įrangos lokalizavimo patirtis, susiję moksliniai šaltiniai ir standartai.

Kano [Kan95] pateikia lokalizavimo proceso modelį remdamasis „Microsoft“ bendrovės lokalizavimo patirtimi (15 pav.). Į modelį įtrauktas programinės įrangos projektavimo procesas bei nemažai rinkos veiksnių, kurių mes šiame darbe nenagrinėsime. Tačiau jis padeda pamatyti projektavimo ir lokalizavimo procesą rinkos kontekste. Šiame modelyje laikoma, kad programą projektuoja ir lokalizuoja ta pati bendrovė. Toliau nagrinėsime lokalizavimo procesą, minėtoje schemoje atitinkantį lokalizavimo, testavimo ir iš dalies projektavimo komponentus, iš lokalizuotojų (kurie nebūtinai priklauso kūrėjų bendrovei) pozicijos, išskyrę jo svarbiausius aspektus.



15 pav. Lokalizavimo procesas projektavimo ir rinkos kontekste. Adaptuota remiantis [Kan95]

Vienareikšmiškai nuspręsti, kuri iš 2.1 skyriuje minėtų dviejų lokalizavimo darbų dalių – programos adaptavimas lokalės normoms ar dialogo tekstų adaptavimas ir vertimas – kelia daugiau problemų, būtų sunku. Pirmoji darbų dalis – programos adaptavimas – atliekama remiantis lokalių formaliaisiais aprašais (2.3 sk.). Antroji – dialogo tekstų vertimas ir adaptavimas – susijusi su didelio kiekio tekstų vertimu ir adaptavimu, jo atnaujinimu, susiduriama su kalbos semantiniais aspektais, kurie nėra formalizuoti nei viename iš egzistuojančių lokalių modelių.

Abiejų lokalizavimo dalių darbai gali būti atliekami lygiagrečiai, tačiau prieš pradedant lokalizuoti svarbu atlikti parengiamuosius darbus. Yang [Ya07] ir kt. autoriai akcentuoja lokalizuotojų, naudotojų tyrimo grupės ir programuotojų bendradarbiavimą. Net jeigu internetinė programinė įranga buvo projektuojama turint omenyje būsimas lokalizacijas (t. y. turi aukštą internacionalizacijos lygį), vis tiek atsiranda problemų, kurias tenka spręsti

lokalizavimo metu [Co02]. Tai rodo, kad yra neišspręstų internacionalizavimo problemų ir reikia gilesnės jų analizės.

Iš 15 paveikslo matyti, kad lokalizavimo darbai yra glaudžiai susiję su testavimu, bendradarbiavimu su kalbos specialistais. Jei programinę įrangą lokalizuoja ne jos kūrėjai, tai ypač svarbus lokalizavimo parengiamųjų darbų etapas.

3.1. Lokalizavimo parengiamieji darbai

Lokalizavimo parengiamųjų darbų tikslas – lokalizavimui parinkti labiausiai reikalingas ir pačias geriausias programas, parengti lokalizavimo planą tokį, kad lokalizuotos programos kokybė būtų kuo aukštesnė. Išskirsime pagrindinius veiksmus, kurie atliekami prieš pradedant realius lokalizavimo darbus.

1. **Būsimų naudotojų skaičiaus prognozė**. Prieš pradedant lokalizuoti programą reikia ištirti jos būsimųjų naudotojų skaičių. Laikoma, kad programos lietuvinimą verta nagrinėti, jei būsimų naudotojų skaičius didesnis negu tūkstantis [GZ00].
2. **Susipažinimas su programos licencija**. Lokalizavimas yra vienas iš programinės įrangos modifikavimo būdų. Programos licencija – tai teisinis dokumentas, kurio reikia griežtai laikytis. Nuosavybines (ypač komercines) programas paprastai galima modifikuoti, taigi ir lokalizuoti, tik autoriui leidus. Kartais lokalizuoja patys autoriai. Nuo licencijos priklauso ir problemų, aptiktų lokalizavimo metu, sprendimo būdas. Jei neleidžiama programą laisvai modifikuoti, aptiktų problemų šalinimas priklausys tik nuo autoriaus veiksmų. Esant leidimui modifikuoti pirminius programos tekstus, dalį klaidų (problemų) gali pašalinti patys lokalizuotojai.
3. **Susipažinimas su esama informacija apie autorių ir jo darbus**. Iš autoriaus darbų galima spręsti apie jo kvalifikaciją, aktyvumą ir pasirengimą pataisyti programos internacionalizaciją, jei lokalizavimo metu bus aptiktos internacionalizavimo klaidos. Autoriaus pasirengimas aktyviai reaguoti į lokalizuotojų pastebėtas programos internacionalizavimo klaidas yra ypač svarbus, jei programos licencija neleidžia laisvai modifikuoti programos pirminius tekstus.
4. **Susipažinimas su programa**. Programinės įrangos lokalizuotojai privalo gerai žinoti lokalizuojamos programos funkcijas, joje vartojamą terminiją. Susipažinti su programos funkcinėmis savybėmis galima tiesiogiai, taip pat iš recenzijų ir atsiliepimų.
5. **Lokalizavimo galimybių analizė**. Programos išteklių tipai, lokalizuojamųjų išteklių tipai ir jų analizė, ar reikalingas programos perkompiliavimas ir pan. Plačiau lokalizuojamųjų išteklių tipai nagrinėjami 2.5 sk.
6. **Savo galimybių įvertinimas**. Prieš pradedant lokalizuoti reikėtų įvertinti, ar pakaks jėgų, ištvermės ir išteklių darbui baigti iki galo, nepamiršti, kad atsiras naujų programos versijų ir reikės operatyviai jas lokalizuoti, kad lokalizuotos programos naudotojams kils klausimų ir juos reikės konsultuoti.
7. **Kontakto užmezgimas su autoriumi**. Naudingas net ir tuo atveju, kai programos išteklius galima laisvai modifikuoti. Atvirųjų programų lokalizavimo atveju reikėtų užsiregistruoti tarptautinėje lokalizuotojų bendruomenėje.
8. **Internacionalizacijos analizė**. Nuo programos internacionalizacijos laipsnio priklausys lokalizavimo sudėtingumas. Programą lengviau lokalizuoti, jeigu ją projektuojant atsižvelgiama į būsimų lokalizacijų poreikius: laikomasi tarptautinių standartų, atskiriamos ištekliais vadinamos programos dalys, kurios turi būti keičiamos lokalizuojant (ekrane matomi tekstai, laukų matmenys, koduotės, matavimo vienetai, datų formatai ir pan.) nuo tų, kurios lokalizuojant neturėtų būti keičiamos (vykdomieji

moduliai ar jų dalys). Tai didžiausia lokalizavimo parengiamųjų darbų dalis. Analizuojant programą internacionalizavimo požiūriu remiamasi klausimynais [Mmv09a].

Jeigu tikrinama programa jau išversta į kurią nors kitą kalbą, tai ja galima pasinaudoti ieškant klaidų. Iš tikrųjų bus ieškoma pasirinktos kalbos lokalizacijos klaidų. Radus tokią klaidą dar reikia patikrinti, ar jos priežastis yra internacionalizavimo klaida (klaidą galėjo padaryti ir kitos kalbos lokalizuotojas).

Klaidas, susijusias su ženklais ir jų koduotėmis, geriau galima pastebėti nagrinėjant tautų, vartojančių ne Vakarų Europos koduotes (*ISO/IEC 8859-1* arba *Windows-1252*), kalbas, nes yra programų, į kurias įkoduota Vakarų Europos koduotė, dėl to jos netinka kitoms kalboms, išskyrus vartojančias tą koduotę.

Praktiškai dirbant su lokalizuota programa galima pastebėti neišverstus tekstus. Radus tokią klaidą reikia nustatyti, ar ta klaida yra lokalizavimo (pvz., pamiršta išversti eilutė) ar internacionalizavimo (eilutės nėra lokalizuojamuose ištekliuose). Reikia atkreipti dėmesį į tai, kad kai kurios programos dalį valdiklių (mygtukų, dialogų langų) su jų tekstais ima iš operacinės sistemos išteklių. Jeigu operacinės sistemos kalba nesutampa su programos kalba, tai tada lokalizuotoje programoje tie tekstai bus matomi kita kalba. Tai nėra internacionalizavimo klaida, nes laikoma, kad programinės įrangos lokalizavimą natūralu pradėti nuo operacinės sistemos. Yra ir tokių programų, kurios turi savus langus, bet jų nuostatose galima pasirinkti, kuriuos naudoti: savus ar operacinės sistemos. Ekrane matomas, bet į išteklius neįdėtas eilutes galima rasti panaudojus išteklių pseudovertimą.

Remiantis aukščiau aprašytais veiksmais galima daryti išvadą, kad į parengiamuosius darbus reikia kreipti deramą dėmesį. Tam reikia nemažų sąnaudų, kurį laiką gali būti pristabdytas realus lokalizavimas, tačiau pasirinkus gerą programą ir tinkamai pasiruošus jos lokalizavimui padidės tikimybė, kad darbas bus padarytas iki galo ir jo rezultatais bus sėkmingai ir ilgai naudojamasi.

3.2. Programos adaptavimas

Programos adaptavimo lokalės normoms – tai realių lokalizavimo darbų dalis, nuo kurios galima pradėti arba atlikti lygiagrečiai su dialogo tekstų vertimu ir adaptavimu. Programos adaptavimo lokalės normoms darbams atlikti reikia standartizacijos ir programavimo žinių. Darbo apimtis priklauso nuo originalios programos internacionalizavo lygio, t. y. galima remtis rezultatais, gautais parengiamųjų darbų internacionalizacijos analizės etape.

Pagrindiniai adaptuotino lygio kultūros elementai apžvelgti lokalių modelių skyrelyje (2.3 sk.), o internetinei programinei įrangai būdingi kultūros elementai analizuojami 3.7 skyrelyje. Kiti, mažiau akivaizdūs, kultūros elementai aptarti kultūrinių dimensijų skyrelyje (2.4 sk.).

3.3. Dialogo tekstų vertimas ir adaptavimas

Daugelyje lokalizavimo bendrovių ir atvirųjų projektų dialogo tekstų vertimo ir adaptavimo išbaigtumas matuojamas išverstų teksto eilučių procentu. Tačiau toks matavimas neatspindi tikrovės. Susidūrę su lokalizavimu žino, kad paskutinės paliktos neišverstos (neadaptuotos) lokalizuotinos eilutės yra sunkiau suprantamos, neaptiktos programos grafinėje sąsajoje, turinčios kalboje dar neegzistuojančių terminų ir reikalauja daugiau darbo sąnaudų norint jas tinkamai lokalizuoti, nes pradžioje lokalizuotojas dažniausiai verčia tuos tekstus, kurie yra jam aiškūs [Ess00].

Tyrime [DG06] pasiūlytas modelis, kuriuo matuojamas lokalizavimo užbaigtumas pagal išverstų eilučių procentą. 16 paveiksle pateikta išverstų eilučių procento ir padaryto lokalizavimo darbo priklausomybė. Išverstų eilučių procentai nurodyti kairiajame stulpelyje ir viršutinėje eilutėje. Pagal šį modelį, pavyzdžiui, jei konstatuojama, kad yra išversta 90% programos sąsajos eilučių, tai reiškia, kad padaryta tik 42% lokalizavimo darbo, jei išversta 96% programos sąsajos eilučių, tai padaryta 58% lokalizavimo darbo.

	0	1	2	3	4	5	6	7	8	9
00	0	0	0	0	0	0	0	0	0	0
10	0	1	1	1	1	1	1	1	1	1
20	1	2	2	2	2	2	2	2	3	3
30	3	3	3	3	4	4	4	4	4	5
40	5	5	5	6	6	6	6	7	7	7
50	8	8	8	9	9	9	10	10	10	11
60	11	12	12	13	13	14	14	15	16	17
70	18	18	19	20	21	21	22	23	24	25
80	26	27	28	29	31	32	34	35	37	39
90	42	43	46	49	51	54	58	62	68	78

16 pav. Programos išverstų eilučių procento ir atlikto lokalizavimo darbo procento santykis [DG06]

Dialogo tekstų vertimas ir adaptavimas yra ilgiausias lokalizavimo darbų etapas, kurį suskirstyti į smulkesnes dalis (etapus):

1. Juodraštinis dialogo frazių vertimas.
2. Frazių derinimas.
3. Žinyno vertimas ir testavimas.
4. Visų tekstų redagavimas.
5. Kompleksinis testavimas.

3.3.1. Juodraštinis frazių vertimas

Dialogo tekstų vertimas ir adaptavimas pradedamas nuo tekstų, įdėtų į lokalizuojamųjų išteklių failus, pirminio vertimo (plačiau apie lokalizuojamųjų išteklių struktūrą rašoma 4 skyriuje).

Sukurta nemažai priemonių [Was04], kuriomis galima versti įvairių formatų išteklių failus bei išteklius, įdėtus į koduotus failus (DLL, EXE). Kai kurios jų gali automatizuoti ir patį vertimą, paimdamos frazes iš frazių žodyno (vertimo atminties) ir įdėdamos į išteklių failus. Tada rankiniu būdu beliktų išversti tik tas frazes, kurių nėra žodyne. Idėja viliojanti, kadangi frazės trumpos, lakoniškos, autonomiškos. Bet nemaža jų dalis priklauso nuo konteksto. Deja, kontekstas yra nematomas dirbant su eilutėmis. Dėl to atsiranda nevienareikšmiškumų. Taigi automatizuotą vertimą galima laikyti juodraštiniu, kurį reikia toliau tvarkyti žmogui.

Jei automatizavimo priemonėmis nesinaudojama, tai dėl tos pačios priežasties – konteksto trūkumo – eilučių vertimai taip pat yra tik pirminiai, bandomieji, kuriuos reikia testuoti ir tobulinti.

3.3.2. Frazių derinimas

Išversti išteklių failai įdedami į programą. Kai kurioms programoms reikalingas perkompiliavimas, kitos programos pačios susikompiliuoja tekstus arba juos interpretuoja.

Derinant išryškėja vertimo netikslumai, kurių daugumos nebuvo galima numatyti pirmajame etape, nes daugelis angliškų žodžių nėra verčiami vienareikšmiškai, daug kur reikia suderinti linksnius, pakeisti gramatines formas.

Programos sąsajos frazių vertimas skiriasi nuo įprasto rišlaus teksto (straipsnio, knygos ir pan.) vertimo tuo, kad programose vartojamos frazės yra lakoniškos ir atitrūkusios nuo programos veiksmų konteksto, į kurį galima patekti tik dirbant su programa. Todėl jas verčiant galima lengvai suklysti – pasirinkti ne tą reikšmę iš daugelio galimų, pavyzdžiui, *From ... To = Kas ... Kam* (el. laiške), *nuo ... iki* (kai kalbama apie laiką); *tab = tabuliacijos ženklas* (dokumente), *įselė* (dialogo lange); *title = antraštė* (dokumente), *titulas* (asmens); *check = pažymėti* (varnele), *tikrinti* (pvz., rašybą), *volume = tomas* (disko), *garso stiprumas* ir t. t.

Nemažai nevienareikšmiškumų atsiranda ir dėl to, kad daugelis tą pačią formą turinčių (vienodai rašomų) angliškų žodžių gali eiti daiktavardžio ir veiksmažodžio funkcijas, pvz., *file = failas, įdėti* (ką nors į aplanką); *display = monitorius* (kompiuterio įtaisas), *rodyti* (pvz., tam tikra koduote ir šriftu); *backup = archyvuoti, atsarginė kopija; list = sąrašas, išvardyti*.

Programose pasitaiko vietų, kur frazės gaunamos sujungiant dvi ar daugiau frazių (žodžių), vartojamų ir kitur: savarankiškai arba kituose frazių junginiuose. Daugelyje programų pasitaikantis pavyzdys, kai jungiamas komandų *Atšaukti* arba *Atstatyti* pavadinimas su pavadinimu tos komandos, kurios veiksmas atšaukiamas arba atstatomas, pavyzdžiui, *Įterpti*, *Iškirpti*. Mechaniškai jungiant gaunamos gramatiškai nesuderinamos žodžių poros: *Atšaukti Įterpti, Atstatyti Iškirpti*. Turėtų būti *Atšaukti įterpimą*, *Atstatyti iškirpimą*. Bet tie patys terminai *Įterpti* ir *Iškirpti* vartojami kaip atskiri žodžiai jais pavadintoms komandoms įvardyti. Iš padėties išsisukama įterpiant kokį nors skirtuką tarp žodžių poros, pavyzdžiui, *Atšaukti–Įterpti*, *Atšaukti>>Įterpti*. Bet tai tėra tik šioks toks klaidos užmaskavimas, bet ne ištaisymas.

Šią problemą turėtų išspręsti programų autoriai, internacionalizuodami programą. Panašios problemos iškyla ne tik lietuvių, bet ir kitose kalbose.

Verčiant ir adaptuojant programos tekstus, jie paprastai tampa ilgesni. Pavyzdžiui, ištirta, kad lokalizuojant anglišką programą vokiečių kalbai tekstai pailgėja 25%, ispanų, prancūzų, italų kalboms – iki 75%, portugalų – 35% [Ya07], hindi – 80% [Co02].

Išteklių koregavimo ir įdėjimo į programą procesas kartojamas, nes kiekvieną pataisymą reikia patikrinti. Dažnai tas pats angliškas žodis (frazė) išteklių failuose yra kartojamas kelis kartus. Tai reiškia, kad skirtingi to paties žodžio egzemplioriai patenka į skirtingus kontekstus, todėl gali turėti skirtingus lietuviškus atitikmenis ir reikia įsitikinti, ar pataisymas nuėjo ten, kur reikia. Blogiau, kai tas pats žodžio egzempliorius skirtinguose kontekstuose vartojamas skirtingomis prasmėmis. Tai gali būti laikoma internacionalizacijos klaida.

Derinimo metu parenkamos meniu ir dialogo languose esančių komandų užrašų raidės prieigos ir komandos klavišams taip, kad jos būtų patogios (lengvai įsimenamos) naudotojui ir nekonfliktuotų (tame pačiame meniu lygyje arba tame pačiame dialogo lange būtų visos skirtingos).

3.3.3. Žinyno vertimas ir dialogo vertimo testavimas

Žinyno vertimas panašus į įprastų tekstų vertimą, todėl paprastai rimtų lokalizavimo problemų nesukelia. Tačiau verčiant žinyną dažniausiai atliekamas ir programos testavimas – dirbama su veikiančia programa ir tikrinama, ar viskas, kas išversta yra teisinga. Tuo pat metu testuojami bei koreguojami ir dialogo frazių vertimai, kadangi verčiant ir nuodugniai nagrinėjant žinyną išryškėja programos taikymo ir veiksmų kombinacijos, kurios nebuvo numatytos antrame etape. Testavimo darbų dalis yra didesnė už patį vertimą. Galima teigti,

kad tik išvertus (arba parašius originalų) žinyną bus užbaigtas ir visas programos lokalizavimas [Sca02]. Be to, žinyno tekstą reikia adaptuoti (dažniausiai papildyti) turint omenyje, kad programa veiks konkrečioje kalbinėje ir kultūrinėje terpėje.

Programos elektroninis žinynas nėra meno ar filosofijos kūrinys ir ne taip svarbu, kad vertimas tiksliai atspindėtų jo originalo autoriaus stilių, pažiūras ir pan., bet labai svarbu, kad jis tiksliai, išsamiai, suprantamai ir vienareikšmiškai aprašytų programos veikimą. Todėl viską, kas jame parašyta, būtina sutikrinti, ar iš tikrųjų taip yra programoje.

Žinyne būna pavyzdžių, susijusių su kalba arba valstybe: pašto adresai, asmenvardžiai, piniginiai vienetai ir pan. Natūralu, kad jie organiškai įsilietų į kalbą, kuria parašytas žinynas. Pavyzdžiui, elektroninio pašto adresą *john.smith@mail.com* derėtų pakeisti į *vardenis.pavardenis@abc.lt* ar panašų.

Ten, kur kalbama apie numatytąją koduotę, reikia rašyti tą, kuri yra iš tikrųjų vartojama, pavyzdžiui, *ANSI* pakeisti *Windows-1257* arba trumpiau – į *Windows*.

Savaime suprantama, kad paveikslai ir kitos iliustracijos turi būti pakeisti tais, kurie iš tikrųjų matomi lokalizuotoje programoje. Taigi reikia išversti ne tik aprašymus, bet ir iliustracijas.

Programų žinynuose būna spragų – kas nors nepakankamai aprašyta arba išvis neaprašyta. Be to, lokalizuojant atsiranda naujų dalykų, kuriuos reikia aprašyti žinyne.

Dėl visų čia išvardytų priežasčių prie žodžio „vertimas“ dažniausiai reikia pridėti „ir adaptavimas“.

Žinynuose būna nuorodų į tinklalapius, kuriuose galima rasti papildomos informacijos, patarimų. Svarbesnius tekstus galima versti ir juos įdėti į programų svetaines arba į elektroninius žinynus. Tokie tekstai nėra originalios programos dalys ir jiems galioja tuose tekstuose nurodytos panaudojimo sąlygos, kurių privalu laikytis.

Be žinyno, programose būna kitų informacinių tekstų, kuriuos reikia išversti. Tai licencija, informacinis failas *readme* (skaityk*)*, informacija apie programą ir pan. Svarbiausias jų, kurį reikia būtinai išversti, yra licencija, nes Lietuvoje ne valstybine kalba parašyta licencija neturi juridinės galios. Dėl to komercinių programų lokalizuotojai visada teisiškai kvalifikuotai išverčia tų programų licencijas.

3.4. Visų tekstų redagavimas

Visi tekstai (sąsajos frazės, žinynas ir kt.) yra redaguojami. Redagavimas esti dalykinis ir kalbinis. Jei lokalizavimo darbus atliko informatikos specialistai, tai kalbinio redagavimo darbą atlieka redaktorius (lituanistas), turintis kompiuterinių tekstų redagavimo patirtį. Kai programas verčia lituanistai, tai tada situacija turėtų būti priešinga – nereikėtų kalbinio redagavimo, bet reikėtų dalykinio.

3.5. Kompleksinis lokalizacijos testavimas

Kompleksinis lokalizuotos programos testavimas atliekamas užbaigus visus (programos adaptavimo ir dialogo tekstų vertimo ir adaptavimo) darbus. Kompleksinį testavimą tikslinga suskirstyti į dvi dalis: 1) vidinis testavimas – atlieka ta pati grupė, kuri lokalizavo programą arba šiam darbui įpareigoti recenzentai ir 2) išorinis – atlieka kiti asmenys, specialiai testuojantys programą arba ją naudojantys kasdieniniame darbe ir informuojantys lokalizuotojus apie pastebėtas klaidas.

O‘Sullivano disertacijoje išskiriami tokie lokalizacijos testavimo bendrieji darbai [Sul01]:

1. Grafinės sąsajos elementų darna. Lokalizacijoje turi būti išlaikytas atitikimas originalios programos sąsajai (originale esantys sąsajos elementai turi būti ir lokalizacijoje).
2. Funkcionalumas. Lokalizavimo metu neturi būti pažeistos programos funkcijos.
3. Vaizdo estetika. Grafinių elementų išdėstymas, grupavimas, lygiavimas bei teksto išdėstymas grafiniuose elementuose. Tikrinama, ar nėra valdiklių ar teksto, kurie netelpa jiems skirtoje ekrano vietoje.
4. Sąsajos su kitomis programomis. Jei originali programa naudojasi kitų programų paslaugomis, tai tikrinamos šių sąsajų išlaikymas adaptavus programą.
5. Scenarijų testavimas. Jei į programą yra įtraukti scenarijai, parašyti, pvz., „Basic“, „JavaScript“ kalbomis, tai tikrinamas šių scenarijų veikimas lokalizacijoje.
6. Klaidų pranešimai. Tyrimai [Sul99] parodė, kad apie 35% programos lokalizuojamų išteklių eilučių yra klaidų pranešimai, todėl svarbu sukelti klaidų situacijas ir išbandyti klaidų pranešimų atitikimą klaidoms.
7. Diegimo programa. Jei programinė įranga turi diegimo programą, tai ji testuojama pagal tuos pačius punktus, kaip ir pati programa.
8. Daugiaplatformis veikimas. Jei programa yra skirta kelioms platformoms (pvz., įvairioms operacinėms sistemoms), tai atliekamas testavimas kiekvienoje iš jų. Testuojama lokalizuotose operacinėse sistemose.
9. Techninės įrangos testavimas. Testuojama su įvairia technine įranga, kuria naudojama lokalizuotoje programoje (įvairūs spausdintuvų, modemų, serverių, protokolų ir t. t. modeliai).
10. Kitkas. Testai, priklausomi nuo programinės įrangos specifikos.

O‘Sullivan visuose išskirtuose testavimo darbuose pabrėžia lokalizacijos identiškumą originalui funkciniu, estetiniu ir kt. požiūriais. Tačiau realizuojant trečiąjį lokalizavimo laipsnį (žr. 2.2 sk.) lokalizuojant gali neišvengiamai pasikeisti programos grafinė sąsaja ir kai kurios funkcijos, siekiant ją kuo daugiau pritaikyti konkrečiai programinei terpei. Todėl kompleksinio testavimo metu reikia turėti omenyje tokius adaptavimus.

3.6. Lokalizavimo tęstinumas

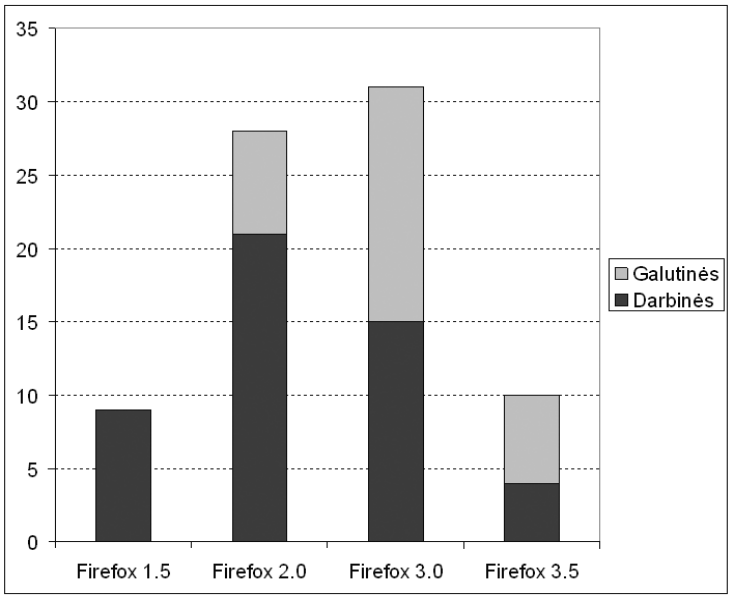
Programos tobulinamos, atsiranda naujų versijų. Tam, kad lokalizuota programos atmaina būtų gyvybinga, reikia operatyviai atnaujinti jos versijas, taisyti pastebėtus netikslumus, atsakyti į naudotojų klausimus.

Komercinės programos atnaujinamos maždaug kasmet. Kartu reikia atnaujinti ir lokalizacijas. Idealiu atveju originali ir lokalizuota versija turėtų pasirodyti vienu metu. Tai galima įgyvendinti esant nuolatiniam ryšiui su programos originalo autoriumi. Programos bandomąsias versijas ir lokalizavimo išteklių failus iš autoriaus galima gauti iš anksto ir juos lokalizuoti. Paskutiniai bandomųjų versijų koregavimai paprastai neturi įtakos programos lokalizavimui, išskyrus jos perkompiliavimą.

Internetinės programinės įrangos naujos versijos išleidžiamos dažnai, kartais atnaujinama internetu, automatiškai. Dėl to internetinės įrangos lokalizacijų išleidimas vienu metu su originalu yra beveik privalomas reikalavimas.

Pailiustruosime interneto naršyklės „Mozilla Firefox“ pavyzdžiu (remiantis duomenimis, skelbiamais „Mozilla.org“ oficialiojoje svetainėje). Per pastaruosius ketverius metus (nuo šio darbo autorės doktorantūros studijų pradžios – 2005 m. rudens – iki 2009 m. rudens) buvo išleistos 78 versijos. Iš jų 4 – pagrindinės versijos (kuriose yra daugiausia funkcinių pakeitimų ir kurios platinamos galutiniam naudotojui), 45 pagrindinių versijų

modifikacijų (kurios taip pat laikomos galutinėmis versijomis ir platinamos naudotojams, tačiau turi mažesnį skaičių funkcinių pakeitimų) ir 29 darbinių (alfa, beta, RC1, RC2 ir RC3) versijų, kurios skirtos testavimui, o ne galutiniam naudotojui (17 pav.).



17 pav. 2005–2009 m. išleistų naršyklės „Mozilla Firefox“ versijų pasiskirstymas

Lokalizavimo požiūriu dauguma darbų atliekama rengiant pagrindines versijas ir darbines versijas, kurių nuo 2005 iki 2009 m. buvo 29.

Kyla klausimas, kaip užtikrinti lokalizavimo tęstinumą, laiku, kartu su originaliomis, išleidžiant ir lokalizuotas versijas.

Kai kurios programos, pvz., internetinės bibliotekos, turi vidinį vertimo ir komponentą, kuriame eilutės yra padalintos į keletą grupių pagal naudojimo dažnumą. Tačiau tai, be abejo, mažiau skatina lokalizuotojus pateikti visų išteklių lokalizaciją [NWT05].

Literatūroje galima rasti nemažai šaltinių, skirtų lokalizavimo ir vertimo automatizavimo priemonėms [Ess00; Bar06; Mus04; Guz04; Was04 ir kt.]: ankstesnių versijų lokalizuotų išteklių importas į naujas versijas, vertimo atmintys, terminų duomenų bazės ir kt. Tačiau yra atlikta tyrimų, kuriais patvirtinama, kad automatizuojant atnaujinimo procesą nukenčia darbo kokybė [Bow05]. Be to, galimybė efektyviai pritaikyti tokias priemones priklauso nuo lokalizuojamųjų išteklių pateikimo metodo programoje (žr. 2.5 sk.) ir nuo programos naudojamo lokalės modelio (žr. 2.3 sk.).

3.7. Kultūros elementai internetinėje programinėje įrangoje

3.7.1. Kultūra lokalizavime

Kultūra suprantama gana įvairiai. Skirtingi kultūros apibrėžimai atspindi skirtingas jos supratimo teorijas arba kriterijus žmogaus veiklai įvertinti. Antropologai ir kiti biheviorizmo mokslininkai dažnai apibrėžia kultūrą kaip „visą spektrą žmogaus išmoktų elgesio šablonų“. Kultūrinė aplinka teikia individui emocinę erdvę, kurioje įsitikinimų rinkinys, nuostatos ir elgesys gali būti būdingi visiems tam tikros socialinės arba etninės grupės nariams [Emb77]. Kultūra teikia kontekstą, kuriame suprantamas pasaulis, elgesio, bendravimo, sąveikos ir supratimo taisykles [Ev01b].

Vaske ir Grantham [VG90] pažymi, kad kultūra yra:

a) adaptuojama (turi savybę prisitaikyti prie fizinės ir socialinės aplinkų tam tikrų sąlygų);

b) integruojama (kultūrą sudarančios savybės paprastai yra suderinamos viena su kita);

c) pastoviai keičiama (dėl adaptavimo prie tam tikrų kultūrinių įvykių arba integravimosi su kitomis kultūromis).

Nemažai autorių pažymi, kad egzistuoja ryšys tarp kultūros ir programinės įrangos tinkamumo naudoti (pvz., [Qin07]). Taip pat galima teigti, kad programinė įranga (ypač internetinė programinė įranga, kuri yra labai populiari dėl interneto paplitimo) veikia kultūros vystimąsi. Todėl programinės įrangos projektuotojas turėtų suprasti kultūrines charakteristikas ir kultūrinius skirtumus tam, kad galėtų sukurti programinę įrangą, kurią būtų galima pritaikyti prie įvairių kultūrų.

Kultūros klasifikacija buvo Hofstede [Hof91], Trompenaar [Tro97], Hall [Hal90] ir kt. mokslininkų tyrimų tema. Jie pasiūlė atitinkamai penkių, septynių ir keturių dimensijų modelius (2.4 sk.). Tačiau tai yra „tikrosios“ kultūros klasifikacija. Kalbėdami apie kultūrą programinėje įrangoje, galime nagrinėti kultūros atspindžius joje – konkrečius programinės įrangos elementus.

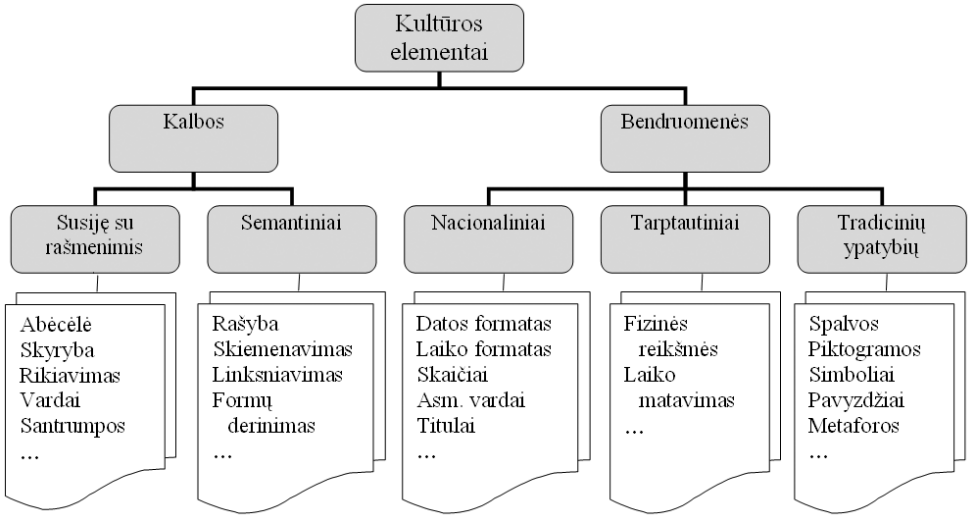
Čia pateikiami kultūros elementai yra išskirti remiantis anksčiau aprašytais lokalių modeliais ir atitinkamais standartais (2.3 sk.), taip pat daugiau kaip 7 metų internetinių programų lokalizavimo darbų, kuriuose dalyvavo šio darbo autorė, patirtimi:

- Naršyklės („Mozilla Firefox“, paketo „Mozilla“ naršyklė, „Opera“).
- Elektroninio pašto programos („Mozilla Thunderbird“, paketo „Mozilla“ elektroninio pašto programa, „Operos“ naršyklės pašto komponentas).
- Virtualiosios mokymosi aplinkos („Moodle“).
- Turinio valdymo sistemos ir jų pagrindų sukurtos mokymo turinio tvarkymo bendradarbiaujant sistemos („Plone“, „Lemill“).
- Pokalbių programos („Chatzilla“).
- Kalendoriaus ir laiko planavimo programos („Mozilla Calendar“, „Sunbird“).

Taip pat remiamasi ir kitų programų lokalizavimo patirtimi (raštinės programų paketas „OpenOffice.org“, tekstų rašyklė „Abiword“, komponentinio programavimo sistema „BlackBox“).

Pagrindiniai šio skyrelio rezultatai pateikiami straipsniuose [DJ08; DJ09].

Kultūros elementus galima sąlyginai suskirstyti į kalbos ir bendruomenės grupes (18 pav.). Kalbos grupės elementai gali būti susiję su rašymo sistema (pvz., abėcėle): rašmenys, skyryba, rikiavimas, vardų ir rašmenų vartojimas varduose, santrumpų vartojimas, arba priklausyti nuo kalbos semantikos: rašyba, skiemenavimas, linksniavimas, formų derinimas, vienos frazės komponavimas iš kelių. Bendruomenės grupės elementai ne tiek priklauso nuo kalbos, kiek nuo kitų susitarimų ir tradicijų, kurios gali būti priimtos nacionaliniu lygiu (pvz., vietiniai datos, laiko, asmenų vardų, titulų rašymo taisyklės), tarptautiniu lygiu (pvz., laiko matavimas, fizinės reikšmės) arba priklausyti nuo gilių ir sunkiau vienareikšmiškai nusakomų tradicijų (spalvų, piktogramų, metaforų ir kt. vartojimas).



18 pav. Kultūros elementų internetinėje programinėje įrangoje struktūra

Kai kurių dažniausiai naudojamų lokalės elementų reikšmės paprastai būna įtrauktos į operacinę sistemą (ženklų koduotė, dešimtainės trupmenos skirtukas, tūkstančių skirtukas, datos ir laiko formatas, pinigų vienetų formatas ir t. t.). Internacionalizuota programa turėtų šiuos elementus aptikti iš operacinės sistemos nuostatų ir juos naudoti. Priešingu atveju turėtų būti galimybė nurodyti su lokale susijusias numatytąsias nuostatas programos lokalizavimo metu, t. y. įtrauktos į lokalizuojamąją programos dalį.

Kurios nors su lokalizavimu susijusios programos nuostatos nebuvimas arba jos neįtraukimas į lokalizuojamus išteklius gali būti laikomas internacionalizavimo klaidomis (trūkumais).

Toliau analizuosime pagrindinius internetinėje programinėje įrangoje naudojamus kultūros elementus, kurie kelia problemų juos lokalizuojant. Pagal aukščiau pateiktą klasifikaciją čia analizuojami kalbos ir bendruomenės nacionaliniai ir tarptautiniai elementai. Tradicinių ypatybių pagrindiniai aspektai pateikti 2.4 skyrelyje.

3.7.2. Kalbos elementai

Nuo gilių įvairių kalbų skirtumų priklausantys kalbos elementai nėra įtraukti į šiuolaikinius lokalių modelius ir standartus dėl sudėtingo jų formalizavimo. Dar kiti elementai, nors ir yra įtraukti į lokalės formaliuosius aprašus, nėra realizuojami programinės įrangos tam tikrose vietose. Pavyzdžiui, nepaisant to, kad kalbos naudojama ženklų koduotė yra įtraukta į formalų lokalės aprašą (kiekviename iš egzistuojančių lokalių modelių, žr. 2.3 sk.), galime pastebėti, kad nacionalinių kalbos ženklų paprastai nepaisoma įvairių internetinės programos objektų varduose (failai, katalogai, interneto sritys, prisijungimo vardai, slaptažodžiai ir kt. [Jev06]. Kita problematiška sritis – tai semantiškai išreiškiami kalbos elementai, pavyzdžiui, žodžių gramatinių formų derinimas, tekstų komponavimas atsižvelgiant į kontekstą. Toliau aptarsime minėtas problemas ir atitinkamus kalbos elementus internetinėje programinėje įrangoje.

3.7.2.1. Abėcėlės ir vardai

Programinė įranga operuoja įvairiais objektais, kurių vardai naudojami ne tik kompiuterių, bet ir žmonių. Gimtąja kalba užrašyti vardai yra lengviau suprantami, įsimenami, asocijuojami, manipuliuojami, taisomi, perteikiami bendraujant, o taip pat

asocijuojami su ta kalba [Dür03]. Kai kurioje senesnėje programinėje įrangoje egzistavęs reikalavimas naudoti tik 26 raides (anglų k. abėcėlė) varduose yra labai ribojantis net toms kalboms, kurios vartoja lotynų abėcėlę (nekalbant apie kitų kalbų sistemų rašmenis), nes jų abėcėlės turi papildomų raidžių (pvz., å, š, ž, …), o kai kurios anglų kalbos raidės nėra vartojamos (dažniausiai q, w, ir x).

Abonento prisijungimo vardai. Daugelyje internetinių programų (pvz., virtualiosiose mokymosi aplinkose, el. pašto programose, pokalbių programose ir kt.) tikrinamas naudotojo tapatumas. Kaip vienas iš tokio tikrinimo komponentų paprastai pateikiamas abonento vardas, kuriuo naudotojas prisijungia prie sistemos. Deja, šiame varde daugelis sistemų leidžia vartoti tik pagrindinės lotynų abėcėlės raides, skaitmenis ir pabraukimo ženklus (_). Kai kurios sistemos (pvz., virtualioji mokymosi aplinka „ATutor“) naudoja abonento vardą ne tik naudotojo vidiniam identifikavimui sistemoje, bet ir kreipiniams į naudotoją sistemoje (abonento vardas tampa matomas jam pačiam ir kitiems naudotojams, naudojamas bendraujant su kitais sistemos naudotojais). Todėl natūralu, kad naudotojas turėtų pasirinkti vardą, sudarytą iš jo gimtosios kalbos abėcėlės raidžių (ne tik iš ASCII ženklų). Nepaisant to, kad techninių kliūčių naudoti įvairių kalbų ženklus abonentų varduose nėra, programų projektuotojai vis dar vengia realizuoti šią savybę savo produktuose. Dėl to lokalizuoti produktai nebeatitinka natūralumo reikalavimo toje kultūrinėje terpėje, kuriai jie yra lokalizuojami.

Asmenvardžiai. Daugelyje internetinių programų egzistuoja naudotojo duomenų (profilio) sritis, kur įrašomi ir laikomi asmeniniai duomenys. Natūralu, kad visos asmens tikrojo vardo ir pavardės (asmenvardžio) raidės turėtų būti priimamos (žmogus neturėtų keisti savo vardo ar pavardės rašybos tam, kad užsiregistruotų kurioje nors sistemoje).

Virtualiųjų mokymosi aplinkų analizė lokalizavimo požiūriu [Jev06] parodė, kad nacionalinių ženklų vartojimas asmenvardyje priklauso nuo serverio nuostatų: jei nuostatos teisingai parinktos lokalės atžvilgiu, tai visi tos lokalės kalbos ženklai gali būti vartojami. Analogiška situacija yra ir su kitomis dabartinėmis internetinėmis programomis, t. y. teisingai suderinus serverį ar operacinę sistemą, tokius ženklus galima surinkti asmenvardyje. Tačiau tenka pastebėti, kad nemažai naudotojų vengia naudoti savo asmenvardžiuose kai kurių raidžių su diakritiniais ženklais, t. y. rašo savo asmenvardžius su rašybos klaidomis. Pokalbių programos „Skype“ naudotojų vardų tyrimai rodo, kad teisingai užrašyti vardai ir pavardės (naudojant raides su diakritiniais ženklais) svyruoja nuo 15% iki 96%, priklausomai nuo kalbos. Tirtos vokiečių, čekų, danų, estų, islandų, latvių, lenkų ir lietuvių kalba užrašyti vardai [GP07]. Toks aukštas „neraštingumo“ lygis gali būti paaiškintas kai kurioje programinėje įrangoje egzistuojančiu ribojimu abonento vardų ženklams (žr. aukščiau).

Slaptažodžiai. Slaptažodis – tai dar vienas naudotojo tapatumo tikrinimo komponentas daugelyje internetinių programų. Jis paprastai sudaromas iš raidžių ir skaitmenų. Tačiau daugelis sistemų susiaurina raidžių aibę iki ASCII ženklų, o tai nėra natūralu lokalėms, kurių kalbose vartojamos kitos abėcėlės. Be to, ženklų aibės susiaurinimas sumažina slaptažodžio saugumą.

Failų vardai. Operacinėse sistemose leidžiama vadinti failus tikrais vardais. Naudojamos raidės, skaitmenys ir kai kurie teksto ženklai, pvz., taškas, brūkšnelis ir kt. Kiti ženklai, turintys specialią reikšmę įvardijant failo kelią, pvz., pasvirę brūkšniai, kableliai ir kt., objektyviai nėra leidžiami. Naudotojas gali parinkti failų vardus savo kalba taip, kad vardas nusakytų failo turinį. Tas pats galioja ir katalogų (aplankų) vardams.

Galima išskirti kelis failų naudojimo atvejus:

1) dokumentų laikymas vietiniame kompiuteryje,

2) dokumentų mainai tarp kompiuterių naudojant keičiamąsias laikmenas,

3) siunčiant dokumentus kaip el. laiško dalį ar priedą arba tiesiogiai, pvz., pokalbių programose,

4) tinklalapių ar kitokio turinio laikymas serveryje,

5) naudojimas programos viduje.

1. Šiuo metu nėra kliūčių failus, laikomus vietiniame kompiuteryje, vadinti vardais, kuriuose yra ne pagrindinės lotynų abėcėlės raidžių, jeigu operacinės sistemos lokalės nuostatos parinktos korektiškai. Galima kurti katalogų medį, kurio mazgų pavadinimai sudaromi ir natūralios kalbos žodyno. Jei programa nepriima tokių failų, ją galima laikyti pasenusia.

2. Dokumentų (failų) mainai tarp kompiuterių naudojant keičiamąsias laikmenas veikia gerai, jeigu tas pats 8 bitų kodavimas naudojamas abiejuose kompiuteriuose. Taigi mainai gali būti atliekami korektiškai nacionaliniame kontekste, taip pat gali būti naudojamos kaimyninių valstybių kalbos, naudojančios tą pačią koduotę. Unikodo realizacija pateiktų universalų sprendimą.

3. Dokumentų varduose, kurie yra siunčiami kaip elektroninių laiškų dalys arba priedai, arba pokalbių programų pagalba, ne ASCII ženklai yra vis dar retai vartojami dėl ribojimų, kurie anksčiau galiojo kai kuriose pašto programose, neteisingo pašto protokolų interpretavimo, kad vardai privalo turėti tik ASCII ženklus. Vardai prieš siunčiant failus yra koduojami UTF-8 kodų sekomis %FF, kuriose yra tik ASCII ženklai, o gavus – vardai iškoduojami. Siuntėjas ir gavėjas mato tik tikrus natūralius failų vardus. Metodas veikia tarptautiniu lygmeniu. Deja, ne visos programos priima vardus su ne ASCII ženklais.

4. Tinklalapių ir kito turinio laikymas serveryje, kai failų vaduose pavartotos nacionalinės kalbos raidės, yra teoriškai išspęstas. Šiuo atveju vardai užkoduojami tokiu pačiu būdu, kaip elektroninių laiškų priedų vardai. Tačiau dar yra problemų dėl praktinės šio sprendimo realizacijos.

5. Failų ir katalogų (aplankų) vardai, kuriais operuoja programa ir kurie nėra matomi programos naudotojui, gali būti palikti neišversti. Tačiau būtų natūralu, kad visi vardai, kurie yra rodomi naudotojui, pvz., failų vardai, naudojami kaip elektroninio pašto aplankų vardai, turi būti lokalizuoti. Šiuo atveju reikalingas dinaminis failo vardo konvertavimas. Šio sprendimo realizacija – programos projektuotojo arba lokalizuotojo užduotis.

Interneto sričių vardai. Naudotojas, dirbdamas su interneto programine įranga, dažnai operuoja sričių vardais. Ilgą laiką tik 26 raidės (anglų kalbos abėcėlė) galėjo būti vartojamos sudarant sričių vardus. 2003 m. buvo publikuotas dokumentas, reglamentuojantis tarptautinių ženklų naudojimą sričių varduose [FHC03]. Pagal šį dokumentą srities vardas (koduotas Unikodu) yra konvertuojamas į ASCII ženklų eilutę, naudojant „Punycode“ kodavimą (tam, kad perduodami duomenys derėtų su esamomis DNS sistemomis), o prieš rodant srities vardą naudotojui, jis yra iškoduojamas į Unikodo ženklus. Naujausios versijos populiariausių naršyklių turi priemonių darbui su internacionalizuotais sričių vardais. Tačiau vis dar daugelis organizacijų vengia registruoti ir naudoti tokius sričių vardus. Viena iš galimų priežasčių – apgaulės rizika naudojant homografinius ženklus iš skirtingų abėcėlių, kai metodas realizuojamas tarptautiniu mastu. Kita priežastis – ne visos programos dar korektiškai veikia su internacionalizuotais sričių vardais, naudotais hipernuorodose.

3.7.2.2. Semantiniai kalbiniai elementai

Vienaskaitos ir daugiskaitos formų derinimas. Internetinėje (ir kitoje) programinėje įrangoje dažnai turi dialogo su naudotoju ekranus, kuriuose šalia skaičiaus (įvesto naudotojo ar programos sugeneruoto) rodomas atitinkamo objekto pavadinimas. Tačiau programa retai kada parenka teisingą objekto vardo formą, kuri atitiktų šalia esantį skaičių, pavyzdžiui, „1

objektas“, „9 objektai“, „11 objektų“. Anglų kalboje vartojamos 2 daiktavardžio formos (vienaskaita ir daugiskaita), tuo tarpu kitose kalbose vartojamos trys formos (lietuvių, rusų, lenkų ir kt. kalbose yra trys formos: viena vienaskaitos, dvi daugiskaitos) ar daugiau formų (pvz., maltiečių, slovėnų kalbose yra keturios formos, o arabų kalboje – šešios) (9 lentelė). Taigi iš tikrųjų internacionalizuotoje programoje formų skaičius ir visos formos turėtų būtų įtrauktos į lokalizuojamus išteklius ir dinamiškai derinamos.

9 lentelė. 75 analizuotų kalbų pasiskirstymas pagal vienaskaitos ir daugiskaitos formų kiekį

Formų skaičius	Kalbų skaičius	Kalbos
1	12	Gruzinų, indoneziečių, korėjiečių, persų, turkų, vengrų ir kt.
2	46	Anglų, baskų, bengalų, bulgarų, danų, estų, farerų, fryzų, hebrajų, hindi, mongolų, norvegų, vokiečių ir kt.
3	14	Afrikanų, airių, baltarusių, bosnių, čekų, kroatų, lietuvių, latvių, lenkų, rumunų, rusų, slovakų, serbų, ukrainiečių.
4	2	Maltiečių, slovėnų
5	0	–
6	1	Arabų

Straipsnyje [Boi05] pateikiamas formų derinimo ir eilučių kintamųjų reikšmių parinkimo būdas taikant baigtinius automatus.

Gramatinės formos. Daugelyje kalbų (pvz., lietuvių, suomių, lenkų ir kt.) vardai programos įvairiuose dialogo languose gali turėti įvairias formas. Į originalią programinę įrangą praktiškai neįmanoma įdėti visų kalbų visų linksnių generavimo modulį, tačiau galimybė prijungti tam tikros kalbos vardų generavimo modulį galėtų būti numatoma.

Pavyzdžiui, lietuvių kalboje šauksmininko linksnio generavimas reikalingas praktiškai visuose prisijungimo prie internetinės programinės įrangos, veikiančios serveryje, languose. Be to, vardas gali būti asmens vardas, pavardė, ar abiejų kombinacija. Nei viena iš analizuotų virtualiųjų mokymosi aplinkų [Jev06] tokios galimybės dar neturi. Tuo tarpu kai kuriuos bankinės ir kitos su verslu susijusios sistemos jau linksniuoja asmenvardžius.

Komponuojamos ir parametrizuotos eilutės. Nemažai problemų lokalizavimo metu kelia sakiniai arba frazės, sukomponuotos iš keleto atskirų eilučių, o taip pat parametrai, įdėti į lokalizuotinas eilutes. Tokia parametrizacija gali tikti vienai kalbai, bet būti visiškai nepriimtina kitoje kalboje, kadangi eilučių eilė (tvarka), žodžių formos didžiųjų ir mažųjų raidžių vartojimas eilutės viduje skiriasi įvairiose kultūrose. Todėl parametrų naudojimas turėtų būtų pakomentuotas ištekliuose, įterpiant į juos lokalizavimo komentarus, kad lokalizuotojas galėtų įdėti parametrą į tinkamą eilutės vietą, arba ieškoma naujų sprendimų, kaip pateikti informaciją apie parametrizavimo ir komponavimo atvejus ir kontekstą, kuriame visa eilutė bus naudojama.

Rašybos tikrinimas yra būtina internetinės programinės įrangos savybė. Rašybos tikrinimo komponentas gali būti naudojamas ne tik rengiant dokumentus raštinės paketo programomis, bet taip pat tikrinant rašybą internetinių pokalbių tekstų, pildant formas internete. Kadangi elektroninių laiškų ir trumpųjų žinučių (SMS) siuntimas iš interneto svetainių tapo labai populiarus, tai el. laiškų ir SMS žinučių rašybos tikrinimas tapo vienu iš interneto svetainėje pildomos formos rašybos tikrinimo taikymu. Taigi tai yra būtinas naršyklės, pokalbių programos, el. pašto programos komponentas. Įvairiose rašto sistemose ir kalbose rašybos tikrinimo taisyklės yra skirtingos. Dėl didelio kiekio išimčių iš pagrindinių formaliųjų taisyklių jas gana sunku realizuoti. Pastaraisiais metais populiariais tapo atvirieji rašybos tikrinimo komponentai, kurių kokybė nuolat gerėja ir kuriuos galima laisvai pritaikyti specifiniams kultūriniams ir programiniams poreikiams.

Skiemenavimas yra susijęs su eilučių laužymu. Jame ieškoma optimalios vietos, ties kuria gali būti laužoma eilutė. Taip pat yra susijęs su rašybos tikrinimu dėl panašių gramatinės ir morfologinės analizės algoritmų. Eilutės laužymo vieta aptinkama dviem etapais: 1) identifikuojama galima eilutės laužymo vieta (ji gali būti žymima valdymo ženklais, nurodančiais galimą eilutės laužymo vietą (pvz., U+000D, U+2028 ir kt.) arba vietą, kurioje negalima laužyti eilutės (pvz., U+00A0)); 2) teksto formatavimo algoritmas išrenka optimalią eilutės laužymo vietą. Skirtingoje programinėje įrangoje skiemenavimas realizuotas taikant įvairius algoritmus. Unikodo standartas teikia specifikaciją bendro algoritmo, tinkamo įvairiems eilučių laužymo atvejams.

3.7.3. Bendruomenės elementai

Šiame skyrelyje analizuojami nacionaliniai ir tarptautiniai bendruomenės elementai. Tradicinių ypatybių elementų pagrindiniai aspektai pateikti 2.4 skyriuje.

Datos formatas. Data gali būti pateikiama trumpu arba ilgu formatais. Ilgas datos užrašas (pvz., 2009 m. liepos 10 d.) yra sudėtingesnis, kadangi jame metus ir dienas atitinkantys skaičiai pateikiami kartu su teksto eilutėmis, kurių forma ir linksniai gali kisti. Trumpas datos užrašas taip pat yra skirtingas įvairiose kultūrose. Pavyzdžiui, Lietuvoje trumpas datos užrašas sudaromas iš trijų dalių. Pirmoji iš kairės – 4 skaitmenys, nusakantys metus, antroji – du skaitmenys, nusakantys mėnesį, ir trečioji – du skaitmenys, reiškiantys dieną. Jei mėnesio arba dienos numeris yra skaičius, mažesnis už 10, tai kairiau nuo reikšminių vienetų skaitmenų prirašoma po nereikšminį nulį. Lietuvoje trumpo datos užrašo dalys skiriamos brūkšneliais (pagal tarptautinį standartą ISO 8601) arba tarpais. Pirmenybė teikiama tarptautiniu standartu nusakytam skirtukui (brūkšneliui), pavyzdžiui, 2009-07-10. Kitose lokalėse naudojama kitokia datos komponentų tvarka (pvz., diena, mėnuo, metai arba mėnuo, diena, metai) ir skirtukai (pvz., /).

Lokalizuotinoje programinėje įrangoje turėtų būti galimybė rodyti datą pagal tarptautinį standartą (ISO 8601) arba nurodyti lokalizuojamuose ištekliuose pageidautiną formatą, datos komponentų (metų, mėnesio ir dienos) skirtukus ir eilę.

Virtualiųjų mokymosi aplinkų analizė lokalizavimo požiūriu [Jev06] parodė, kad dažnai vartojami sutrumpinti datos komponentų pavadinimai, pavyzdžiui, mėnesių santrumpos „Aug“, „Oct“ ir kt. Ne visose kalbose analogiškos santrumpos yra priimtinos.

Laiko formatas. Laiko matavimas gali būti priskirtas prie tarptautinių kultūros elementų, tačiau jo pateikimo formatas gali būti tarptautinis arba nacionalinis. Tarptautinis laiko formatas yra apibrėžtas tarptautiniame standarte ISO 8601.

Laikas gali būti pateikiamas 12 ar 24 valandų formatais su įvairiais valandų ir minučių skirtukais (pvz., taškas, dvitaškis, tarpas). Pavyzdžiui, Didžiojoje Britanijoje naudojamas 12 valandų formatas (pvz., 9:30 AM, 5:45 PM), o daugelyje kitų Europos lokalių (pvz., Lietuvoje) naudojamas 24 valandų formatas (pvz., 21:15).

Laiko pateikimas internetinėje programinėje įrangoje, veikiančioje serveryje (pvz., virtualiojoje mokymosi aplinkoje) dažniausiai priklauso nuo serverio nuostatų ir pačios programinės įrangos nuostatų.

Pirmoji savaitės diena. Įvairiose lokalėse yra skirtingi susitarimai dėl pirmosios savaitės dienos. Pavyzdžiui, JAV – tai sekmadienis, tuo tarpu Lietuvoje, Lenkijoje, Rusijoje ir kt. šalyse – tai pirmadienis.

Dešimtainės trupmenos skirtukas. Tarptautiniame standarte ISO 31-1 apibrėžti du dešimtainės trupmenos skirtukų tipai: kablelis ir taškas. JAV dešimtainės trupmenos skirtukas yra taškas, o daugelyje Europos šalių – kablelis. Neteisingas dešimtainės trupmenos skirtukas tam tikroje lokalėje gali būti supainiotas su tūkstančių skirtuku ir tapti klaidingų

skaičių reikšmių suvokimų priežastimi. Kai kurios programos automatiškai nuskaito skirtuko tipą iš operacinės sistemos lokalės nuostatų. Jei ne, tai skirtuko tipą turėtų būti galima pasirinkti programos lokalizavimo metu. Deja, yra programų, kuriose šis skirtukas yra užkoduotas programos viduje ir jį labai sunku pakeisti – reikia papildomų programavimo pastangų lokalizavimo metu.

Tūkstančių skirtukas. JAV naudojamas tūkstančių skirtukas kablelis, o daugelyje Europos šalių – taškas arba tarpas. Neteisingas tūkstančių skirtukas gali būti supainiotas su dešimtainės trupmenos skirtuku ir tapti klaidingų skaičių reikšmių suvokimų priežastimi.

Telefono numerio formatas. Telefono numerių formatas įvairiose šalyse gali būti skirtingas. Internetinėje programinėje įrangoje jį paprastai galima rasti naudotojo duomenų profilyje. Jei sritis telefono numeriui surinkti yra iš keleto langelių, tai lokalizavimo metu turėtų būti galimybė pakeisti langelių skaičių ir (arba) eilę. Vienas iš nesuderinamumų šalinimo sprendimų – naudoti vieną langelį telefono numeriui surinkti.

Dialogų laukų eilė. Internetinėje programinėje įrangoje paprastai naudojami atskiri laukai (langeliai) ar valdikliai duomenims, kurie priklauso nuo lokalės, įvesti. Pavyzdžiui, adreso įvedimo dialoge gali būti daugeliui lokalių nereikalingas laukas „Valstija“, datai ir laikui nurodyti pateikiami trys atskiri langeliai: diena, mėnuo, metai. Lokalizuojant reikia pakeisti jų tvarką į atvirkščią (tam reikia keisti programos pirminių tekstų fragmentą).

Asmens vardo formatas. Asmenų vardai ir pavardės yra vartojami įvairiai, skiriasi jų pateikimo tvarka įvairiose kultūrose (pvz., „Vardas Pavardė“, „Pavardė Vardas“, „Pavardė, Vardas“). Internetinėje programinėje įrangoje vardai paprastai naudojami pasveikinant prisijungusį naudotoją, pradedantį darbo seansą, taip pat kaip išorinis naudotojo identifikavimo būdas, kuris gali būti matomas kitiems naudotojams (pvz., virtualiojoje mokymosi aplinkoje ar turinio valdymo sistemoje). Todėl vardo ir pavardės tvarka turėtų būti įtraukta į lokalizuojamuosius išteklius, pvz., kaip atskiras parametras, arba vardui ir pavardei vartojant skirtingus parametrų (kintamųjų) vardus.

3.8. Išvados

Atlikus lokalizavimo proceso tyrimo analizę, galima padaryti išvadas, kad:

1. Prieš lokalizuojant programą reikalingas parengiamasis darbas, kurio metu įvertinama programa internacionalizavimo požiūriu, jos autoriaus ir savo galimybės prognozuojant situaciją keleriems metams į ateitį, numatant lokalizacijų atnaujinimo darbus. Kadangi dalis programos internacionalizavimo klaidų yra aptinkamos lokalizavimo metu, tai lokalizacijos kokybė priklausys ne tik nuo lokalizuotojų kvalifikacijos, bet ir nuo autoriaus pasirengimo bendradarbiauti su jais ir įkelti aptiktų klaidų taisymus į programos internacionalizaciją.
2. Vien programos grafinės naudotojo sąsajos frazių vertimas be testavimo ir vertimų derinimo sudaro tik nedidelę, tačiau itin svarbią, viso lokalizavimo sąnaudų dalį. Dialogo lokalizuotinų tekstų yra daug, tačiau jie pateikiami be kontekstinės informacijos. Tai pagrindinė priežastis, dėl ko pirminio (juodraštinio) sąsajos tekstų vertimo kokybė būna žema. Korektiškas terminų parinkimas, taiklus frazių vertimas nuo pat pradžių palengvintų lokalizacijos testavimą. Todėl būtina ieškoti būdų, kaip pagerinti tekstinių išteklių lokalizavimo kokybę dar iki testavimo.
3. Identifikavus, klasifikavus ir išanalizavus pagrindinius internetinės programinės įrangos kultūros elementus, galima pastebėti, kad tradicinių ypatybių elementų bei kalbos semantinių elementų nėra įtraukta į egzistuojančius lokalių formaliuosius aprašus, dėl to jie dažniausiai nėra numatomi internacionalizuojant programą ir kelia daugiausia problemų lokalizavimo metu.

4. LOKALIZUOJAMŲJŲ IŠTEKLIŲ METAINFORMACIJA IR JOS FORMALIZAVIMO METODAS

Atsižvelgiant į antroje ir trečioje šio darbo dalyse atliktą literatūros, lokalizavimo proceso bei pagrindinių internetinės programinės įrangos kultūros elementų analizės rezultatus, šioje darbo dalyje pateikiamas metodas, skirtas įtraukti į lokalizuojamuosius išteklius lokalizavimui naudingą kontekstinę informaciją (metainformaciją).

Pastebėta, kad įvairių tipų dvejetainiai ištekliai nekelia rimtų lokalizuojamos programos naudotojo sąsajos ir funkcionalumo problemų [Sul01, p. 205]. Nepaisant to, kad su tokiais ištekliais susiję kultūriniai aspektai nėra įtraukti į lokalių modelis (žr. 3 sk. 3 išvadą), jų lokalizavimas yra pakankamai nepriklausomas: dvejetainis objektas (pvz., paveikslas) pakečiamas tokių pačių parametrų lokalizuotu objektu. Lokalizuojama remiantis atitinkamos kultūros tradicijomis, taikant 2.4 skyriuje apžvelgtas kultūrines dimensijas. Tam tikrais atvejais pakeičiama nuoroda iš išteklių į lokalizuotą objektą (svarbu, kad nuoroda į lokalizuotiną dvejetainį objektą būtų iškelta į lokalizuojamuosius išteklius). Žymiai daugiau problemų kyla, kai yra lokalizuojami tekstiniai ištekliai. Dauguma tekstinių išteklių semantikos aspektų nėra apibrėžta egzistuojančiuose lokalių modeliuose (2.3 sk.). Todėl nagrinėsime tekstinius lokalizuojamuosius išteklius ir vartodami terminą *lokalizuojamieji ištekliai* toliau turėsime omenyje tekstinius lokalizuojamuosius išteklius.

4.1. Tekstinių lokalizuojamųjų išteklių struktūra ir ypatumai

Apibendrindami 2.5 skyriaus rezultatus, pastebėsime, kad tekstiniai programinės įrangos ištekliai, skirti lokalizavimui, pateikiami lokalizuotojams vardų ir reikšmių porų aibe

$$L = \{v, e\},$$

čia $v \in V$,

$e \in E$,

V – teksto eilučių vardų aibė,

E – teksto eilučių (tekstinių lokalizuojamųjų išteklių elementų turinio) aibė.

Čia teksto eilutės – tai ne tik ekrane rodomi tekstai programai veikiant, bet taip pat kai kurių funkcijų parametrų reikšmės, šriftų, koduočių ir kt. vardai, kurių vertimas gali turėti ir funkcinį poveikį programai.

L aibė gali būti suskirstyta į failus ir katalogus (priklauso nuo išteklių atskyrimo metodo ir programuotojo pasirinkto sprendimo).

Lokalizuojamos eilutės tokių failų viduje pateikiamos iš eilės (tiesiškai), nenurodant (arba iš dalies nurodant ir tik retais atvejais, pvz., XLIFF formatas) realiai egzistuojančių sąryšių su kitomis eilutėmis ir naudotojo grafinės sąsajos elementais, kuriuose eilutės bus vaizduojamos programai veikiant (priklausomai nuo konteksto) (žr. 2.5 sk.).

Programinės įrangos tekstai, pateikiami dialogo languose, yra lakoniški, atsieti nuo konteksto, juose gausu naujų terminų (kurių gali dar nebūti kalboje, kuriai lokalizuojama). Todėl programos lokalizuotojas susiduria su šiomis pagrindinėmis problemomis:

1. mato tik atskirus žodžius ar frazes be konteksto – lokalizuodamas programą, žmogus dirba su dialogo eilučių duomenų baze (aukščiau apibrėžtos aibės L elementais);
2. lokalizuojamų tekstų kontekstas pamatomas tik programai veikiant ir (iš dalies) nagrinėjant pirminius programos tekstus (jeigu tai leidžia daryti programos licencija), o tai labai sunkus ir imlus darbas;

3. dalis programos dialogo tekstų atsiranda tik esant ypatingoms situacijoms (klaidoms, kitų programų poveikiui), kurias sudėtinga arba neįmanoma sumodeliuoti testuojant lokalizuotą programą.

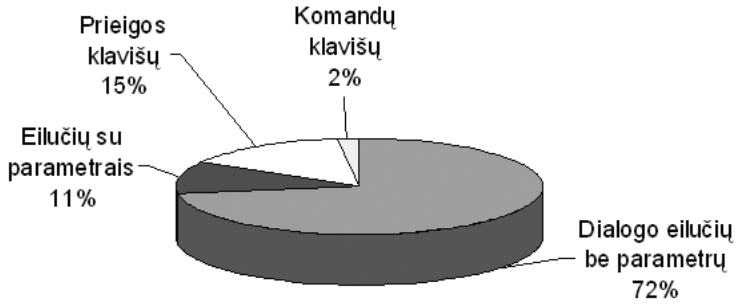
Kadangi dauguma lokalizuojamų programų lokalizuojama iš anglų kalbos, tai dar viena grupė problemų, kurios iškyla lokalizavimo metu, kyla iš anglų kalbos neapibrėžtumo bei jos kompiuterininkų žargono, vartojamo programavime. Anglų kalboje gausu sinonimų, tuo pačiu žodžiu vadinami skirtingi dalykai, dažnai, kai nėra konteksto, sudėtinga suprasti, ar žodis yra veiksmažodis, ar daiktavardis (anglų kalba rašoma vienodai, pvz., *File* (*failas* – daiktavardis, *įtraukti* [į aplanką] – veiksmažodis), *login* – *prisijungti* (veiksmažodis), *prisijungimas*, *prisijungimo vardas* (daiktavardis)), anglų kalbos leksika įvairuoja, net toje pačioje programoje tai pačiai funkcijai įvardyti vartojami skirtingi pavadinimai, kompiuterių terminijoje gausu metaforų, žargono, terminijai stokoja sistemingumo [Gri08].

Dėl minėtų veiksnių programos dialogo tekstų vertimo ir adaptavimo sąnaudos yra kelis kartus didesnės, palyginti su paprasto rišlaus teksto vertimu.

4.1.1. Išteklių eilučių struktūros tyrimas remiantis internetinių programų pavyzdžiais

Šiame skyrelyje apibūdinsime lokalizuojamųjų išteklių struktūrą remdamiesi internetinių programų pavyzdžiais: naršykle „Mozilla Firefox“ kaip kliento programa, virtualiąja mokymosi aplinka „Moodle“ kaip serverio programa.

Išanalizavę interneto naršyklės „Mozilla Firefox“ trijų iš eilės leistų versijų lokalizuojamųjų išteklių struktūrą, eilučių tipų procentinį pasiskirstymą pateikiame 19 paveiksle.



19 pav. Lokalizuojamųjų tekstinių išteklių eilučių pasiskirstymas remiantis „Mozilla Firefox“ pavyzdžiu

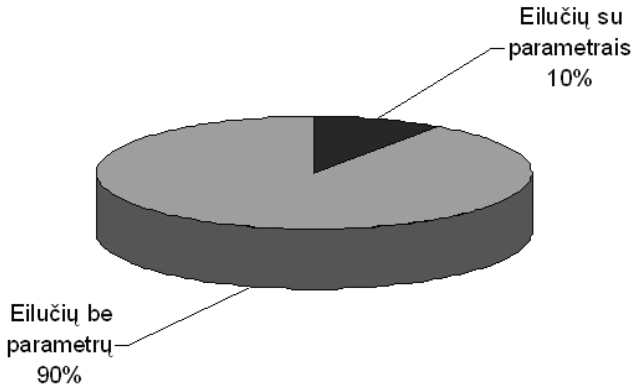
17% visų išteklių eilučių sudaro prieigos ir komandų klavišai, kuriuos reikia parinkti lokalizavimo metu, ir ne tik parinkti juos prasmingus, bet ir patikrinti, kad jie nėra padubliuoti jų galiojimo srityje.

11% eilučių viduje yra pavartotas vienas ar keli parametrai – kintamieji, kurių vietoje bus įrašyta arba kita eilutė iš išteklių, arba tam tikra reikšmė, kurį gali būti skaičiuojama programos vykdymo metu. Kiekvienoje tokioje eilutėje yra vidutiniškai 1,4 parametrai. Tai potencialių klaidų lokalizavimo metu eilutės, kadangi parametro reikšmė dažniausiai išaiškėja tik programai veikiant.

Kitos eilutės – tai eilutės be parametro, atitinkančios programos dialogų tekstus.

Lokalizavimo komentarų skaičius sudaro tik 4% visų lokalizuojamųjų išteklių eilučių. Tai reiškia, kad visoje likusioje eilučių dalyje trūksta konteksto.

Remiantis „Moodle“ virtualiosios mokymosi aplinkos lokalizuojamųjų išteklių tyrimais, gavome panašius rezultatus eilučių su parametrais kiekio požiūriu (20 pav.).



20 pav. Lokalizuojamųjų tekstinių išteklių eilučių pasiskirstymas remiantis „Moodle“ pavyzdžiu

Lokalizavimo komentarų „Moodle“ lokalizuojamuose ištekliuose nėra, o eilučių vardai yra neprasmingi (arba atkartoja eilutę, arba vienas-du žodžiai, nenusakantys eilutės konteksto programoje).

Kadangi „Moodle“ sistema yra serveryje veikianti internetinė programa, tai joje, kaip daugelyje kitų serveryje veikiančių programų, nenaudojami prieigos ir komandų klavišai.

4.1.2. Lokalizuojamųjų išteklių eilučių ilgiai ir kontekstas

Išanalizavus „Mozilla Firefox“ naršyklės lokalizuojamųjų eilučių ilgius, paaiškėjo, kad eilučių ilgis svyruoja nuo 1 ženklo iki 549 ženklų (angliškoji versija) ir nuo 1 ženklo iki 633 ženklų (sulietuvinta versija). Atmetus prieigos ir komandos klavišus, kurie yra vieno ženklo ilgio, lieka 4839 eilutės, jas vadinsime reikšminėmis eilutėmis. Reikšminių eilučių vidutinis ilgis yra 29,7 ženklai. 72% reikšminių eilučių yra ne ilgesnės kaip 30 ženklų (t. y. 2–4 vidutinio ilgio žodžiai). Atsižvelgus į eilučių skaičių pasiskirstymą pagal ženklų skaičių gauta, kad dažniausiai pasitaiko 3–18 ženklų ilgio eilutės, t. y. nuo vieno iki trijų–keturių žodžių eilutės.

Išanalizavus virtualiosios mokymosi aplinkos „Moodle“ lokalizuojamuosius išteklius, paaiškėjo, kad eilučių ilgiai svyruoja nuo 1 ženklo iki 1041 ženklų. Reikšminių eilučių vidutinis eilutės ilgis yra 37 ženklai (5–8 žodžiai). 41% visų eilučių yra ne ilgesnės kaip 15 ženklų (1–3 žodžiai). 20% visų eilučių yra ne ilgesnės kaip 10 ženklų (vidutiniškai 1–2 žodžiai).

Panašūs skaičiai gauti atlikus mokymosi išteklių kūrimo bendradarbiaujant sistemos „Lemill“, veikiančios serveryje, lokalizuojamuosius išteklius.

Apibendrinti trijų programų lokalizuojamųjų išteklių eilučių ilgiai pateikiami 10 lentelėje.

10 lentelė. Eilučių ilgių pasiskirstymas internetinių programų lokalizuojamuose ištekliuose

Eilučių ilgis, ženklais	Mozilla Firefox	Moodle	Lemill	Vidurkis
≤ 10	33,6 %	20,8 %	23,6 %	26,0 %
≤ 15	47,6 %	41,1 %	44,0 %	44,2 %
≤ 20	60,0 %	37,0 %	59,7 %	52,2 %
≤ 25	66,2 %	66,0 %	68,6 %	66,9 %
≤ 30	72,0 %	72,0 %	74,4 %	72,8 %
≤ 35	75,4 %	76,3 %	78,2 %	76,6 %
Vidutinis ilgis, ženklais	29,7	37,0	30,2	32,3

Maždaug pusė visų lokalizuotinų eilučių yra iki 20 ženklų ilgio. Tai atitinka eilutes – frazes, kuriose yra nuo 1 iki 4 žodžių. Apie 20% visų lokalizuotinų eilučių yra iki 10 ženklų ilgio – tai 1–2 žodžių eilutė (žodis arba trumpa frazė). Skaičiai gauti remiantis trijų tirtų programų pavyzdžiais, bet panašus santykis būtų gautas ištyrus ir daugiau programų. Taigi lokalizuojamuose ištekliuose dominuoja trumpos eilutės, kuriose nėra konteksto, kuris padėtų taikliai jas išversti.

Lokalizuotojui būtų gerokai paprasčiau parinkti tinkamą eilutės atitikmenį, jeigu būtų žinomas kontekstas: vieta programoje, ar tai yra meniu komanda, ar užrašas ant mygtuko, ar dialogo lango pavadinimas, ar eilutė yra suduriama su kita (ir kokia yra ta kita eilutė), ką reiškia parametras eilutėje ir kt. informacija. Šias problemas iš dalies bandoma spręsti anksčiau nagrinėtu XLIFF failų formatu, tačiau šio formato specifikacijoje priemonės išsamesniam kontekstui nurodyti yra ribotos. Dėl to iškeliama idėja pasiūlyti išteklių pateikimo būdą, kuriame eilučių pateikimas atspindėtų ryšius su grafinės naudotojo sąsajos elementais, kuriuose jos naudojamos, o eilutės turėtų atributus, kurie suteiktų lokalizavimui naudingos informacijos ir padėtų gerinti lokalizacijų kokybę.

Lokalizuojamųjų išteklių kontekstas gali būti kelių lygių. Išskirsime du lygius:

1. Visos lokalizuojamos eilutės (aibės E elemento) kontekstas, kurio pavyzdžiai būtų:
   - Programos komponento, kuriame vartojama eilutė, pavadinimas;
   - Lango (tinklalapio), kuriame vartojama eilutė, identifikavimas;
   - Konkretus lango (tinklalapio) elementas, kuriame rodoma eilutė;
   - Ryšiai su kitomis eilutėmis, pvz., eilutės (frazės), grafinėje naudotojo sąsajoje esančios šalia duotosios eilutės;
   - Situacijos įvardijimas, kuriai esant eilutė pateikiama naudotojui (jei eilutė atsiranda naudotojo grafinėje sąsajoje dinamiškai);
   - Ar yra eilutė suduriama su kita eilute (ir su kuria eilute) formuojant vieną sakinį ar pranešimą, pateikiamą kompiuterio ekrane;
   - Valdantysis eilutės-frazės žodis;
   - Vidinė programos funkcija, kuri realizuoja eilute pavadintą komandą.
2. Lokalizuojamos eilutės dalies (frazės, parametro, sąvokos) kontekstas, kurio pavyzdžiai galėtų būti:
   - Eilutėje pavartoto parametro ryšiai su kitomis eilutės dalimis;
   - Eilutės dalių tarpusavio priklausomybė ir formų derinimas;
   - Konkrečių žodžių semantikos paaiškinimai (pvz., ar tai veiksmažodis, ar daiktavardis).

4.2. Atributinių gramatikų taikymas lokalizuojamiesiems ištekliams

Šiame skyriuje pateiksime minėtų problemų sprendimo metodą, paremtą formaliosiomis atributinėmis gramatikomis.

Metodu čia laikysime veiksmų būdų sistemą tikslui pasiekti. Metodo struktūra: 1) tikslas, 2) priemonės tikslui realizuoti, 3) veiksmų būdai.

Metodo tikslas: siekiant pagerinti programinės įrangos lokalizavimo kokybę, pateikti lokalizuojamuosius išteklius tarpiniu hierarchiniu pavidalu, išreiškiančiu lokalizuojamųjų eilučių ryšius su programos grafinės sąsajos elementais, eilučių tarpusavio ryšius, programos komandų semantiką.

Metodo priemonė: formaliosios atributinės gramatikos, papildytos naujomis priemonėmis.

Metodo veiksmų būdai aprašomi tolesniuose šios dalies skyriuose.

Pagrindinės priežastys, dėl kurių nebuvo pasirinka viena iš egzistuojančių grafinės sąsajos aprašo kalbų (šiuo metu daugėja XML pagrindu sukurtų daugiaplatformių sąsajos su naudotoju aprašo kalbų [SV03]) yra šios:

1. Grafinės sąsajos su naudotoju aprašo kalbos, kurios yra nepriklausomos nuo programavimo ar ženklinimo kalbų bei platformų, nėra specialiai pritaikytos internacionalizavimui ir lokalizavimui, t. y. jose nenumatyti atributai, kurie galėtų būti naudingi lokalizavimui [SV03].
2. Grafinės sąsajos su naudotoju aprašo kalbų naudojimas dar nėra pakankamai paplitęs (iš analizuotų internetinių programų tik „Mozilla“ šeimos programos naudoja savo atvirąją grafinės sąsajos aprašo kalbą XUL [BKO02].
3. Šiame darbe vengiama prisirišti prie konkretaus formato. Siekiama nepriklausomo formalaus išteklių struktūrizavimo ir pateikimo būdo, kuris tiktų taikyti prie įvairių šiuo metu egzistuojančių formatų. Sukurto formalaus pateikimo būdo konvertavimas į kurį nors iš formatų (pvz., XML), yra techninis konkrečios realizacijos dalykas.

Jeigu programoje būtų naudojama tam tikra grafinės naudotojo sąsajos aprašo kalba (ateityje sukurta arba viena iš esamų), tai formalią gramatiką kurti būtų paprasčiau, tektų esamą formalųjį grafinės naudotojo sąsajos aprašą papildyti lokalizavimui reikšmingais simboliais ir atributais, pateiktais šiame skyriuje, aprašyti semantikos taisykles.

Pradėsime nuo formaliųjų gramatikų ir atributinių gramatikų apibrėžimo ir programos išteklių dalių metainformacijos formalizavimo. Tolesniuose skyreliuose idėja išplečiama ir į kitas lokalizuojamųjų išteklių ir grafinės sąsajos sritis.

Lokalizuojamiems ištekliams ir jų metainformacijai formalizuoti galėtų tikti bekontekstės arba kontekstinės formaliosios gramatikos (11 lentelė).

11 lentelė. Gramatikos, kurios galėtų tikti lokalizuojamiems ištekliams pateikti

Gramatika	**Išvedimo taisyklių ($\alpha \to \beta$) pavidalas**	**Ribojimas**	**Atitikmuo algoritmų teorijoje**
Kontekstinė	$uAv \to uwv$	$\lvert\alpha\rvert \le \lvert\beta\rvert$	Tiuringo mašina
Bekontekstė	$A \to w$	$\lvert\alpha\rvert = 1$	Dėklas

Teoriškai mūsų uždavinys būtų išsprendžiamas naudojant kontekstines gramatikas. Tačiau jų realizacija yra daug sudėtingesnė negu bekontekstčių gramatikų. Programavimo kalbų sintaksei formaliai aprašyti praktiškai naudojamos bekontekstės gramatikos. Jos gerai ištirtos, sukurta daug jomis aprašytų programavimo kalbų.

Šiame darbe pasirinktos atributinės gramatikos, kurios yra bekontekstės gramatikų išplėtimas, t. y. įvedamų atributų ir semantikos taisyklių dėka gaunamas tarpinis variantas tarp kontekstinių ir bekontekstės gramatikų.

Atributinių gramatikų formalizmo pradinis tikslas buvo programavimo kalbų ir jų semantinių aspektų formalizavimas [Knu68], todėl atributinių gramatikų pagrindinė ir plačiausia tyrimų ir taikymų sritis – kompiliatorių projektavimas. Jos intensyviai nagrinėjamos ir koncepciniu, ir praktiniu požiūriais. Yra žinomi keli atributinių gramatikų poklasiai, turintys savitus realizacijos algoritmus. Taikant atributines gramatikas sukurta įvairių sistemų: metakompiliatorių, kompiliatorių, derintuvių. Kiekviena tokia sistema turi savo specifikavimo kalbą, kuri gali būti laikoma atributinės gramatikos dialektu.

Nepaisant to, kad atributinių gramatikų pradinė ir geriausiai ištirta taikymo sritis yra (tekstinės) programavimo kalbos, jų idėja gali būti kūrybiškai taikoma ir kitose srityse, kur yra svarbūs ryšiai tarp struktūrizuotos informacijos elementų [Paa95]. Pastarųjų dešimtmečių publikacijose pateikiami atributinių gramatikų sėkmingo taikymo pavyzdžiai tokiose srityse, kaip bendroji programinės įrangos inžinerija [ShK88; Fro92; LVS92], paskirstytasis programavimas [KK93], loginis programavimas [DM93], programų statinė analizė [HR92], duomenų bazės ir dokumentų užklausų formavimas [Nev00], vizualusis programavimas [HK87; CGP90], natūralios kalbos naudotojo sąsajos projektavimas [AGH90], aparatinės kompiuterių įrangos projektavimas [JS86], kompiuterių komunikacijos protokolai [BT89], naudotojo modelio kūrimas elektroninėse parduotuvėse [MP00], XML semantika [RLH98; PC99] ir kt.

Kadangi programinei įrangai lokalizuoti skirti ištekliai gali būti laikomi struktūrizuota informacija (pvz., paėmus struktūros pagrindu ryšius tarp eilučių ir programos grafinės naudotojo sąsajos elementų), tai dar vienas naujas atributinių gramatikų pritaikymas būtų lokalizuojamųjų išteklių struktūrizuoto semantinio pateikimo formato kūrimas.

Atributinė gramatika AG = <G, A, R> sudaroma iš trijų komponentų [Knu68]:

1. Bekontekstė gramatika G.
2. Baigtinė atributų aibė *A*.
3. Baigtinė semantikos taisyklių aibė *R*.

Čia bekontekstė gramatika G = <N, T, P, S>:

1. N – neterminalinių simbolių baigtinė aibė;
2. T – terminalinių simbolių baigtinė aibė (kalbos abėcėlė);
3. P – gramatikos taisyklių baigtinė aibė. Bendru atveju taisyklė išreiškiama pavidalu $X \to \alpha$, čia $X \in N$, $\alpha \in (N \cup T)^*$. V^* – aibė visų eilučių, sudarytų iš abėcėlės V simbolių, įskaitant ir tuščią eilutę (ε). Visų abėcėlės V eilučių, išskyrus tuščią, aibę žymėsime V^+. T. y. išvedimo taisyklės kairėje dalyje yra vienas neterminalinis simbolis, o dešinėje dalyje – terminalinių ir neterminalinių simbolių eilutė.
4. S ($S \in N$) – pradinis neterminalinis simbolis, nuo kurio pradedamas eilučių generavimas.

Elementas, priklausantis aibei $V = N \cup T$, vadinamas gramatikos simboliu.

Su kiekvienu gramatikos simboliu $X \in V$ susiejama baigtinė atributų aibė A(X). A(X) aibė dalijama į du poaibius: paveldėti atributai I(X) ir sintezuoti atributai S(X), taip kad $I(X) \cap S(X) = \varnothing$. Visų gramatikos atributų aibė $A = \cup A(X)$.

Tokia yra bendra AG apibrėžtis, kuri vienoda įvairiose atributinių gramatikų taikymams skirtose publikacijose, o sintezuotų ir paveldėtų atributų aibės bei semantikos taisyklių aibės įvairiuose šaltiniuose, skirtuose atributinėms gramatikoms ir jų taikymams nagrinėti,

apibrėžtys skiriasi. Originaliame Knutho AG apibrėžime [Knu68] laikoma, kad terminaliniai simboliai gali turėti paveldėtus atributus, tačiau negali turėti sintezuotų atributų. Ši taisyklė buvo pakeista daugumoje atributinių gramatikų realizacijose. Dažniausiai skirtingų autorių darbuose skiriasi pradinio ir terminalinių gramatikos simbolių paveldėtų ir sintezuotų atributų taisyklės.

Kai kurie autoriai be terminalinių ir neterminalinių simbolių įveda papildomų pagalbinių simbolių. Pavyzdžiui, [LRS74] pateikia atskirą atributinių gramatikų rūšį – atributinę transliacinę gramatiką.

Atributinių transliacinių gramatikų pagrindinė koncepcija yra *atributinis simbolis*. Atributinių simbolių aibė apibrėžiama naudojant baigtinę pagrindinių simbolių aibę, kiekvieno pagrindinio simbolio baigtines atributų aibes ir kiekvieno atributo reikšmių aibes (galbūt, begalines). Atributinis simbolis yra pagrindinio simbolio ir atributų reikšmių rinkinio visuma. Atributų reikšmės interpretuojamos kaip semantinė informacija, susieta su pagrindinio simbolio konkrečiu įėjimu.

Terminas *atributinė transliacija* reiškia tam tikrų atributinių įėjimo simbolių (įėjimo kalbos) grandinių atvaizdavimą į atributinių operacinių simbolių grandines. *Operacinis simbolis* – tai simbolis, žymintis tam tikrą semantinę operaciją. Paprasčiausiais atvejais tai gali būti tiesiog išvedimo operacija. Taigi operacinius simbolius galima laikyti išėjimo simboliais.

Atributinė transliacinė gramatika – tai bekontekstės gramatikos išplėtimas. Išplečiama dviem etapais: pirma bekontekstė gramatika išplečiama iki transliacinės gramatikos (be atributų), o po to pridedami atributai.

Transliacinė gramatika – tai bekontekstė gramatika, kurioje terminalinių simbolių aibė suskirstyta į *įėjimo simbolių* aibę ir *operacinių simbolių* aibę. Transliacinės gramatikos kalbos žodžiai vadinami *aktyviosiomis grandinėmis*. Transliacinės gramatikos *įėjimo gramatika* yra gramatika, gauta pašalinus iš gramatikos taisyklių visus operacinius simbolius.

Kiekvienos aktyviosios grandinės atitinkama operacinė dalis vadinama įėjimo dalies *transliacija*.

Gramatikos generuojamos kalbos kiekviena aktyvioji grandinė sugretina įėjimo dalį su operacine dalimi. Taip gaunamų visų porų aibė vadinama gramatikos *sintaksiškai valdoma transliacija*.

Praktiškai aktyvioji grandinė gali būti laikoma programa, kuri valdo kalbos apdorojimo įrenginį. Įėjimo simbolio įėjimą į aktyviąją grandinę galima (grubiai) interpretuoti kaip signalą įrenginiui, skirtą šiam simboliui skaityti. Operacinio simbolio įėjimą į aktyviąją grandinę galima interpretuoti kaip šio simbolio išvedimo operaciją. Kita operacinių simbolių interpretacija – tai semantinių programų, iškviečiamų apdorojant įėjimo seką, vardai.

Atributinė transliacinė gramatika – tai transliacinė gramatika, turinti papildomų savybių.

Pagal atributinių transliacinių gramatikų taisykles, apibrėžtas straipsnyje [LRS74]:

1. Kiekvienam įėjimo simboliui atitinka baigtinė aibė atributų; kiekvienas atributas įgyja reikšmę iš tam tikros (galbūt, begalinės) reikšmių aibės.
2. Kiekvienas neterminalinio simbolio arba operacinio simbolio atributas gali būti arba paveldėtas, arba sintezuotas.
3. Paveldėti atributai tenkina šias taisykles:
   3.1. Kiekvienam paveldėto atributo įėjimui į dešinę taisyklės dalį egzistuoja taisyklė (formulė), skirta apskaičiuoti jo reikšmei kaip funkcijai atributų kitų simbolių, įeinančių į kairę ar dešinę taisyklės dalis.

3.2. Kiekvienas pradinio simbolio (aksiomos) paveldėtas atributas turi pradinę reikšmę.

4. Sintezuotiems atributams taikomos šios taisyklės:

4.1. Kiekvienam neterminalinio simbolio sintezuoto atributo įėjimui į kairę taisyklės dalį egzistuoja taisyklė (formulė), skirta apskaičiuoti jo reikšmei kaip atributų funkciją kitų simbolių, įeinančių į kairę ar dešinę taisyklės dalis.

4.2. Kiekvienas operacinio simbolio sintezuotas atributas turi formulę, skirtą apskaičiuoti šio atributo reikšmę kaip kitų šio operacinio simbolio atributų funkciją.

Vėlesniuose straipsniuose dominuoja susitarimas, kad gramatikos pradinis simbolis ir terminaliniai simboliai neturi turėti paveldėtų atributų [Paa95; Reg09]. Pagrindinė priežastis, kodėl terminaliniams simboliams priskiriami tik sintezuoti atributai, – tai modulinis transliatorių projektavimas. Norint komponuoti didesnę gramatiką iš smulkesnių dalių yra svarbu, kad terminaliniai simboliai būtų nepriklausomi nuo konteksto, tuomet leksikos analizatoriai gali būti projektuojami nepriklausomai nuo kitų komponentų.

Laikysime, kad išvedimo taisyklė $p \in P$, $p: X_0 \rightarrow X_1 \ldots X_n$ $(n \geq 0)$ turi atributo $X_i.a$ *įeitį*, jeigu $a \in A(X_i)$, $0 \leq i \leq n$.

Semantikos taisyklių baigtinė aibė yra susieta su išvedimo taisykle *p*. Pateikiama po vieną taisyklę kiekvienam sintezuotam atributui $X_0.a$ ir po vieną taisyklę kiekvienam paveldėtam atributui $X_i.a$, $1 \leq i \leq n$. Taigi R_p yra taisyklių rinkinys, turintis pavidalą $X_i.a = f(y_1, \ldots, y_k)$, $k \geq 0$, čia:

- arba $i = 0$ ir $a \in S(X_i)$, arba $1 \leq i \leq n$ ir $a \in I(X_i)$;
- kiekvienas y_j, $1 \leq j \leq k$, yra atributo atvejis išvedimo taisyklėje p;
- f yra funkcija, vadinama semantikos funkcija, atvaizduojanti $y_1, \ldots, y_k$ reikšmes į $X_i.a$ reikšmę. Taisyklėje $X_i.a = f(y_1, \ldots, y_k)$ $X_i.a$ atributo įeitis priklauso nuo kiekvieno y_j atributo įeities, $1 \leq j \leq k$. $R = \cup R_p$.

Pagal šią apibrėžtį sintezuoti atributai yra simbolių, esančių išvedimo taisyklių kairėje pusėje, išvestis, o paveldėti atributai yra simbolių, esančių išvedimo taisyklių dešinėje pusėje, išvestis. (Laikoma, kad terminalinių simbolių sintezuoti atributai apibrėžiami išorėje.)

Struktūros *St*, generuotos gramatikos G, *išvedimo medis* – tai medis, kuriame:

- kiekvienas mazgas yra pažymėtas simboliu $X \in V$ arba tuščia eilute ε;
- medžio šaknis žymima pradiniu simboliu S;
- jei mazgas, pažymėtas X, turi dukterinių mazgų, pažymėtų $X_1, \ldots, X_n$, tai $X \rightarrow X_1, \ldots, X_n$ yra išvedimo taisyklė iš aibės P;
- lapų simboliai, sujungti iš kairės į dešinę, formuoja struktūrą *St* (programavimo kalbų projektavimo atveju – programa, programinės įrangos lokalizavimo atveju – išteklių visuma, išteklynas).

St struktūros *atributinis medis* – tai jos išvedimo medis, kuriame kiekvienas mazgas n, pažymėtas simboliu X, turi jam priskirtus atributų reikšmes, atitinkančias X atributus. Tai medis, gautas atlikus atributų skaičiavimo procesą.

Atributų skaičiavimas – tai procesas, kurio metu skaičiuojamos atributų reikšmės atributiniame medyje *T*, remiantis semantikos taisyklėmis *R*. T. y. atributo atvejo *n.a*, atitinkančio simbolio *Y* atributą *a*, reikšmė skaičiuojama pagal semantikos taisyklę $Y.a = f(y_1, \ldots, y_k) \in R_p$, čia arba (1) $a \in I(Y)$ ir *p* yra išvedimo taisyklė $X \rightarrow \ldots Y \ldots$, taikoma generuojant *T* iš *n*, arba (2) $a \in S(Y)$ ir *p* yra išvedimo taisyklė $Y \rightarrow \ldots$, taikoma generuojant *T* iš *n*.

Semantikos taisyklė vykdoma kreipiantis į funkciją *f* su argumentais – reikšmėmis tų atributų atvejų, kurie atitinka $y_1, ..., y_k$, ir priskiriant tos funkcijos reikšmę atributo atvejui *n.a*. Struktūros *St* prasmė yra sudaryta iš (sintezuotų) atributų atvejų, susietų su struktūros *St* šakniniu mazgu (toks principas taikomas klasikinėse atributinėse gramatikose).

Kadangi šiame darbe taikysime atributines gramatikas ne programavimo kalbos kompiliatoriui projektuoti, kur aukščiau minėtas reikalavimas yra svarbus, o lokalizuojamiesiems ištekliams su metainformacija pateikti, imant struktūros pagrindu programos grafinę naudotojo sąsają, tai originalaus atributinių gramatikų apibrėžimo reikalavimas, kad visa galutinė semantinė informacija renkama šakniniame medžio mazge, netenka prasmės. Dirbant su lokalizuojamaisiais ištekliais svarbus ne automatizavimas, o atributų priskyrimas ir jų pateikimas bei analizė išteklių lokalizavimo metu.

Atributinės gramatikos lokalizuojamiesiems ištekliams pateikti bus sudaromos tolesniuose skyreliuose, o prieš tai aptarsime bendruosius tokių gramatikų sudarymo principus.

4.2.1. Lokalizuojamųjų išteklių formalios gramatikos sudarymo principai

Ankstesniuose skyriuose (3.3.2 ir 3.7.2.2 sk.) minėjome, kad lokalizavimo metu nemažai problemų kelia parametrizuotos lokalizavimui skirtos teksto eilutės ir ekrane matomos eilutės, sudarytos sujungiant (suduriant) kelias išteklių eilutes. Dėl to sudarant gramatiką lokalizuojamųjų išteklių eilutę tikslinga išskaidyti į segmentus, kuriuos skiria parametrai (jei yra). Jeigu eilutės yra suduriamos, tai gramatikos medyje jos atsiranda greta: pateikiamos kaip visą eilutę įvardijančio simbolio mazgo žemesnio lygio mazgai. Dėl to reikalingas atskiras neterminalinis simbolis grafinės sąsajos elementui nurodyti ir atskiras neterminalinis simbolis visai eilutei nusakyti.

Lokalizuojamųjų išteklių formalizavimas pradedamas nuo bekontekstės gramatikos kūrimo. Žemiau pateiksime tokios gramatikos kūrimo veiksmus.

1. Bekontekstė gramatika G = <N, T, P, S> sudaroma konkrečiai programai, atspindint jos grafinės sąsajos struktūrą ir siejant ją su lokalizuotinomis eilutėmis.
2. Gramatika pagal jos simbolių naudojimą sudaroma iš dviejų pagrindinių dalių: programos grafinės naudotojo sąsajos struktūra ir lokalizuojamos eilutės ir jų struktūra.
3. Neterminalinių simbolių naudojimas:
    a. programos grafinės naudotojo sąsajos kiekvienam elementui įvardyti (pvz., mygtukui, išskleidžiamajam sąrašui, žymimajam langeliui) neterminaliniai simboliai parenkami remiantis programos grafinės naudotojo sąsajos specifika;
    b. parenkamas neterminalinis simbolis visai eilutei iš lokalizuojamųjų išteklių įvardyti;
    c. jei išteklių eilutėje yra pavartotas parametras, tai jam naudojamas neterminalinis simbolis, o išvedimo taisyklėje, kurios kairėje yra parametro neterminalinis simbolis, dešinėje pusėje yra vienas ar keli neterminaliniai simboliai, žymintys eilutes, skirtas įrašyti vietoje parametro, arba ε, jeigu parametro reikšmė skaičiuojama (gaunama) dinamiškai programos vykdymo metu. Lokalizuojamųjų išteklių eilutėje parametras paprastai žymimas taip, kaip priimta naudojamame lokalizuojamųjų išteklių atskyrimo metode arba programavimo kalboje, kuria parašyta programinė įranga, pavyzdžiui, %S, #1, #3, @vardas. Sudarant gramatiką, šis vardas pakeičiamas neterminaliniu simboliu, reiškiančiu parametrą.

d. Jei meniu ar kitame grafinės sąsajos elemente (valdiklyje) rodomas tekstas, suduriamas iš kelių eilučių, tai gramatikos išvedimo medyje jos atsiduria greta. Tam įvedami atskiri neterminaliniai simboliai grafinės sąsajos elementui ir visai eilutei.

4. Terminaliniai simboliai – tai lokalizuojamos eilutės ar eilučių dalys (čia eilutės dalis vadinsime segmentais). Jei eilutėje nėra parametrų, tai visa eilutė atitinka terminalinį simbolį. Jei eilutėje yra parametrų, tai eilutė skaidoma į segmentus, kuriuos skiria parametrai.

Sukūrus programos lokalizuojamiems tekstiniams ištekliams aprašyti skirtą bekontekstę gramatiką, tolesniais žingsniais ji yra išplečiama iki atributinės gramatikos:

5. Gramatikos simboliams priskiriami atributai, skirti lokalizavimo požiūriu svarbiai semantinei informacijai nusakyti. Visą eilutę atitinkančiam neterminaliniam simboliui priskiriamas atributas, apibūdinantis visą eilutę, t. y. eilutės aprašas. Atributų parinkimas remiasi lokalizavimo klaidų analize ir kalbos ypatumais.

6. Apibrėžiamos taisyklės kiekvieno gramatikos simbolio atributų reikšmėms skaičiuoti.

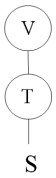
Pastebėsime, kad programos grafinė naudotojo sąsaja, kuria remiantis sudaroma atributinė gramatika, čia yra supaprastinama iki vaizdo, kuris veikiant programai bus rodomas kompiuterio ekrane. Į aprašą įtraukiami visi elementai, kurie nėra neutralūs lokalizavimo požiūriu. Šiame darbe nenumatomas išsamios grafinių elementų tarpusavio sąveikos aprašas ir sąveikos realizacijos priemonės, kadangi tai nėra reikšminga lokalizavimo požiūriu ir tik daro formalųjį aprašą sudėtingesnį. Tie sąveikos elementai, kurie teikia svarbios informacijos lokalizuotojams, yra įtraukiami per atributus, pavyzdžiui, išskleidžiamojo sąrašo elementų eilė, ryšys su komanda realizuojančia vidine funkcija. Į aprašą neįtraukiami tie elementai, kuriuose nėra vaizduojama lokalizuotina eilutė ar lokalizavimo požiūriu svarbi informacija, pvz., pelės grafinis žymeklis.

Tolesniuose skyreliuose plačiau aptarsime gramatikos simbolių parinkimą, gramatikos išvedimo taisyklių sudarymą ir išplėtimą iki atributinių gramatikų.

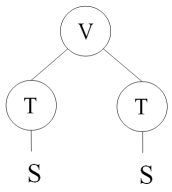
4.2.2. Grafinės sąsajos elementų įtraukimas į lokalizuojamųjų išteklių gramatiką

Lokalizuojamųjų išteklių ir jų metainformacijos formalus aprašas pradedamas kurti nuo bekontekstės gramatikos, kuri sudaroma remiantis grafinės sąsajos elementų struktūra, pagal kurią galima atsekti, kur lokalizuojamoji eilutė bus rodoma naudotojui programos grafinėje sąsajoje. Išskirsime pagrindinius atvejus, kaip lokalizuojamos eilutės gali būti rodomos grafinės naudotojo sąsajos elementuose (valdikliuose) – nuo to priklausys gramatikos simbolių parinkimas ir išvedimo taisyklių sudarymas. Vaizdumo dėlei naudosime grafinį gramatikos simbolių parinkimo ir struktūros vaizdavimą (gramatikos išvedimo medžio atvejų fragmentus), tolesniuose skyriuose gramatikos simbolių ryšius aprašysime išvedimo taisyklėmis.

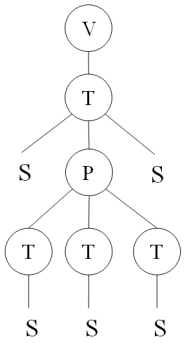
1. Ištisa eilutė paprastame valdiklyje. Žemiau pavaizduotas atvejis, kai eilutė be parametro (vienas segmentas S, mazgas V atitinka valdiklio neterminalinį simbolį, mazgas T – visos teksto eilutės neterminalinį simbolį).



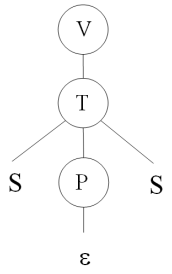
2. Viename paprastajame valdiklyje rodomos kelios sudurtos eilutės. Kiekviena tokia eilutė turi savo vardą lokalizuojamuose ištekliuose (t. y. eilutės pateikiamos atskirai). Toks būdas lokalizavimo praktikoje pasitaiko gana dažnai, bet yra nepageidautinas, nes gali sukelti klaidų lokalizuojant programas tam tikroms kalboms, ypač, kai reikia derinti suduriamų eilučių gramatines formas (žr. 3.7.2.2 sk.). Teoriškai galimas atvejis, kad suduriamos eilutės turės parametrų. Praktiškai toks atvejis pasitaiko retai, nes paprastai eilučių sudūrimas pakeičia parametro funkciją. Žemiau pateiktas dviejų suduriamų eilučių be parametrų atvejis.



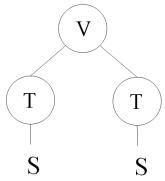
3. Lokalizavimui skirtoje eilutėje pavartotas vienas ar keli parametrai. Galimi atvejai:
   a. Vietoje parametro įrašoma kita eilutė iš lokalizuojamųjų išteklių, pvz., viena iš kelių galimų eilučių grupės. Laikysime, kad parametro vietoje įrašoma eilutė be parametro. Tuomet eilučių grupę, iš kurios programos vykdymo metu pasirenkama eilutė įrašyti vietoje parametro, žymėsime neterminaliniu simboliu P.



   b. Parametro vietoje įrašoma reikšmė, kurios nėra lokalizuojamuose ištekliuose, ji nežinoma iš anksto, bet įrašoma dinamiškai, programos vykdymo metu. Pavyzdžiui, kokių nors objektų skaičius, naudotojo vardas ir pavardė (iš registracijos duomenų bazės) ir pan. Tuomet reikalingas parametro atributas, suteikiantis informacijos lokalizuotojui, kokio tipo duomenys bus įrašomos vietoje parametro programos vykdymo metu. Žemiau pavaizduotas atvejis, kai eilutės viduje yra vienas parametras su dinamiškai parenkama reikšme.

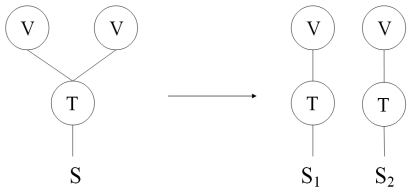


4. Valdiklyje rodoma viena iš kelių eilučių, priklausomai nuo konteksto, kuris išaiškėja vykdant programą.



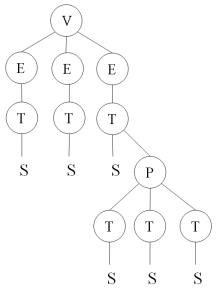
Toks būdas dažnai taikomas, kai programos kūrėjai bando išvengti parametrų naudojimo lokalizuotinose eilutėse: kiekvienam konteksto atvejui sukuriama atskira eilutė. Šis atvejis įtraukiant į gramatiką turi tokią pačią struktūrą, kaip ir 2 atvejis, todėl papildoma informacija, ar eilutės yra suduriamos, ar parenkamos dinamiškai rodant vieną iš jų, pateikiama naudojantis atributais.

5. Ta pati lokalizuojamųjų išteklių eilutė naudojama keliuose naudotojo grafinės sąsajos elementuose (pvz., mygtuko tekstas ir kontekstinio meniu elementas). Tokia eilutė būtų dubliuojama, lyginami atributai. Jei dubliuotų eilučių atitinkami atributai dera tarpusavyje, tai ištekliuose gali likti viena eilute. Jei atributai nedera, tai reiškia, kad viename valdiklyje reikia vienokio vertimo, kitame – kitokio. Šitaip aptinkamos internacionalizavimo klaidos. Jas galima ištaisyti į lokalizuojamus išteklius įtraukiant atskiras eilutes.



Šiuo atveju kuriama gramatika taip pat padėtų minimizuoti eilučių skaičių. Kartais programinės įrangos kūrėjai siekdami išvengti aukščiau minėtos internacionalizavimo klaidos įtraukia į išteklius per daug tos pačios eilutės egzempliorių. Sudarius atributinę gramatiką ir nagrinėjant atributus galima spręsti, ar reikia atskiro eilutės egzemplioriaus, ar ne.

6. Sudėtinis valdiklis (t. y. valdiklis, kuriame pagal jo paskirtį rodomos kelios eilutės), pavyzdžiui, išskleidžiamasis ar kitoks sąrašas. Programos meniu taip pat yra sudėtinis valdiklis, tačiau apie jį kalbėsime atskirai (jis yra sudėtingesnis, turi eilučių hierarchiją, savo lokalizavimo ypatumų). Nagrinėjant sudėtinio valdiklio kiekvieną paprastą elementą E, jam gali būti taikomas vienas iš aukščiau aprašytų atvejų. Žemiau pavaizduotas atvejis, kai sudėtinis valdiklis turi tris paprastus elementus (E), viename iš jų pavartota eilutė su parametru (3a atvejis).



Neterminalinis simbolis T, reiškiantis visą eilutę iš išteklių sistemos, įvestas laikantis vienodo bendro gramatikos sudarymo principo, kadangi teoriškai viename paprastajame

elemente (E) gali būti vaizduojamos kelios eilutės iš lokalizuojamųjų išteklių aibės (viena greta kitos arba kaip alternatyvos).

Programų, veikiančių klientų kompiuteriuose, grafine naudotojo sąsaja pateikiamos komandos paprastai turi prieigos klavišus (pvz., skirtą iškviesti komandą paspaudus tokį klavišą kartu su alternatyvos klavišu, veikia tik tam tikrame kontekste, pvz., tuo metu atvertame lange) bei komandos klavišus (kurį paspaudus su vienu ar daugiau kitais klavišais (pvz., valdymo) vykdoma tam tikra komanda, nesvarbu koks kontekstas, t. y. kuris langas tuo metu atvertas arba kur yra žymeklis), taip pat komandą paaiškinančią etiketę. Pavyzdžiui, naršyklės „Mozilla Firefox“ prieigos ir komandų klavišai sudaro 17% visų lokalizuojamųjų išteklių eilučių (4.1.1 sk.)

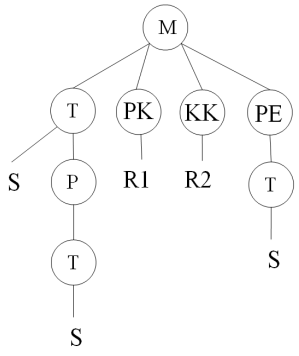
Prieigos ir komandų klavišams stengiamasi parinkti labiausiai įsimenamą raidę [DGJ08]. Pvz., prieigos klavišui geriausiai tinka pirmoji komandos ar meniu punkto pavadinimo raidė. Jeigu toje pačioje pakopoje yra keli meniu punktai, prasidedantys ta pačia raide, tai tada tenka pasirinkti kitą raidę: kito žodžio pirmąją (jeigu pavadinimas iš kelių žodžių), antrojo skiemens pirmąją raidę, kirčiuotą raidę ir pan. Vengiama raidžių, turinčių žemiau bazinės linijos išsikišusių dalių (ą, ę, g, į, j, ų), kad su jomis nesusilietų pabraukimo brūkšnys. Komandos klavišui taip pat geriausiai tinka arba pirmoji komandos raidė, arba originali raidė (jei toks klavišas jau yra prigijęs, pvz., kaip daugumoje tekstų rengyklių – *Vald* + *C* reiškia kopijavimą).

Kadangi lokalizavimo metu komandų pavadinimai (eilutės) keičiamos, tai turi būti pakeisti ir prieigos bei komandos klavišai. Todėl jie yra iškeliami į lokalizuojamuosius išteklius. Išanalizavus daugelio programų lokalizuojamuosius išteklius galima padaryti išvadą, kad šiuo metu naudojami du pagrindiniai tokių klavišų pateikimo ištekliuose būdai:

1. pateikiami kaip atskiros įvardytos eilutės (pvz., „Mozillos“ šeimos programos, pvz., *button.accesskey „E“*, *button.commandkey „E“*);
2. prieigos klavišai kaip nors pažymimi eilutės viduje, dažniausiai ženklu & (pvz., *&Edit image*), o komandos klavišai rašomi po komandos pavadinimo (pvz., *&Edit image <F4>*).

Valdikliai gali turėti paaiškinančiųjų etikečių (valdiklį paaiškinantis užrašas, parodomas atvedus ant valdiklio žymeklį). Lokalizuojamuose ištekliuose toks paaiškinimas pateikiamas kaip atskira eilutė.

Sudarant gramatiką siekiama, kad visos lokalizuojamųjų išteklių eilutės, susijusios su tam tikrų valdikliu būtų pateikiami taisyklėje taip, kad valdiklį žymintis neterminalinis simbolis būtų išvedimo taisyklės kairėje, o visą valdiklio pagrindinę eilutę, prieigos klavišą, komandos klavišą, paaiškinančiąją etiketę žymintys neterminaliniai simboliai būtų išvedimo taisyklės dešinėje. Tipinio mygtuko su lokalizuotinomis eilutėmis gramatikos išvedimo medžio fragmentas galėtų atrodyti taip:



Čia M – mygtukas, T – visa teksto eilutė (mygtuko atliekamos komandos pavadinimas), PK – prieigos klavišas (R1 – konkreti raidė, žyminti šį klavišą), KK – komandos klavišas (R2 – konkretus simbolis, žymintis tą klavišą), PE – paaiškinančioji etiketė, P – parametras, S – terminalinis simbolis (eilutės segmentas arba visa eilutė).

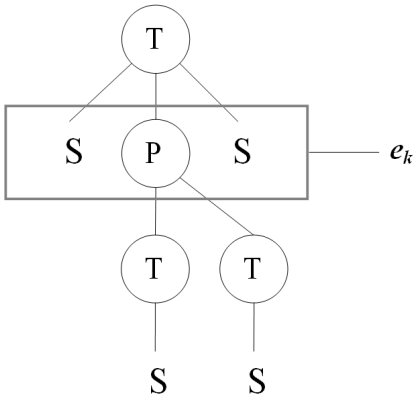
Prieigos klavišus, komandų klavišus ir paaiškinančias etiketes išskyrėme tam atvejui, kai programose jos numatytos ir naudojamos. Dalis internetinių programų tokių naudotojo sąsajos elementų neturi, tuomet struktūra būtų paprastesnė (be PK, KK, PE ir jų išvestinių mazgų).

4.2.3. Gramatikos simbolių ir lokalizuojamųjų išteklių elementų santykis

Tekstinių lokalizuojamųjų išteklių L eilučių aibę E galima suskirstyti į dvi dalis:

- tikrosios teksto eilutės, kurios skirtos rodyti ekrane (įskaitant eilutes su parametrais), žymėkime tokių eilučių aibę E_T;
- su lokale susijusias programos nuostatas apibrėžiančios eilutės (pvz., pirmoji savaitės diena, kurios reikšmė gali būti 0 (sekmadienis) arba 1 (pirmadienis); lango plotis ir aukštis; laiko formatas – 12 ar 24 valandų), žymėkime tokių eilučių aibę E_P. Tam tikrų programų atvejais $E_P = \varnothing$. Šios aibės elementų skaičius priklauso nuo programos projektuotojų pasirinkto sprendimo. Kai kurie autoriai laiko, kad iškelti programos nuostatas apibrėžiančias eilutes į lokalizuojamuosius išteklius, nėra saugu: lokalizavimo metu dėl nepakankamos informacijos galima nurodyti netinkamą nuostatos reikšmę, ir programa bus pažeista. Pavyzdžiui, mokymosi išteklių kūrimo bendradarbiavimo sistemoje „Lemill“ aibė $E_P = \varnothing$.

Atskiro neterminalinio simbolio visai lokalizuojamųjų išteklių tekstinei eilutei pažymėti įvedimas, žymėkime juos T_i, formuoja šių simbolių aibę, kuri atitinka lokalizuojamųjų išteklių eilučių aibės E poaibį E_T. Kalbant apie lokalizuojamųjų eilučių tekstines reikšmes, priimtas tik tas skirtumas, kad, dalis T elementų atitiks padubliuotus aibės E_T elementus (4.2.2 sk. 5 atvejis), o taip pat, jei eilutėje iš E_T aibės yra pavartotas parametras (kintamasis), tai vietoje jo faktinio vardo eilutėje (*%S*, *#1*, *#3*, *@vardas* ir pan.) naudojamas neterminalinis simbolis, reiškiantis parametrą, pvz., *P*. 21 paveiksle pavaizduotas gramatikos išvedimo medžio fragmentas, kuriame viršutinis T mazgas žymi visą eilutę, o stačiakampiu apvesti mazgai žymi atitinkamos eilutės $e_k \in E_T$ tekstą.



21 pav. Lokalizuojamos eilutės su parametru formalizavimas

Eilučių, priklausančių aibei E_P negalima tiesiogiai pamatyti kompiuterio ekrane, grafinės sąsajos elementuose (išimtis, kai su lokale susijusios programos nuostatos

parenkamos naudojantis tam skirtais programos grafinės sąsajos elementais). Todėl natūralu tokias eilutes atvaizduoti į gramatikos simbolių atributus. Pavyzdžiui, tam tikro dialogo lango plotį ir aukštį – kaip to lango neterminalinio simbolio atributus, pirmos savaitės dienos atributą – kaip visos programos pagrindinio lango atributą.

Kiekvieno simbolio T atributų, reiškiančių eilutės identifikatorių ($T.id$), aibė atitinka lokalizuojamųjų išteklių aibės L poaibį V. Bendru atveju $\{T.id\}$ ir V aibės elementai gali nesutapti, nes sudarant gramatika gali būti vartojamas ir išplėstas eilutės identifikatorius (pvz., kelias iki lokalizuojamųjų išteklių failo ir eilutės vardas), ne tik vardas, nurodytas lokalizuojamuose ištekliuose.

Taigi matome, kad tie duomenys, kurie pateikiami programinės įrangos tekstiniuose lokalizuojamuose ištekliuose atitinka tik vieno iš sudaromos atributinės gramatikos elementų aibę, to elemento vieno iš atributų aibę ir dalį atributų (jei yra) kitų gramatikos neterminalinių simbolių:

$$L = \{T.id\} \cup \{T\} \cup \{X_i.a_j\},$$

čia $X_i \in N$ (N – gramatikos neterminalinių simbolių aibė), $a_j \in A_{Xi}$ (A_{Xi} – simbolio X_i atributų aibė).

Visi kiti gramatikos simboliai ir atributai išplečia lokalizuojamuosius išteklius (laikant, kad jie pateikiami šiuo metu daugumos programinės įrangos gamintojų priimtu būdu, t. y. aibe $L = \{v, e\}$), įtraukia struktūrinę informaciją apie programos grafinę naudotojo sąsają, elementų semantiką ir eilutės dalis – segmentus bei parametrus.

Laikantis minėto neterminalinių simbolių parinkimo susitarimo, išteklių gramatikos išvedimo medžio šakos kiekvienas $n - 1$ mazgas (laikant, kad visa medžio šaka turi n mazgų) – tai T neterminalinis simbolis, atitinkantis visą teksto eilutę iš lokalizuojamųjų išteklių.

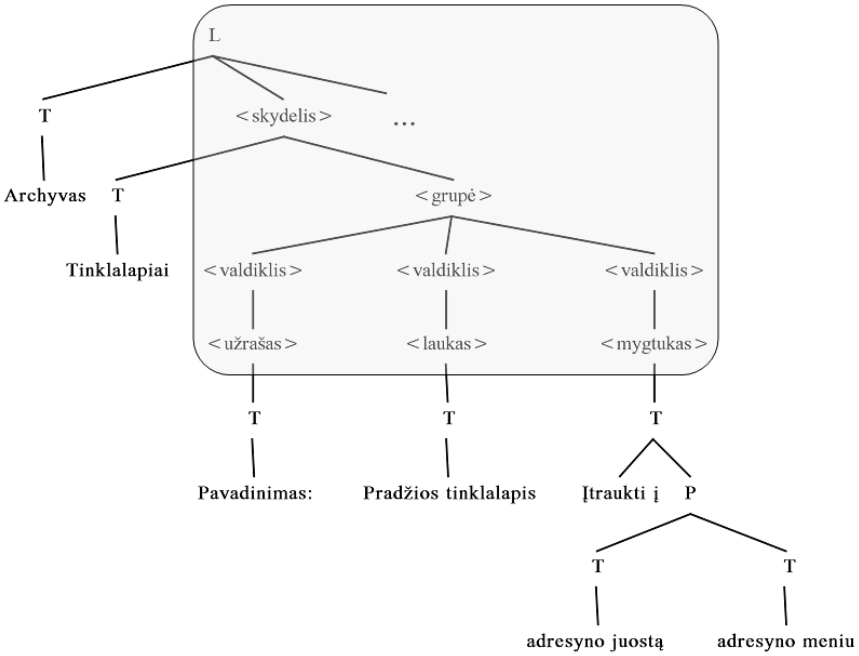
Jei gramatikos išvedimo medžio šakos $n - 2$ mazgas atitinka eilutės parametrą, tai lokalizuojant programą reikia nagrinėti $n - 3$ mazgą kartu su visais jo žemesnių lygių mazgais. Šis mazgas atitinka teksto eilutę, skirtą rodyti tam tikrame valdiklyje su parametru bei vietoje jo skirtomis įrašyti eilučių alternatyvomis.

Minėtu būdu sudarytos lokalizuojamųjų išteklių gramatikos išvedimo medį galima sąlyginai suskirstyti į 2 dalis:

- programos grafinės naudotojo sąsajos elementai;
- lokalizuojamos eilutės $\in E_T$ ir jų dalys.

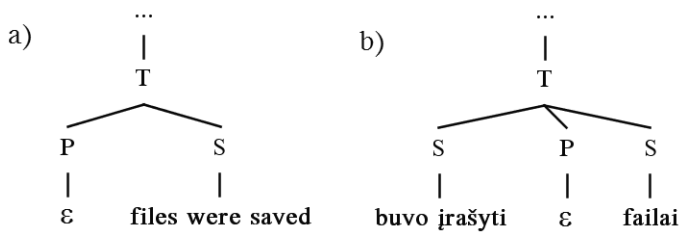
22 paveiksle pavaizduotas išvedimo medžio fragmentas gramatikos, atitinkančios hipotetinės internetinės programos dialogo lango dalį ir jame skirtas rodyti 7 eilutes iš lokalizuojamųjų išteklių. Viena iš eilučių – su parametru. Parametro vietoje įrašoma viena iš galimų dviejų eilučių, esančių lokalizuojamuose ištekliuose. Apvesta ir pilkai nuspalvinta išvedimo medžio fragmento dalis atitinka gramatikos programos grafinės sąsajos elementų dalį, o pusjuodžiu šriftu pateikti gramatikos simboliai, atitinkantys lokalizuojamųjų eilučių dalį. Aukščiau naudotus simbolius eilutės segmentui įvardyti S čia pakeitėme konkrečiais lokalizuojamųjų eilučių tekstais.

Gramatikos dalis, atitinkanti grafinės naudotojo sąsajos aprašą (22 pav. apvestas fragmentas) paprastai išlieka ta pati programinės įrangos originalui ir jos lokalizacijoms. Išimtis – kai lokalizuojant programinę įrangą modifikuojama ir jos grafinė sąsaja.



22 pav. Lokalizuojamųjų išteklių gramatikos struktūra: grafinės sąsajos ir eilučių dalys

Tolesnė gramatikos dalis (mazgai *T* ir jų visi žemesnio lygio mazgiai) yra nevienoda programinės įrangos originalui ir lokalizacijoms. Nuo simbolio T pradedamas kalbos keitimas. Lokalizuojant eilutę su parametru gali būti keičiama parametrų bei segmentų eilės tvarka, pavyzdžiui, eilutę „%S files were saved“ (laikykime, kad parametro reikšmė yra sveikasis skaičius, kuris priklauso intervalui [2; 5]) atitiktų *T* mazgo žemesnio lygio mazgų eilė *P S* (23 pav., a), o išverstą į lietuvių kalbą eilutę „buvo įrašyti %S failai“ atitiktų mazgo *T* žemesnio lygio mazgų eilė *S P S* (23 pav., b).



23 pav. Simbolių skaičiaus ir eilės tvarkos pasikeitimas lietuviškoje lokalizacijoje

Todėl lokalizacijos gramatikoje gali būti skirtumų, lyginant su originalia, pradedant T simboliu link išvedimo medžio lapų.

4.2.4. Gramatikos simbolių atributai ir semantikos taisyklės

Šiuo metu egzistuojančiuose tekstinių lokalizuojamųjų išteklių pateikimo formatuose, kaip jau minėjome 2.5 skyriuje, trūksta kontekstinės informacijos apie eilutes, kuri padėtų lokalizuotojams numatyti lokalizuotinos eilutės vietą programos grafinėje sąsajoje, taikytiną gramatinę formą ir kt. Pagrindinis metainformacijos šaltinis šiuo metu egzistuojančiuose formatuose – tai lokalizavimo komentarai. Tačiau lokalizuotinų eilučių komentavimo praktika yra labiau panaši į išimtį negu taisyklę. Pavyzdžiui, remiantis naršyklės „Mozilla Firefox“ lokalizuojamųjų išteklių analize, tik 4% visų eilučių turi lokalizavimo komentarus.

Kai kuriose programose dalį atributų galima išskirti iš *V* aibės elementų – vardo struktūros, pvz., „PageInfoTitle“, dalį – iš failų ir katalogų vardų (jie paprastai atspindi programos komponento pavadinimą, kuriame naudojama eilučių grupė), iš lokalizavimo komentarų. Tačiau tokias vardų schemas pasirenka ne visi programinės įrangos kūrėjai, be to net toje pačioje programoje vardų schema nėra sisteminga. Kai kurių programų *V* aibės elementai yra eilučių eilės numeriai, lokalizuotojui nesuteikiantys jokios informacijos apie kontekstą. Kiti formatai (pvz., *Gettext*) priimta lokalizuotinas eilutes identifikuoti ne atskiru vardu, o visos eilutės tekstu.

Dalį konteksto teikia suskirstymas eilučių į lokalizavimui skirtus failus (failas ar jų grupė atitinka tam tikros programų teminės dalies eilučių rinkinį). Tuomet failų pavadinimai gali teikti tam tikros informacijos apie kontekstą, paprastai apie programos komponentą, kuriame naudojamos eilutės, pavadinimą. Taip yra „Mozilla“ šeimos programose, VMA „Moodle“, tačiau kai kurių programų autoriai sudeda visas lokalizavimui skirtas eilutes į vieną failą (pvz., „LeMill“ aplinka). Taigi bendros taisyklės nėra.

Atlikus internetinės programinės įrangos pagrindinių kultūrinių elementų analizę ir pasirinkus tuos elementus, kurie yra vienaip ar kitaip atspindimi tekstiniuose lokalizuojamuose ištekliuose, galima išskirti bazinį sąrašą atributų, kuris yra svarbus sudarant programos lokalizuojamųjų išteklių atributinę gramatiką.

Dauguma pasaulyje paplitusios programinės įrangos sąsajos tekstų nuo pradžios rašoma anglų kalba, o tik vėliau lokalizuojama adaptuojant ją kitoms rinkoms. Todėl reikia numatyti atributus, kurie padėtų atspindėti kalbos, į kurią lokalizuojama, savybes, išreikštas taip, kad originalios programinės įrangos projektuotojas neturėdamas kitų kalbų, kurioms ruošiamasi lokalizuoti jo projektuojamą programinę įrangą, visapusiškų žinių, galėtų tokius atributus priskirti. T. y. atributus reikėtų reikšti neprisirišant prie kalbos, kuriai numatoma lokalizuoti, specifikos.

Rengiant atributų sąrašą vadovautasi internetinės programinės įrangos lokalizavimo lietuvių kalbai patirtimi [DGJ04; DJ09; Jev06] bei aptiktomis problemomis (2.7, 3.8 sk.). Buvo stengtasi parinkti pakankamai bendrus atributus, tinkančius daugumai indoeuropiečių fleksinių kalbų. O lietuvių kalba yra viena seniausių ir palyginti daug subtilumų turinti kalba Europoje. Todėl tie patys atributai tiks ir kitoms fleksinėms abėcėlinėms Europos kalboms, nes dauguma jų turi ne daugiau fleksijų negu lietuvių kalba.

Kontekstas, kurį vaizduoja gramatikos medis ir perduoda atributai, padeda tiksliau išversti ir adaptuoti programos originalo teksto eilutes. Pavyzdžiui, angliška eilutė „File“ galėtų būti verčiama vienaip, kai ji yra pagrindiniame meniu (pagal susitarimą galimas dažniau vartojamas variantas „Failas“, tačiau kai kuriuose specializuotose programose galimi ir kt. variantai, pavyzdžiui, „Dokumentas“ (jei tai dokumentų rengyklė), „Tinklalapis“ (jei tai naršyklė ar tinklalapių rengyklė) ir pan.), kitaip ši eilutė galėtų būti verčiama kitose programos vietose (pvz., tai gali būti veiksmažodis *file –> įtraukti į [aplanką]*, arba *failas*, net jeigu ir pagrindiniame meniu buvo pavartotas kitas šio meniu punkto vertimas).

Konteksto žinojimas taip pat leidžia išvengti nereikalingo žodžių kartojimo, pvz., meniu „Failas“ –> „Įrašyti failą“, „Failas“ –> „Atverti failą“ pakaktų pateikti taip: „Failas“ –> „Įrašyti“, „Failas“ –> „Atverti“.

Išskirkime universalius atributus, kurie gali būti taikomi įvairioms kalboms ir įvairioms programinės įrangos išteklių dalims lokalizuoti. Tolesniuose skyreliuose šie atributai bus papildyti specifiniais atributais, atspindinčiais konkretaus programos komponento grafinės sąsajos ir lokalizavimo savybes.

Daiktavardinė ar veiksmažodinė frazė. Šiuo atributu siekiama išvengti dažnos klaidos dėl daiktavardžių ir veiksmažodžių vienodos rašybos anglų kalboje. Ypač naudingas trumpoms eilutėms (frazėms ar net vieno žodžio eilutėms). Pavyzdžiui, „Open file“ arba

tiesiog „Open“ meniu kontekste būtų verčiamas „Atverti failą“ arba „Atverti“ (tai komandos pavadinimas, kuriam labiau priimtina veiksmažodinė forma), o dialogo lango pavadinime ta pati eilutė turėtų jau daiktavardinį atitikmenį „Failo atvėrimas“ arba „Atvėrimas“. Galima suformuluoti reikalavimus, kada vartojama daiktavardinė frazė (pvz., programos pagrindinio meniu juostoje – pirmojo lygio komandos, langų ir kt. objektų pavadinimai), o kada veiksmažodinė (pvz., mygtuko pavadinimas, meniu komandos, esančios gilesniame negu pirmojo lygio meniu). Šis atributas yra rekomendacinio pobūdžio, kai priskiriamas automatiškai.

Valdantysis eilutės ar segmento žodis. Tai žodis, nuo kurio priklauso kitų frazės žodžių formos. Atributas ypač naudingas trumpoms eilutėms (esant ilgoms – vieno ar kelių sakinių eilučių – šis atributas gali būti nevartojamas, kadangi kontekstą teikia pati eilutė). Pavyzdžiui, „Save file“ – „Įrašyti failą“ (valdantysis žodis yra „save“ – „įrašyti“), „New page“ – „Naujas tinklalapis“ (valdantysis žodis yra „page“ – „tinklalapis“), „Properties set“ – „Nuostatų rinkinys“ (valdantysis žodis yra „set“ – „rinkinys“), „Set properties“ – „Parinkti nuostatas“ (valdantysis žodis yra „set“ – „parinkti“). Pastarųjų dviejų pavyzdžių frazėse tiksliau parinkti vertimą padėtų veiksmažodinės ar daiktavardinės frazės atributas.

Eilutės aprašas. Visos teksto eilutės iš lokalizuojamųjų išteklių aprašas. Šis aprašas padeda geriau suprasti eilutės paskirtį programinėje įrangoje. Ypač naudingas tada, kai dėl grafinės sąsajos elemente skiriamos tekstui rodyti vietos trūkumo eilutė yra sutrumpinama arba pakeičiama kita, trumpesne, bet mažiau aiškia, ir kontekstas yra prarandamas. Projektuotojas žino, ką turi reikšti ta eilutė, tačiau lokalizuotojai tokios informacijos neturi. Atributas padėtų užfiksuoti eilutę paaiškinančią informaciją, kuri yra svarbi parenkant taiklų vertimą, o paprastai programos eilučių vertimas būna netiesioginis. Šis atributas taip pat yra labiau naudingas trumpoms eilutėms, tačiau reikalingas ir ilgesnėms eilutėms, nes gali padėti suprasti programos situaciją, kada ta eilutė yra rodoma (pvz., kuris nors retai pasitaikančios klaidos pranešimas).

Pirmoji eilutės raidė (didžioji ar mažoji). Anglų kalboje daugeliu atveju yra priimta sakinio viduje naudoti didžiąsias raides, pavyzdžiui, mėnesių ir dienų pavadinimus, antraštines frazes. Lietuvių kalboje jei tokie ar panašūs žodžiai yra sakinio viduryje, tai jie rašomi iš mažųjų raidžių. Didžioji raidė vartojama paprastai tik sakinio pradžioje, esant santrumpoms, tikriniam žodžiui arba pavadinimui, kuris paprastai papildomai rašomas kabutėse. Programos meniu tos pačios šakos gilyn einančios komandos dažnai yra susijusios tarpusavyje, gali formuoti vieną sakinį arba frazę. Tuomet pirmos komandos eilutę vertėtų pradėti didžiąja raidę, o susijusių komandų – mažosiomis, pavyzdžiui, meniu „File –> Save As –> Draft“ (čia rodyklės žymi perėjimą į žemesnio lygio meniu) lietuvių kalboje turėtų atitikmenį „Failas –> Įrašyti kaip –> juodraštį“. Atributo reikšmė gramatikoje gali būti apskaičiuojama automatiškai, pvz., pagal meniu lygį. Tuomet atributo reikšmė būtų rekomendacinio pobūdžio: jei meniu lygis yra lygus 3 ar gilesnis, tai komanda gali prasidėti iš mažosios raidės, o galutinį sprendimą priima žmogus, nes gali pasitaikyti tikrinis pavadinimas, kuris turi būti rašomas iš didžiosios raidės nepaisant meniu lygio.

Forma. Kadangi negalime tiesiogiai nurodyti linksnio atributo (dėl tokio linksnio atitikmens nebuvimo anglų kalboje), kalbos dalies ir kt. specifinių atributų, tai reikėtų kitaip išreikšto atributo, kuris galėtų padėti nusakyti arba iš dalies nusakyti žodžių gramatinę formą. Taigi žodžio ar frazės, arba eilutės, skirtos įrašyti parametro vietoje, forma gali būti apibūdinama klausimu, į kurį atsako: *who*, *what*, *whom*, *where*, *when* arba prielinksniu: *to*, *from*, *in*. *Who* ir *what* klausimus galima išskirti į dvi grupes: *subject* (veikiantysis objektas) ir *object* (veikiamasis objektas, atitiktų lietuvių kalbos galininko linksnį). Pavyzdžiui, sakinyje *the browser cannot display this page* „the browser“ yra subjektas (*what, subject*), o „this page“ – objektas (*what, object*). Atributo reikšmė ir būtų šis trumpas klausimas arba prielinksnis, kuris gali iš dalies nusakyti eilutės ar jos dalies linksnį ar formą. Atributas ypač

naudingas parametro simboliui, kai vietoje parametro reikšmė įrašoma programos vykdymo metu. Šiuo atributu nesiekiama tiksliai nusakyti bet kurį kalbos linksnį, bet pateikti informacija, kuri padėtų lokalizuotojui pasirinkti tinkamą formą, nesupainioti veikiantįjį objektą su veikiamuoju.

Terminas. Kaip jau minėjome, programose pasitaiko kompiuterijos terminų, kurie anglų kalba rašomi vienodai, bet turi skirtingas reikšmes. Tos reikšmės turi skirtingus atitikmenis kitose kalbose, pavyzdžiui, *call* – (1) skambutis, (2) kreipinys; *key* – (1) klavišas, (2) raktas, (3) kodas, (4) šifras; *tab* – (1) ąselė, (2) kortelė, (3) tabuliavimo klavišas, (4) tabuliavimo žymė; *release* – (1) laida, (2) atleisti, (3) išleisti; *manager* – (1) administratorius, (2) tvarkytuvė, (3) tvarkytojas it t. t. Norint vienareikšmiškai nurodyti reikšmės numerį, būtina turėti žodyną. Tai gali būti programoje vartojamų terminų žodynas arba bendras kompiuterijos žodynas, kuriuo sutarta vadovautis lokalizuojant programą. Tam, kad programos autorius galėtų sužymėti terminų reikšmes, žodynas turi būti verčiamasis-aiškinamasis. Žodyno sudarymas gali būti vienas iš lokalizavimo darbų etapų bendradarbiaujant programos autoriams su lokalizuotojais: prieš pradedant lokalizuoti suderinama terminija, verčiamas autoriaus parengtas pagrindinių terminų žodynas.

Atsižvelgiant į tai, kokiu žodynu sutarta naudotis nurodant termino reikšmės atributą, tuo pačiu atributu gali būti nusakyta ir kalbos dalis. Pavyzdžiui, enciklopediniame kompiuterijos žodyne [DGJ08] pateikiami ir kompiuterinių programų komandų pavadinimai, kurie turi veiksmažodinę formą, taip pat būdvardžiai, nusakantys tam tikrų objektų savybes. Dėl to šis žodynas nėra vadinamas terminų žodynu, jame aprašoma kompiuterijos leksika. Jei priskiriant atributų reikšmes būtų naudojamasi terminų žodynu, tai jame būtų mažiau atskirų terminų reikšmių, atitinkančių skirtingas kalbos dalis. Todėl būtų reikalingas ir anksčiau aptartas atributas, skirtas nusakyti, kokia yra frazė: daiktavardinė ar veiksmažodinė. Be to, kalbos dalį nusakantis atributas būtų reikalingas toms frazėms, kuriose nėra nevienareikšmių terminų, t. y. termino atributui nebus priskirta reikšmės iš sutartinio žodyno.

Vidinė funkcija. Dar daugiau kontekstinės informacijos suteiktų informacija apie vidinę funkciją, kuri realizuoja eilute iš lokalizuojamųjų išteklių pavadintą komandą. Vienas iš pavyzdžių galėtų būti toks. Įvairiose programose (originaluose) vartojami įvairūs tų pačių ar panašių komandų tekstiniai pavadinimai, pavyzdžiui, komandų grupė „OK“, „Done“, „Finish“, „Apply“, reiškianti atliktų veiksmų patvirtinimą ir išsaugojimą, ir „Cancel“, „Exit“, reiškianti atliktų veiksmų atsisakymą, išėjimą iš tam tikro lango. Žinant, kuri vidinė funkcija realizuoja komandą, lokalizacijoje galima įvesti vieningus terminus tą patį veiksmą atliekančioms komandoms, kurios anglų kalba vadinamoms skirtingai.

Eilutės identifikatorius (vardas) programos lokalizuojamųjų išteklių sistemoje. Atitinka lokalizuojamųjų išteklių aibės L poaibio V elementą. Jei eilutėje vartojamas parametras, o parametro vietoje gali būti įrašyta viena iš keleto statinių eilučių iš išteklių, tai šiame atribute gali būti kaupiamas visų tokių eilučių identifikatorių sąrašas. Jei parametro reikšmė gaunama dinamiškai, tai sąrašas tuščias. Jei vietoje parametro įrašoma viena fiksuota eilutė iš išteklių sistemos – sąrašas turi vieną identifikatorių. Atributas taip pat gali būti naudojamas tos pačios komandos tekstą formuojančių suduriamų eilučių identifikatoriams kaupti.

Plotis arba (ir) aukštis. Labai svarbus aspektas programinės įrangos lokalizavime, kuris gali mažinti lokalizuoto produkto kokybė, tai programos tekstų, skirtų rodyti ekrane, vietos ribojimai.

Paprastai ribojama grafinės sąsajos valdiklio pločiu ir aukščiu. Kai kuriose programose valdiklių plotis yra fiksuotas, kitos – automatiškai prasiplečiantis, prisitaikantis prie išverstos eilutės ilgio. Tačiau ne visos programos naudoją antrąjį sprendimą, be to net ir prasiplečianti eilutei skirta vieta turi savo ribas, kurias gali viršyti išversta eilutė. O kaip rodo tyrimai ir lokalizavimo patirtis, lokalizuotos eilutės pailgėja daugumoje kalbų. IBM įvertino ir

apibendrino teksto pailgėjimo tendencijas verčiant programinės įrangos naudotojo sąsajos tekstus iš anglų kalbos į įvairias Europos kalbas [IBM94] (12 lentelė):

12 lentelė. Vertimų į Europos kalbas ilgiai

Ženklų sk. tekste anglų kalba	Vidutinis vertimo ilgio didėjimas
Iki 10	200–300%
11–20	180–200%
21–30	160–180%
31–50	140–160%
51–70	130–140%
Daugiau kaip 70	150%

Kuo trumpesnė programos naudotojo sąsajos eilutė anglų kalba, tuo labiau ji pailgėja verčiant į kitas kalbas. Taigi tikslinga įvesti atributą, nusakantį plotį ir (arba) aukštį, skirta tekstui rodyti atitinkamame valdiklyje. Tokie atributai gali padėti ne tik matyti pločio (aukščio) ribojimus, bet ir automatiškai skaičiuoti išverstos eilutės ilgį bei pranešti apie vietos trūkumą, jei buvo parinktas per ilgas vertimas. Plotį ir aukštį patogu matuoti tam programinėje įrangoje priimtais matavimo vienetais (pvz., „em“, „ex“ – dažnai vartojami internetinėje programinėje įrangoje; pikseliais ar eilutės teksto ženklais). Tam tikrais atvejais teksto plotis nėra žinomas, pvz., kai tekstas įrašomas vietoje parametro dinamiškai, programai veikiant. Tuomet skaičiuojant, ar netrūksta vietos naudojama apytikslė reikšmė.

Sudurtinės eilutės rodiklis. Mūsų parinktame grafinės naudotojo sąsajos atvaizdavimo gramatikoje variante sudurtinės (eilutės, kurios rodomos kompiuterio ekrane greta ir formuoja vieną komandą, pranešimą, užrašą ir pan.) ir alternatyvios eilutės (kai viename valdiklyje vienu rodoma viena iš alternatyvių eilučių grupės) aprašomos vienodai (žr. ankstesnį skyrelį). Todėl verta įvesti atributą, kuris nusakytų, ar tam tikro grafinės sąsajos elemento tekstas, yra suduriamas iš kelių eilučių, ar ne. Šis atributas gali būti taip pat ir kaip pagalbinis komandos teksto užimamam pločiui skaičiuoti.

Duomenų tipas. Kai išteklių eilutėje pavartotas parametras, kurio vietoje programos vykdymo metu įrašoma tam tikra reikšmė (lokalizavimo metu ji nėra matoma), tai lokalizuotojui yra naudinga žinoti, koks tos reikšmės duomenų tipas, pavyzdžiui, teksto eilutė, asmens vardas, natūralusis skaičius, dešimtainė trupmena, data, laikas ar data ir laikas kartu. Jei parametro reikšmė yra dešimtainė trupmena, tai reikia atkreipti dėmesį į jos skirtuką (taškas ar kablelis), jei data, laikas ar data ir laikas, tai reikėtų atkreipti dėmesį į laiko ir datos formatus, laiko ir datos komponentų skirtukus. Parametro reikšmės duomenų tipo žinojimas lokalizavimo metu taip pat gali pagerinti eilutės, kurioje yra parametras, pagrindinio teksto vertimo kokybę. Jei duomenų tipas – skaitinis, tai reikėtų derinti šalia esančio daiktavardžio formą (pvz., 1 objektas, 2 objektai, 100 objektų). Jei yra tokių atvejų, programos autoriai turi pasirūpinti formų derinimo mechanizmo įtraukimu. Jei vietoje parametro įrašomas asmens vardas, tai reikėtų derinti šalia esančio dalyvio ar pusdalyvio giminę (pvz., Šiuo metu *Jūratė* yra *prisijungusi*, šiuo metu *Vytas* yra *atsijungęs*). Tam, kad lokalizuotojas įrašytų teisingą reikšmę, turi būti paruoštos tam skirtingos eilutės (pvz., du egzemplioriai eilutės *online*, du egzemplioriai eilutės *offline* iš anksčiau minėto pavyzdžio). Tačiau norint tinkamą formą parinkti, naudotojo registracijos sistemoje metu turi būti numatytas giminę nurodantis laukas (pvz., kreipinys: *Mr*, *Ms*), o pagal jo reikšmę automatiškai parenkama tinkamos giminės lokalizuota eilutė, skirta įrašyti prie asmens vardo.

Valdiklio tipas. Teksto eilutės vertimas gali priklausyti nuo valdiklio tipo, kuriame jis bus rodomas, pavyzdžiui, mygtuko pavadinimas dažnai būna veiksmažodinis (*enter new password – įvesti naują slaptažodį*), teksto užrašas (etiketė) turėtų jau kitokią formą (*enter new password – įveskite naują slaptažodį*), lango pavadinimas turėtų daiktavardinę formą (*enter new password – naujo slaptažodžio įvedimas*). Be abejo, valdiklio tipus ir jų

hierarchiją teikia formalios gramatikos išvedimo medžio mazgų pavadinimai ir struktūra (grafinės naudotojo sąsajos gramatikos dalis). Tačiau jie turi sintaksinę paskirtį, o semantiką perteikia atributai. Todėl verta įdėti valdiklio tipą kaip visos eilutės atributą.

Dialogo lango iškvietimo rodiklis. Žinojimas, ar tam tikra komanda, pavadinta atitinkama eilute iš lokalizuojamųjų išteklių, iškviečia dialogo langą ar ne gali padėti tiksliau išversti eilutę, parinkti tinkamą formą (dažniau – veiksmažodį), sistemingai laikytis susitarimo visoje programoje, kaip tokias komandas žymėti (pvz., daugelyje programų po tokias komandas įvardijančio teksto rašomas tritaškio ženklas). Dialogo lango iškvietimo rodiklis taip pat galėtų būti ženklas, kad reikia kartu nagrinėti ir kai kurias to dialogo lango eilutes, lyginti terminiją.

Prieigos ir komandų klavišų tikrinimas. Prieigos ir komandų klavišų temą aptarėme ankstesniame skyrelyje. Lokalizavimo metu tokių klavišų parinkimas kelia nemažai problemų. Visų pirma, reikia parinkti prasmingas raides (klavišus). Antra, jie neturi dubliuotis savo galiojimo srityje. Specialių atributų, skirtų kaupti lokalizuotų prieigos ir komandų klavišų sąrašui, įvedimas padėtų iš karto aptikti tokių klavišų lokalizavimo klaidas, tikrinti, ar nesidubliuoja tos pačios galiojimo srities lokalizavimo metu parinktas klavišas.

Čia išskyrėme tik bendriausius atributus. Atributinėje gramatikoje taip pat naudojami ir pagalbiniai atributai, ir kiti atributai, kurie priklausys nuo konkrečios programos ar jos komponento paskirties ir specifikos. Galimų atributų, jų reikšmių aibių ir aprašų sąrašai pateikiami tolesniuose skyriuose.

Priskyrus gramatikos simboliams atributus, apibrėžiamos jų semantikos taisyklės – funkcijos, priklausančios nuo kitų atributų reikšmių arba nuo reikšmių, pateikiamų iš išorės. Kai atributinės gramatikos taikomos programavimo kalbos sintaksei ir semantikai formaliai aprašyti, tai atributai skaičiuojami automatiškai pagal pradinių reikšmių aibę (kadangi tos funkcijos nurodo taisykles kompiliatoriui). Mūsų atveju, dirbant su lokalizuojamaisiais ištekliais, visiško automatizavimo siekti nėra tikslinga. Dalis semantikos taisyklių – tai atributo reikšmės priskyrimas iš išorės (pvz., juos priskiria programuotojas, internacionalizavimo ar lokalizavimo specialistas). Atributams apdoroti taip pat naudojamos papildomos išorinės funkcijos. Parinkus atributus ir aprašius jų semantikos funkcijas, patikrinama, ar atributų priklausomybės grafe nėra ciklų, t. y. ar atributo reikšmės rezultato skaičiavimas nepriklauso nuo pačios to atributo reikšmės.

Atributų parinkimas ir semantikos taisyklių (funkcijų) sudarymas pailiustruotas 4.3 skyriuje.

4.3. Lokalizuojamųjų išteklių apibendrintų atributinių gramatikų kūrimo atvejai

Sudaryti atributinę gramatiką, kuri būtų universali bet kuriai programinei įrangai, nėra tikslinga, kadangi kaip gramatikos struktūros pagrindas imami grafinės sąsajos elementai, vaizduojami ekrane konkrečiai programai veikiant ir priklausantys nuo programavimo kalbos, kuria parašyta programa, naudojamos programos ir jos grafinės sąsajos projektavimo priemonės. Todėl teoriniam nagrinėjimui pasirinkime keletą bendrų ir įvairioms programoms būdingų dalių bei sudarykime joms atributines gramatikas.

Šioje darbo dalyje remsimės ne konkrečių programų grafinių sąsajų pavyzdžiais, o programų pagrindinių grafinės naudotojo sąsajos elementų bendrosiomis schemomis, pateikiant apibendrintą galimos grafinės sąsajos elementų specifikaciją. Formaliai aprašant konkrečios programos lokalizuojamuosius išteklius ir jų metainformacija, užduotis būtų paprastesnė, kadangi būtų žinomi grafinės sąsajos elementų tipai, pasikartojimų skaičius bei konkretūs terminaliniai simboliai – lokalizuojamosios eilutės ir jų segmentai.

Formaliosios gramatikos taisyklės paprastai užrašomos dviem pagrindiniais būdais:

1. klasikiniu [Knu68], kai simbolių pasikartojimams realizuoti išvedimo taisyklėse naudojama rekursija, pvz.,

   S –> A

   A –> B A

   A –> B

2. iteraciniu, kai išvedimo taisyklių dešinėse pusėse naudojamos ciklinės konstrukcijos: reguliarieji reiškiniai virš gramatikos simbolių arba EBNF tipo reiškiniai, kur, pavyzdžiui, pakartojimas nulį ar daugiau kartų žymimas riestiniais skliaustais. Taip užrašytos atributinės gramatikos kai kurių autorių yra vadinamos išplėstosiomis atributinėmis gramatikomis [Nev00]. Aukščiau pateiktas išvedimo taisyklių pavyzdys antruoju būdu būtų užrašytas taip:

   S –> B {B}.

Antrasis užrašo tipas yra trumpesnis, tačiau išvedimo medis, generuotas pagal tokias taisykles, neturi iš anksto žinomo skaičiaus mazgų, pirmos taisyklės atveju išvedimo medžio mazgų skaičius žinomas, bet toks medis turi daug papildomų mazgų, kurie reikalingi tam tikrais atvejais, pvz., programavimo kalbai projektuoti, bet mūsų atveju, kai kalbama apie lokalizuojamuosius išteklius, tai neturi reikšmės, o tik pailgina aprašą. Iteracinis užrašas leidžia vaizdžiau pateikti viename lygyje esančius elementus (pvz., to paties lygio meniu komandas) ir taip sumažinant atributų skaičių, reikalingų vieno lygio elementų ryšiams nusakyti rekursinio tipo apraše. Be to, lokalizuojamos programinės įrangos visų elementų, aprašomų gramatika, skaičius yra baigtinis, taigi rekursinis užrašas nėra būtinas. Todėl toliau naudosime iteracinį tipą, kur gramatikos išvedimo taisyklių dešinėse pusėse naudojama EBNF sintaksė (2-asisis užrašo tipas).

Kadangi simbolių kartojimas žymimas specialiaisiais ženklais, pvz., { }, o ne simbolių eilute, tai šio užrašo tipo atveju reikia numatyti atributų grupę, taikytiną kiekvienai pasikartojančiai gramatikos simbolio įeičiai į taisyklę.

Sudarinėdami gramatikas laikysimės tokių žymėjimų susitarimų:

1. Ženklas | reiškia alternatyvą. Naudodami šį ženklą apjungsime išvedimo taisykles, kurių kairės pusės vienodos.
2. Ženklai [ ] žymi nebūtiną simbolių seką. Laužtiniuose skliaustuose [ ] esanti simbolių seka gali būti praleista, t. y. laužtiniai skliaustai reiškia simbolio ar simbolių įeities į išvedimo taisyklę nulį arba vieną kartą.
3. Simbolių grupė suskliaudžiama į paprastuosius skliaustus ( ).
4. Ženklai { } žymi simbolių kartojimą. Simbolių seka, kurią galima kartoti vieną arba daugiau kartų, rašoma į figūrinius skliaustus { }.
5. Ženklai **{ }**, užrašyti pusjuodžiu šriftu, žymi aibės elementų išvardijimą, pvz., neterminalinių simbolių aibė, išvedimo taisyklių aibė.

Atributinės gramatikos tam tikro simbolio atributas įvardijamas simbolio vardu ir atributo vardu skiriant juos tašku: *Simbolis.atr_vardas*, pavyzdžiui, *T.aprašas*.

Po kiekvienos išvedimo taisyklės pateikiamų semantikos funkcijų apibrėžimus rašysime pseudokodu. Kitus gramatikos užrašo žymėjimo susitarimus pateiksime toliau prieš juos tiesiogiai naudojant.

Atsižvelgdami į aukščiau pateiktus gramatikos bendrus sudarymo principus, sudarykime atributines gramatikas programos pagrindiniam meniu ir vienam iš galimų dialogo langų

(nuostatų langui). Kitiems programos elementams gramatikos sudaromos analogiškai, tik parenkami tuos elementus atitinkantys neterminaliniai simboliai.

4.3.1. Programos meniu atributinė gramatika

Dalis internetinių programų (pvz., naršyklių, el. pašto programų) lokalizavimui skirtų eilučių patenka į programos meniu. Pradėsime nuo jo. Meniu – tai dažnas programų komponentas, turintis savo lokalizavimo ypatumų, kuriuos įtrauksime į atributus.

Aprašykime meniu bekontekstę gramatiką $G_M = < N_M, \Im_M, \wp_M, M>$ ir papildykime ją atributais bei semantinėmis atributų skaičiavimo taisyklėmis, tokiu būdu pademonstruodami, kaip atributinės gramatikos gali būti taikomos lokalizuojamiems ištekliams ir jų metainformacijai formaliai pateikti.

$G_M = <N_M, \Im_M, \wp_M, M>$, čia

N = {M, K, P, T, PK, KK, PE} – neterminalinių simbolių aibė

$\Im$ = {S, R} – terminalinių simbolių aibė

$\wp$ = {M $\rightarrow$ K {K}

K $\rightarrow$ T {T} [PK] [KK] [PE] {K}

PK $\rightarrow$ R

KK $\rightarrow$ R

PE $\rightarrow$ T

T $\rightarrow$ (S | P) {(S | P)}

P $\rightarrow$ T {T} | ε

} – išvedimo taisyklių aibė

Čia naudojami tokie žymėjimai:

- M – meniu (gramatikos pradžios neterminalinis simbolis).
- K – meniu komanda.
- P – parametras.
- T – visa teksto eilutė, rodoma ekrane vienoje meniu komandoje ar viename valdiklyje (ištekliuose gali atitikti kelios eilutės dėl parametrų naudojimo).
- PK – prieigos klavišas.
- KK – komandos klavišas.
- PE – paaiškinančioji etiketė.
- S – teksto segmentas (lokalizuojamųjų išteklių eilutė arba jos dalis; vietoj S įrašoma konkreti eilutė).
- R – prieigos arba komandos klavišas; vietoj R įrašoma konkreti reikšmė.

Priskirkime gramatikai atributus. Naudojami atributai, kurie yra naudingi lokalizuojant meniu eilutes, išvardyti 13 lentelėje.

Pirmame ir paskutiname lentelės stulpeliuose pateikiami atributų vardų komponentai. Kadangi tie patys atributų vardai vartojami su įvairiais gramatikos simbolių vardais, tai lentelėje atributo vardas minimas tik vieną kartą (13 lentelės pirmas stuplelis), nurodant su kuriais gramatikos simboliais jis gali būti komponuojamas (13 lentelės paskutinis stuplelis). Toliau konkrečiose išvedimo taisyklėse šie vardų komponentai bus skiriami tašku.

Stulpelyje *Atributo rūšis* esanti rodyklė arba rodyklių pora žymi atributo *A* rūšį:

- ↑ – A ∈ S(A), sintezuojamas atributas;
- ↓ – A ∈ I(A), paveldimas atributas;
- ← – A ∈ E(A), *pirminis atributas*. Tai toks atributas, kurio reikšmė priskiriama iš išorės. Jeigu pirminis atributas perduodamas išvedimo medžiu aukštyn arba žemyn (t. y. priskyrus jam reikšmę iš išorės toliau atlieka sintezuojamo arba paveldimo atributo vaidmenį), tai jo rūšis žymima rodyklių pora: ↑← arba ↓←.

Pirminio atributo sąvokos nėra klasikiniuose atributinių gramatikų apibrėžimuose. Čia jį įvedėme, kadangi dirbant su lokalizuojamaisiais tekstiniais ištekliais dalį metainformacijos (atributų pradinių reikšmių) nurodo gramatikos sudarytojas. Be to, pirminis atributas gali būti priskiriamas ne tik prie gramatikos pradžios simbolio arba terminalinio simbolio, bet ir bet kurių kitų neterminalinių simbolių ir nebūtinai perduodamas išvedimo medžiu žemyn arba aukštyn.

Taigi mūsų atveju atributų aibė A(X) yra trijų aibių sąjunga:

$$A(X) = S(A) \cup I(A) \cup E(A),$$

čia S(A) – sintezuojamų atributų aibė, I(A) – paveldimų atributų aibė, E(A) – pirminių atributų aibė.

Trečiame lentelės stulpelyje apibūdinama atributo paskirtis, o ketvirtame – nurodomas atributo galimų reikšmių duomenų tipas.

13 lentelė. Meniu gramatikos atributų ir juos paaiškinančių aprašų sąrašas

Atributo vardas	**Atr. rūšis**	**Atributo aprašas**	**Atributo reikšmių tipas**	**Grama-tikos simbolis**
aprašas	↑←	Visos komandos arba teksto eilutės aprašas (ypač naudinga trumpoms arba sutrumpintoms eilutėms).	Teksto eilutė	K, T, PE, S
daikt_veiksm	↓←	Daiktavardinė ar veiksmažodinė frazė. Reikšmės ∈ {DAIKT, VEIKSM, NA}. Jeigu frazė nėra nei veiksmažodinė, nei daiktavardinė, tai atributo reikšmė neapibrėžta (NA). (Šiuo atributu siekiama išvengti dažnos klaidos dėl daiktavardžių ir veiksmažodžių vienodos rašybos anglų kalboje.)	Teksto eilutė	T, K, P, S
dialogas	↓←	Ar iškviečia dialogo langą. Jei iškviečia, tai dialogas = 1, jei neiškviečia, tai dialogas = 0.	Loginis	K, T
duom_tipas	↓←	Duomenų tipas. Tipas ∈ {ASM_VARDAS, TEKST, N_SKAIT, T_SKAIT, DATA, LAIK, DAT_LAIK}. Jei tipas yra T_SKAIT (dešimtainė trupmena), tai reikia atkreipti dėmesį į skirtuką, jei DATA (data), LAIK (laikas) ar DAT_LAIK (data ir laikas) – laiko ir datos formatus.	Teksto eilutė	P, S
eilutė	←	Lokalizuojamųjų išteklių atitinkama tekstinė eilutė arba eilučių sąrašas, skirtas meniu komandai įvardyti.	Teksto eilutė (teksto eilučių masyvas)	K
failo_vardas	←	Ištekliaus failo vardas, kuriame yra jo eilutės (g. b. sąrašas kelių failų).	Teksto eilutė (teksto eilučių masyvas)	M

Atributo vardas	**Atr. rūšis**	**Atributo aprašas**	**Atributo reikšmių tipas**	**Grama-tikos simbolis**
fakt_plotis	↑←	Eilutės ar jos segmento faktinis plotis pasirinktais matavimo vienetais (vienetai pasirenkami atsižvelgiant į atributo *plotis* matavimo vienetus). Tam tikrais atvejais teksto plotis nėra žinomas, pvz., kai tekstas įrašomas vietoje parametro dinamiškai, programai veikiant. Tuomet laikysime, kad *P.fakt_plotis = x* – apytikslė reikšmė.	Skaičius	M, K, T, P, S
forma	↓←	Žodžio ar frazės, skirtos įrašyti parametro vietoje forma, kuri apibūdinama klausimu, į kurį atsako (*who, what, whom, where, when*), veikiamasis ar veikiantysis objektas (*who (subject)*, *who (object)*, *what (subject)*, *what (object)*) arba prielinksniu: *to, from* ir kt. Atributo reikšmė ir yra šis trumpas klausimas ar žodis, kuris gali iš dalies nusakyti eilutės linksnį ar formą.	Teksto eilutė	P, S, T
funkcija	↓←	Vardas vidinės funkcijos programos pirminiame tekste, susietos su komanda.	Teksto eilutė	K, T
id	↑←	Meniu eilutės identifikatorius (vardas) programos lokalizuojamųjų išteklių sistemoje. Gali būti su keliu iki išteklių failo.	Teksto eilutė	K, T, PK, KK, PE, S
idsąrašas	↑	Vietoje parametro įrašomų eilučių identifikatorių sąrašas. Jei parametro reikšmė gaunama dinamiškai, sąrašas tuščias. Jei vietoje parametro įrašoma viena fiksuota eilutė iš išteklių sistemos – sąrašas turi vieną identifikatorių. Atributas taip pat naudojamas komandos eilučių identifikatoriams kaupti.	Teksto eilučių masyvas	P, K
KKsąrašas	↑	Meniu komandų klavišų sąrašas, skirtas kaupti viso meniu komandų klavišams ir tikrinti, ar jie nėra dubliuojami.	Teksto eilučių masyvas	K, M
klaida	↓	Klaidos rodiklis (pvz., tada, kai tas pats komandos klavišas pavartotas kelis kartus arba kai viršijamas leidžiamas elemento plotis).	loginis	KK, PK, M
meniu_gylis	↑	Meniu gylis, t. y. meniu komandų aibės maksimalus meniu lygis.	Natūralusis skaičius	M, K
meniu_lygis	↓	Komandos lygis meniu medyje. Nuo jo gali priklausyti komandos primoji raidė (didžioji ar mažoji), taip pat gali padėti išvengti tų pačių žodžių kartojimo gretimuose meniu lygiuose.	Natūralusis skaičius	K
meniu_tipas	↓←	Meniu tipas. Galimos reikšmės ∈ {kontekstinis, pagrindinis}.	Teksto eilutė	M, K, T
PKsąrašas	↑	Meniu prieigos klavišų sąrašas, skirtas kaupti prieigos klavišų sąrašui ir tikrinti, ar nėra dubliuojami to paties lygio meniu lokalizavimo metu parinkti klavišai.	Teksto eilučių masyvas	K

Atributo vardas	**Atr. rūšis**	**Atributo aprašas**	**Atributo reikšmių tipas**	**Grama-tikos simbolis**
plotis	↑←	Pagrindinei meniu juostai (komandoms) maksimali leidžiama vieta tam tikrais priimtais matavimo vienetais (pvz., „em“, „ex“, pikseliais, ženklais).	Skaičius	M, K
raidė	↓	Pirmoji raidė (didžioji ar mažoji), raidė = 1, jei didžioji, raidė = 0, jei mažoji. Apskaičiuojama pagal meniu lygį. Atributo reikšmė yra rekomendacinio pobūdžio. Jei meniu lygis yra lygus 3 ar gilesnis, tai komanda gali prasidėti iš mažosios raidės), tačiau taip ne visada turi būti – sprendžia žmogus, nes gali pasitaikyti tikrinis pavadinimas, kuris turi būti rašomas iš didžiosios raidės nepaisant meniu lygio.	Loginis	K, P
sudurtinė	↑←	Nurodo, ar tekstas, vaizduojantis meniu komandą, yra suduriamas iš kelių atskirų eilučių (sudurtinė = 1), ar ne (sudurtinė = 0). Naudojamas kaip pagalbinis atributas komandos teksto pločiui skaičiuoti.	Loginis	K
terminas	←	Nurodo nevienareikšmį terminą ir skliaustuose jo reikšmės numerį, pavyzdžiui, *Tab(1)* (tabuliacijos ženklas), *Tab(2)* (kortelė). Jei eilutėje yra keli tokie terminai, tai pateikiamas sąrašas skiriant sąrašo elementus kabliataškiu. Jei nėra nevienareikšmio termino, tai atributo reikšmė – tuščia eilutė. Terminų reikšmių numeriai imami iš sutarto verčiamojo-aiškinamojo žodyno, pvz., programos terminų žodyno.	Teksto eilutė	S
tikrinis	←	Nurodo, ar žodis, skirtas įrašyti vietoje parametro yra tikrinis pavadinimas, ar ne. Pagal tai sprendžiama, kokia turi būti pirmoji vietoje parametro skirto įrašyti teksto eilutės raidė: didžioji ar mažoji.	Loginis	P
vald_žodis	↓←	Valdantysis eilutės ar segmento žodis, pvz., „Įrašyti failą“ (vald_žodis = įrašyti), „Naujas tinklalapis“ (vald_žodis = tinklalapis). Esant ilgai eilutei (pvz., kelių sakinių eilutei), gali būti neįmanoma išskirti valdantįjį žodį, tuomet vald_žodis = NA. Pastebėsime, kad meniu atveju tokios ilgos eilutės paprastai nepasitaiko.	Teksto eilutė	T, P
reikšmė	↑←	Komandos arba prieigos klavišo simbolis (raidė).	Teksto eilutė	KK, PK
vietos_rodiklis	↑	Meniu komandai skiriamos vietos išnaudojimo rodiklis (nurodo, ar pakanka vietos, ar nepakanka).	loginis	M, K

Įvedus naują atributų tipą išplėčiamas atributinių gramatikų formalizmas. Klasikinių atributinių gramatikų teorijoje yra įrodyta, kad atributinės gramatikos semantikos taisyklės

yra korektiškos tada ir tik tada, kai nei vienas iš orientuotų atributų priklausomybės grafų neturi orientuotų ciklų [Knu68; Paa95].

1 teiginys. Įvedus pirminį atributą papildomų ciklų atributų priklausomybės grafe susidaryti negali, kadangi priskiriant atributo reikšmę iš išorės grafas papildomas lanku, ateinančiu į kurį nors grafo mazgą iš papildomo izoliuoto mazgo (išorės). Todėl semantikos taisyklių korektiškumo patikrinimas nesiskiria nuo klasikinių atributinių gramatikų.

Kitas atributinės gramatikos išplėtimas – tai atributų perdavimo srities valdymas, išreiškiamas atributų perdavimo ribojimu (atributų reikšmės nėra perduodamos visu atributiniu medžiu, o tik aktualių mazgų aplinkoje).

2 teiginys. Ribojant atributų perdavimo sritį papildomų ciklų atributų priklausomybių grafe susidaryti negali, nes dalis lankų išbraukiama, o naujų lankų neatsiranda. Todėl semantikos taisyklių korektiškumo patikrinimas nesiskiria nuo klasikinių atributinių gramatikų.

Žemiau pateiksime meniu gramatikos išvedimo taisykles (P_i), papildytas atributais ir semantikos funkcijomis (14 lentelė). Semantikos funkcijose atributai įvardijami 13 lentelės paskutiniame stulpelyje pateiktu gramatikos simbolio vardu ir pirmame stulpelyje pateiktu atributo vardu skiriant juos tašku.

Tie atributai, kuriems reikia priskirti pradines reikšmes (pirminiai atributai), pateikiami 15 lentelėje – anketoje. Tokią anketą užpildo gramatikos sudarytojas. Galimas užpildymas rankiniu būdu pasirenkant vieną iš kelių atributų reikšmių (kai reikšmių aibė iš anksto žinoma) arba nurodant reikšmę (kai reikšmės negalima prognozuoti iš anksto), galimas automatizavimas kai kurioms atributų reikšmėms gauti, pavyzdžiui, failo vardas ir kelias, teksto eilutės identifikatorius. Į anketą įtraukti visi atributai, kurie atributų sąrašo lentelėje pažymėtei ženklais ←,↑← arba ↓←.

Panašiai, kaip ir programavimo kalbų semantikos ir sintaksės formalizavimo praktikoje, išvedimo taisyklėse atributams skaičiuoti naudosime tokias pagalbines funkcijas (abibrėžiamas išoriškai):

1. *add(a, b)* – sąrašo *a* papildymas reikšme *b*.
2. *isin(list, x)* – tikrinimas, ar reikšmė *x* yra sąraše *list*. Rezultatas loginis: jei yra, tai 1, jei nėra – 0.
3. *error(t)* – klaidos pranešimo *t* pateikimas.
4. *length(t, m)* – teksto segmento *t* ilgis (plotis) pasirinktais matavimo vienetais *m*.

Kadangi išvedimo taisyklių dešinėse pusėse naudojame EBNF sintaksę, nurodant neapibrėžtą simbolių kartojimų skaičių, tai reikia numatyti, kaip bus skaičiuojami ir žymimi kiekvienos simbolio įeities atributai.

Laikysime, kad išvedimo taisyklės kairėje dalyje esantys simbolis atributų skaičiavimo taisyklėse žymimas nuliniu indeksu, pvz., K_0, o dešinėje išvedimo taisyklės dalyje esantys simboliai turi indeksus ne mažesnius kaip 1.

Jei gramatikos simbolis turi daugiau kaip vieną įeitį į išvedimo taisyklę (pvz., *K {K}*), tai visus simbolius K_1, K_2, ..., K_n žymėsime raidiniu indeksu, pvz., K_i, ir laikysime, kad i = 1, 2, ..., n, čia n – simbolio įeities į išvedimo taisyklę skaičius.

Užrašas pavidalo $X_i = b$ reiškia, kad visiems kartojamiems simboliams (X_1, X_2, ..., X_n), esantiems dešinėje išvedimo taisyklės dalyje, reikia priskirti reiškmę *b*. Toks susitarimas supaprastina užrašą, išvengiant ciklo konstrukcijų semantikos funkcijų aprašuose.

Semantikos funkcijų (atributų skaičiavimo ir priskyrimo taisyklių) aprašai pateikiami pseudokodu.

14 lentelė. Programos pagrindinį meniu atitinkančių lokalizuojamųjų išteklių atributinė gramatika

P_1:	**M → K {K}**
	$M.fakt\_plotis = \sum_{i=1}^{n} K_i.fakt\_plotis$
	M.vietos_rodiklis = M.plotis – M.fakt_plotis
	if M.vietos_rodiklis < 0 then M.klaida
	M.meniu_gylis = max(K_i.meniu_gylis)
	M.KKsąrašas:
	add(M.KKsąrašas, K_i.KKsąrašas)
	//kom. klavišų sąrašas jungimui su kt. pr. dalimis
	K_i.meniu_lygis = 1
	K_i.raidė = 1
	K_i.daikt_veiksm = DAIKT
	//pirmojo lygio meniu komandos – daiktavardžiai
	K_i.meniu_tipas = M.meniu_tipas

P_2:	**K → T {T} [PK] [KK] [PE] {K}**
	K_i.meniu_lygis = K_0.meniu_lygis + 1
	K_0.meniu_gylis = max{K_i.meniu_lygis}
	T_i.daikt_veiksm:
	if K_0.meniu lygis = 1 then T_i.daikt_veiksm = DAIKT
	K_i.raidė:
	if K_0.meniu_lygis + 1 > 2 then K_i.raidė = 0
	else K_i.raidė = 1
	K_0.aprašas = T.aprašas
	K_0.idsąrašas:
	add(K_0.idsąrašas, T_i.id)
	if K_0.sudurtinė then
	begin
	$K_0.fakt\_plotis = \sum_{i=1}^{n} T_i.fakt\_plotis$
	K_0.vietos_rodiklis = K_0.plotis – K_0.fakt_plotis
	end
	else
	begin
	K_0.fakt_plotis = Max(T_i.fakt_plotis)
	K_0.vietos_rodiklis = K_0.plotis – K_0.fakt_plotis
	end
	if K_0.vietos_rodiklis < 0 then K_0.error
	//ar nėra dubliuojami to paties meniu lygio prieigos klavišai:
	if isin (K_0.PKsąrašas, PK.reikšmė) then PK.klaida = 1 else
	add(K_0.PKsąrašas, PK.reikšmė)
	K_i.PKsąrašas = () //išvalomas žemesnio lygio komandų prieigos klavišų sąrašas
	K_0.KKsąrašas:
	add(K_0.KKsąrašas, K_i.KKsąrašas)
	if isin(K_0.KKsąrašas, KK.reikšmė)
	then KK.klaida = 1
	else add(K_0.KKsąrašas, KK.reikšmė)
	T_i.dialogas = K_0.dialogas
	T_i.funkcija = K_0.funkcija
	T_i.meniu_tipas = K_0.meniu_tipas

P_3:	**PK → R** PK.reikšmė = R

P_4:	**KK → R** KK.reikšmė = R
P_5:	**PE → T** PE.id = T.id
P_6:	**T → (S | P) {(S | P)}** $T.fakt\_plotis = \sum_{i=1}^{n} (S_i.fakt\_plotis + P_i.fakt\_plotis)$ $S_i.fakt\_plotis = length(S_i, mat\_vienetai)$ $S_i.id = T.id$ $P_i.vald\_žodis = T.vald\_žodis$ $P_i.daikt\_veiksm = T.daikt\_veiksm$

P_7:	**P → T {T} | ε**
	P.fakt_plotis = Max(T_i.fakt_plotis) | x //x – apytikslė reikšmė
	P.raidė:
	if P.tikrinis then P.raidė = 1 else P.raidė = 0
	P.idsąrašas:
	add(P.idsąrašas, T_i.id)
	T_i.duom_tipas = P.duom_tipas
	T_i.forma = P.forma

P_8:	**T → S** T.aprašas = S.aprašas T.fakt_plotis = length(S, mat_vienetai) S.id = T.id S.daikt_veiksm = T.daikt_veiksm S.forma = T.forma

Išvedimo taisyklė P_6 apima ir P_8 išvedimo taisyklės atvejį, tačiau čia pakartojome šį atvejį atskiros išvedimo taisyklės P_8 pavidalu, kad būtų patogiau atskirti P_8 atvejui būdingus atributus ir jų semantikos taisykles. P_8 taisyklė taikoma po P_7 ir gali būti taikoma prieš tai pritaikius P_2 arba P_5 išvedimo taisykles (jei teksto eilutė neturi parametrų, nėra sudurtinė ir neturi rodymo tame pačiame valdiklyje ar valdiklį paaiškinančioje etiketėje alternatyvių eilučių).

Žemiau pateikiamą anketą užpildo gramatikos sudarytojas. Stulpelyje „Atsakymas arba atsakymų variantai“ jis nurodo gramatikos atitinkamos išvedimo taisyklės (P_i) pirminio atributo reikšmę (jei langelis tuščias) arba pasirenka vieną iš galimų reikšmių. Paskutiniame stulpelyje pateikiami reikšmių pavyzdžiai bei pastabos, kurios gali padėti pastebėti galimą internacionalizavimo klaidą (pvz., formų derinimas, sudurtinių eilučių naudojimas).

15 lentelė. Pirminių atributų reikšmių priskyrimo anketa

Išv. tais.	**Atributo vardas**	**Klausimas**	**Atsakymas arba atsakymų variantai**	**Reikšmių pavyzdžiai, pastabos**
P_1	M.failo_vardas	Koks meniu lokalizuojamųjų išteklių failo vardas ir kelias? (Kelias – santykinis, nuo lokalizuojamųjų išteklių šakninio katalogo. Jei meniu ištekliai pateikiami keliuose failuose, jų keliai pateikiami atskiriant juos kabliataškiu.)		locale/browser/browser.dtd locale/browser/dialog.dtd; locale/browser/main.dtd
P_1	M.meniu_tipas	Koks meniu tipas?	Pagrindinis Kontekstinis	

Išv. tais.	**Atributo vardas**	**Klausimas**	**Atsakymas arba atsakymų variantai**	**Reikšmių pavyzdžiai, pastabos**
P_1	M.plotis	Koks meniu juostai ar kontekstiniam meniu skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 300
P_2	K_0.dialogas	Ar komanda iškviečiamas dialogo langas?	Taip Ne	
P_2	K_0.eilutė	Komandos teksto eilutė iš lokalizuojamųjų išteklių arba eilučių, skiriamų kabliataškiu, sąrašas.		Save As... Undo; Typing; Insert; Font change
P_2	K_0.plotis	Koks komandai skiriamas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 60
P_2	K_0.sudurtinė	Ar teksto eilutės T_i sudaro vieną frazę, ar skirtingas?	Taip Ne	*Pastaba*. Jei reikšmė „Taip“, tai yra potenciali internacionalizavimo klaida, kadangi lokalizuojant tokios eilutės komponentus nebus galima sukeisti vietomis. Reikėtų svarstyti sudurtinės eilutės keitimą viena (parametrizuota) eilute arba kelių eilučių alternatyvomis.
P_2	K_0.funkcija	Koks vardas vidinės programos funkcijos, susietos su meniu komanda?		manageAccounts()
P_3	PK.id	Koks prieigos klavišo identifikatorius (jei yra)? Jei nėra, tai palikite lauką tuščią.		Save.Accesskey
P_4	KK.id	Koks komandos klavišo identifikatorius (jei yra)? Jei nėra, tai palikite lauką tuščią.		Save.Commandkey
P_5	PE.aprašas	Koks eilutės aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		Hint for Save As menu command
P_6	S_i.aprašas	Koks kiekvienos eilutės segmento S_1, S_2, ..., S_n aprašąs? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas		Saves web page in a text file

Išv. tais.	**Atributo vardas**	**Klausimas**	**Atsakymas arba atsakymų variantai**	**Reikšmių pavyzdžiai, pastabos**
		paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		
P_6	T.aprašas	Koks eilutės aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		Undo previous action
P_6	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	VEIKSM DAIKT NA	*Pastaba*. Šio atributo reikšmė naudojama kaip numatytoji, bet gali būti keičiama kita taikant gramatikos semantikos taisykles.
P_6	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.		Save NA Page
P_6	T.id	Koks teksto eilutės identifikatorius?		BrowserSaveAs
P_6	S_i.terminas	Jei eilutės segmente pavartotas nevienareikšmis terminas, tai koks jis ir koks jo reikšmės numeris? Jei viename segmente yra keli tokie terminai, tai nurodykite juos skirdami kabliataškiu. Jei minėto tipo terminų nėra, palikite lauką tuščią.		call(1) certificate(2); key(4)
P_7	P.duom_tipas	Koks parametro duomenų tipas?	ASM_VARDAS TEKST N_SKAIT T_SKAIT DATA LAIK DAT_LAIK	*Pastaba.* Jeigu parametro reikšmė yra skaitinė (P.duom_tipas = N_SKAIT), tai reikia derinti toliau einančio segmento S formą (žr. pastabą po lentele). *Pastaba.* Jei P.duom_tipas = ASM_VARDAS, tai reikia numatyti šalia esančio segmento giminės (pvz., Jonas yra *prisijungęs*), kurią galima gauti iš naudotojo registracijos duomenų), linksnio (pvz., *Jono* laiškas) derinimą.
P_7	P.tikrinis	Ar vietoje parametro skirtas įrašyti tekstas yra tikrinis pavadinimas arba santrumpa?	Taip Ne	

Išv. tais.	Atributo vardas	Klausimas	Atsakymas arba atsakymų variantai	Reikšmių pavyzdžiai, pastabos
P_7	P.forma	Į kurį klausimą atsako parametro (galima) reikšmė? Pavyzdžiui, Who (subject)? Who (object)?, Where?, What (subject)? What (object)?, When?, Why? How many? ir t. t. Kuriuo prielinksniu galima apibūdinti žodžio formą?	Who (object) Who (subject) Where What (object) What (subject) When Why How many Whom To From In Kita:_____	
P_8	S.aprašas	Koks eilutės segmento, atitinkančio visą eilutę, aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		Saves web page in a text file
P_8	S.terminas	Jei eilutės segmente pavartotas nevienareikšmis terminas, nurodykite jį ir jo reikšmės numerį. Jei viename segmente yra keli tokie terminai, tai nurodykite juos skirdami kabliataškiu. Jei minėto tipo terminų nėra, palikite lauką tuščią.		call(1) certificate(2); key(4)
P_8	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	VEIKSM DAIKT NA	*Pastaba*. Šio atributo reikšmė naudojama kaip numatytoji, bet gali būti pakeičiama kita taikant gramatikos semantikos taisykles.
P_8	T.vald_žodis	Koks yra eilutės valdantysis žodis? (Jei yra, nurodykite NA).		Save Find NA
P_8	T.id	Koks teksto eilutės identifikatorius?		BrowserSaveAs

Pastaba. Derinant skaitvardžio ir daiktavardžio formas, naudojamos įvairių kalbų formų scenarijai, pvz., lietuvių turi 3 formas: n%10==1 && n%100!=11 ? 0 : n%10>=2 && (n%100<10 or n%100>=20) ? 1 : 2; arabų – 6 formas: n==0 ? 0 : n==1 ? 1 : n==2 ? 2 : n%100>=3 && n%100<=10 ? 3 : n%100>=11 && n%100<=99 ? 4 : 5.

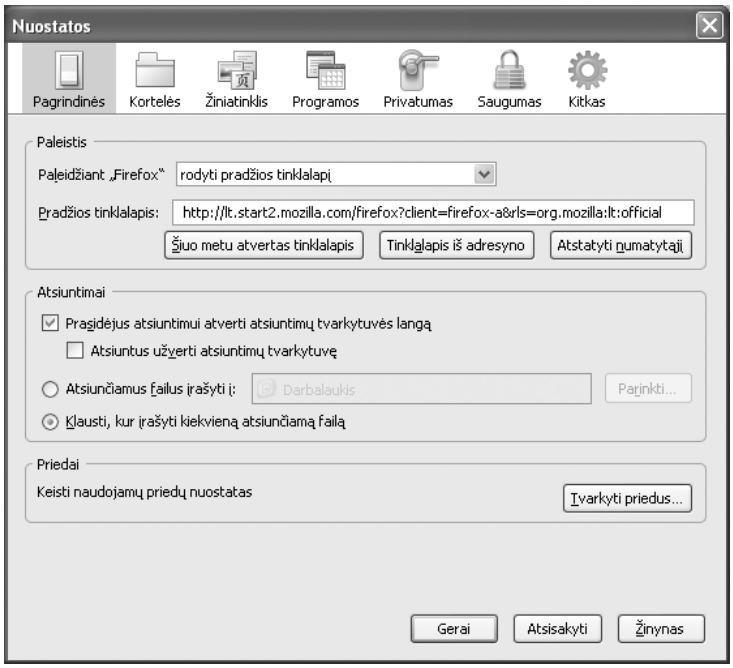
Analogiškai galima sudaryti kitų programos sąsajų elementų gramatikas, pvz., langų, tinklalapių (visų elementų, kuriuose vartojamos eilutės iš lokalizuojamųjų išteklių).

Sudaryti universalią gramatiką visiems sąsajos elementams aprašyti, tai reikštų sukurti naują grafinės naudotojo sąsajos aprašo kalbą (arba jos poaibį), papildytą lokalizavimo atributais, tačiau tai nėra šio darbo tikslas. Kiekvienos atskiros programos atveju galima

sudaryti tokią gramatiką, atsižvelgus į konkrečioje programoje pavartotus sąsajos elementus ir tekstus, skirtus rodyti ekrane. Be to, grafinės sąsajos elementų specifika priklausys nuo programos projektavimo priemonės ir programavimo kalbos.

4.3.2. Dialogo langa gramatika

Imkime tipinį nuostatų langą (24 pav.), kitus langus galima aprašyti analogiškai, pridėjus kitų, jiems būdingų elementų.



24 pav. Nuostatų lango pavyzdys (lokalizuota naršyklė „Mozilla Firefox“)

Lango bekontekstė gramatika galėtų būti tokia:

$G_L = <N_L, \Im_L, \wp_L, L>$, čia

N_L = {<L>, <skydelis>, <kortelė>, <grupė>, <h_grupė>, <v_grupė>, <valdiklis>, <tekst_užrašas>, <išskl_sąrašas>, <teksto_laukas>, <mygtukas>, <žym_langelis>, <žym_akutė>, <E>, <T>, S, <P>}

$\Im_L$ = {S}

$\wp_L$= {

<L> → <T> <skydelis> {<skydelis>} <mygtukas> {<mygtukas>}

<skydelis> → <T> <kortelė> {<kortelė>}

<skydelis> → <T> <grupė> {<grupė>}

<kortelė> → <T> <grupė> {<grupė>}

<grupė> → <T> <valdiklis> {<valdiklis>}

<grupė> → <T> <h_grupė> {<h_grupė >}

<grupė> → <T> <v_grupė> {<v_grupė >}

<h_grupė> → <valdiklis> {<valdiklis>}

<v_grupė> → <valdiklis> {<valdiklis>}

<valdiklis> → <tekst_užrašas>
<valdiklis> → <išskl_sąrašas>
<valdiklis> → <teksto_laukas>
<valdiklis> → <mygtukas>
<valdiklis> → <žym_langelis>
<valdiklis> → <žym_akutė>
<valdiklis> → ... // kiti valdiklių tipai, naudojami programoje
<tekst_užrašas> → <T>{<T>}
<teksto_laukas> → <T>{<T>} | ε
<mygtukas> → <T>{<T>}[<PK>][<KK>][<PE>]
<žym_akutė> → <T>{<T>}[<PK>][<PE>]
<žym_langelis> → <T>{<T>}[<PK>][<PE>]
<išskl_sąrašas> → <E>{<E>}
<PK> → R
<KK> → R
<PE> → <T>
<E> → <T>{<T>}
<T> → (S | <P>) {(S | <P>)}
<P> → T {T} | ε
}

Čia naudojami tokie žymėjimai:

- L – pradžios simbolis, žymintis langą.
- P – parametras.
- T – visa teksto eilutė, rodoma ekrane vienoje meniu komandoje ar viename valdiklyje (ištekliuose gali atitikti kelios eilutės dėl parametrų naudojimo).
- E – sudėtinio valdiklio (pvz., išskleidžiamojo sąrašo) paprastasis elementas.
- PK – prieigos klavišas.
- KK – komandos klavišas.
- PE – paaiškinančioji etiketė.
- S – teksto segmentas (lokalizuojamųjų išteklių eilutė arba jos dalis; vietoj S įrašoma konkreti eilutė).
- R – prieigos arba komandos klavišas; vietoj R įrašoma konkreti reikšmė.
- skydelis – dialogo lango dalis, tam tikrų nuostatų vienas skyrius, kuris gali turėti daugelį kortelių.
- kortelė – ąselę turinti lango dalis, kurioje rodomos tam tikros vienos rūšies nuostatos ar informacija.
- grupė – keli tarpusavyje susiję valdikliai, išdėstyti į lango stačiakampę sritį. Sritį ribojančiame kontūre paprastai užrašomas grupės pavadinimas.
- h_grupė – horizontaliai išdėstytų valdiklių grupė, paprastai be pavadinimo.
- v_grupė – vertikaliai išdėstytų valdiklių grupė, paprastai be pavadinimo.
- valdiklis – bet koks valdymo elementas grafinės sąsajos ekrane, pavyzdžiui, mygtukas, žymimasis langelis.
- tekst_užrašas – teksto užrašas, valdiklis tekstui užrašyti dialogo lange.

- išskl_sąrašas – išskleidžiamasis sąrašas, valdiklis, kuriame normaliai matomas vienas sąrašo elementas, o norint pamatyti kitus sąrašo elementus, reikia jį išskleisti.
- teksto_laukas – laukas (langelis) tekstui įrašyti.
- mygtukas – ekrane pavaizduotas mygtuko paveikslėlis, kurį paspaudus pelės klavišu vykdomas tam tikras veiksmas – komanda.
- žym_langelis – žymimasis langelis, mažas kvadratinis langelis, turintis dvi būsenas: pažymėtas arba nepažymėtas, kurio būsena nepriklauso nuo kitų šalia esančių žymimųjų langelių būsenos. Šalia langelio pateikiamas jo paskirtį apibūdinantis tekstas.
- žym_akutė – žymimoji akutė, mažas skritulėlis, turintis dvi būsenas: pažymėtas arba nepažymėtas, ir esantis tokių akučių grupėje, iš kurių tik viena gali būti pažymėta. Šalia akutės pateikiamas jos paskirtį apibūdinantis tekstas.

Neterminaliniai simboliai turi ilgesnius pavadinimus negu meniu gramatikos atveju, todėl juos rašome kampiniuose skliaustuose, kad būtų galima atskirti nuo terminalinio simbolio.

Visų atributų ir atitinkamų gramatikos simbolių sąrašas pateiktas 16 lentelėje, kurioje naudojami tie patys žymėjimo susitarimai, kaip ir 13 lentelėje. Taip pat galioja 1 ir 2 teiginiai iš 4.3.1 skyrelio.

16 lentelė. Nuostatų lango atributai, jų paaiškinimai ir atitinkami simboliai

Atributo vardas	**Atr. tipas**	**Atributo aprašas**	**Atributo reikšmių tipas**	**Gramatikos simbolis**
aprašas	↑←	Viso lango elemento arba teksto eilutės aprašas. Nurodomas, jei eilutė yra trumpa, eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas papildomas paaiškinimas.	Teksto eilutė	<L>, <skydelis>, <kortelė>, <grupė>, <h_grupė>,<v_grupė>, <valdiklis>, <tekst_užrašas>, <išskl_sąrašas>, <teksto_laukas>, <mygtukas>, <žym_langelis>, <žym_akutė>, <E>, <T>, S, <P>
ąselės_plotis	↑←	Kortelės pavadinimui užrašyti jos ąselėje skirtas maksimalus plotis pasirinktais matavimo vienetais.	Skaičius	<kortelė>
ąselės_vietos_ rodiklis	↑	Kortelės pavadinimui užrašyti ąselėje skiriamos vietos išnaudojimo rodiklis (nurodo, ar pakanka vietos, ar nepakanka).	loginis	<kortelė>
aukštis	↑←	Maksimali elementui skiriama vertikali vieta ekrane tam tikrais matavimo vienetais (pvz., „em“, „ex“, pikseliais, ženklais ir pan.).	Skaičius	<L>, <grupė>, <h_grupė>, <v_grupė>,
daikt_veiksm	↓←	Daiktavardinė ar veiksmažodinė frazė. Reikšmės ∈ {DAIKT, VEIKSM, NA}. Jeigu frazė nėra nei veiksmažodinė, nei	Teksto eilutė	<T>, <P>, S

Atributo vardas	Atr. tipas	Atributo aprašas	Atributo reikšmių tipas	Gramatikos simbolis
		daiktavardinė, tai atributo reikšmė neapibrėžta (NA).		
dialogas	↓←	Nurodo, ar mygtuku iškviečiamas dialogo langas. Jei iškviečia, tai dialogas = 1, jei neiškviečia, tai dialogas = 0. Jei dialogas = 1, tai šioje vietoje jungiamos dalinės atributinės gramatikos.	Loginis	<mygtukas>
duom_tipas	↓←	Parametro duomenų tipas. Tipas ∈ {ASM_VARDAS, TEKST, N_SKAIT, T_SKAIT, DATA, LAIK, DAT_LAIK}. Jei tipas yra T_SKAIT (dešimtainė trupmena), tai reikia atkreipti dėmesį į skirtuką, jei DATA (data), LAIK (laikas) ar DAT_LAIK (data ir laikas) – laiko ir datos formatus.	Teksto eilutė	<P>, S
eilė	↓←	Valdiklių eilės tvarka valdiklių grupėje: 1, 2, 3 ir t. t. Jei nepriskiriama, tai laikoma, kad eilė = 0. Naudojamas eilučių tarpusavio ryšiams nustatyti.	Skaičius	<valdiklis>, <tekst_užrašas>, <išskl_sąrašas>, <teksto_laukas>, <mygtukas>, <žym_langelis>, <žym_akutė>, <E>, <T>
eilutė	←	Lokalizuojamųjų išteklių atitinkama tekstinė eilutė arba eilučių sąrašas, skirtas rodyti grafinės sąsajos elemente.	Teksto eilutė (teksto eilučių masyvas)	<L>
elem_sk	↑	Išskleidžiamojo sąrašo elementų skaičius.	Natūralusis skaičius	<išskl_sąrašas>
failo_vardas	↑←	Ištekliaus failo vardas, kuriame yra jo eilutės (g. b. sąrašas kelių failų).	Teksto eilutė	<L>, <skydelis>
fakt_plotis	↑←	Išskleidžiamojo sąrašo, jo elemento, parametro, eilutės ar jos segmento faktinis plotis pasirinktais matavimo vienetais (vienetai pasirenkami atsižvelgiant į atributo *plotis* matavimo vienetus).	Skaičius	<išskl_sąrašas>, <E>, <T>, <P>, S
Fsąrašas	↑←	Lango ar skydelio atitinkamų lokalizuojamųjų išteklių failų vardų seka	Teksto eilučių masyvas	<L>, <skydelis>
forma	↓←	Žodžio ar frazės, skirtos įrašyti parametro vietoje forma, kuri apibūdinama klausimu, į kurį atsako (*who, what, whom, where, when*),	Teksto eilutė	<P>, <T> S

Atributo vardas	Atr. tipas	Atributo aprašas	Atributo reikšmių tipas	Gramatikos simbolis
		veikiamasis ar veikiantysis objektas (*who (subject)*, *who (object)*, *what (subject)*, *what (object)*) arba prielinksniu: *to, from* ir kt. Atributo reikšmė ir yra šis trumpas klausimas ar žodis, kuris gali iš dalies nusakyti eilutės linksnį ar formą.		
funkcija	↓←	Vardas vidinės funkcijos programos pirminiame tekste, susietos su mygtuko komanda.	Teksto eilutė	<mygtukas>, <T>
h_tarpas	↑←	Horizontalus tarpas tarp gretimų valdiklių pasirinktais matavimo vienetais. Naudojamas grupės pločiui skaičiuoti.		
id	↑←	Elemento idintifikatorius lokalizuojamųjų išteklių sistemoje.	Teksto eilutė	<PK>, <KK>, <PE>, <T>
idsąrašas	↑	Vietoje parametro įrašomų eilučių identifikatorių sąrašas. Jei reikšmė gaunama dinamiškai – sąrašas tuščias. Jei vietoje parametro įrašoma viena fiksuota eilutė iš išteklių sistemos – sąrašas turi vieną identifikatorių.	Teksto eilučių masyvas	<P>
PKklaida	↓	Atributas, skirtas pažymėti, ar nepadubliuotas prieigos klavišas. Jei dubliuojamas, tai klaida = 1, priešingu atveju klaida = 0.	Loginis	<skydelis>, <kortelė>, <grupė>, <h_grupė>, <v_grupė>,<valdiklis>, <mygtukas>, <žym_langelis>, <žym_akutė>, <PK>
KKklaida	↓	Atributas, skirtas pažymėti, ar nepadubliuotas komandos klavišas. Jei dubliuojamas, tai klaida = 1, priešingu atveju klaida = 0.	Loginis	<L>, <skydelis>, <kortelė>, <grupė>, <h_grupė>, <v_grupė>, <valdiklis>, <mygtukas>, <KK>
KK	↑	Komandos klavišas. Skirtas lokalizuotiems komandos klavišams perduoti siekiant tikrinti, ar nėra dubliavimo.	Teksto eilutė	<valdiklis>, <mygtukas>
KKsąrašas	↑	Pagalbinis atributas, skirtas kaupti komandų klavišų sąrašui ir naudojamas pranešti apie klaidą, jei tame pačiame lange dubliuojamas komandos klavišas.	Teksto eilučių masyvas	<L>, <skydelis>, <kortelė>, <grupė>, <h_grupė>, <v_grupė>
pav_plotis	↑←	Skydelio pavadinimui užrašyti skirtas maksimalus plotis	Skaičius	<skydelis>

Atributo vardas	**Atr. tipas**	**Atributo aprašas**	**Atributo reikšmių tipas**	**Gramatikos simbolis**
pav_vietos_rodiklis	↑	Skydelio pavadinimui užrašyti skiriamos vietos išnaudojimo rodiklis (nurodo, ar pakanka vietos, ar nepakanka).	Loginis	<skydelis>
PK	↑	Prieigos klavišas. Skirtas lokalizuotiems prieigos klavišams perduoti siekiant tikrinti, ar nėra dubliavimo.	Teksto eilutė	<valdiklis>, <mygtukas>, <žym_langelis>, <žym_akutė>
PKsąrašas	↑	Pagalbinis atributas, skirtas kaupti prieigos klavišų sąrašui ir naudojamas pranešti apie klaidą, jei to paties lango skydelyje ar kortelėje dubliuojamas prieigos klavišas.	teksto eilučių masyvas	<skydelis>, <kortelė>, <grupė>, <h_grupė>, <v_grupė>
plotis	↑←	Maksimali elementui skiriama horizontli vieta ekrane tam tikrais matavimo vienetais (pvz., „em“, „ex“, pikseliais, ženklais ir pan.).	Skaičius	<L>, <T>, <grupė>, <h_grupė>, <v_grupė>, <valdiklis>, <tekst_užrašas>, <išskl_sąrašas>, <teksto_laukas>, <mygtukas>, <žym_langelis>, <žym_akutė>
raidė	↑←	Pirmoji parametro ar visos eilutės reikšmės raidė: didžioji (1) ar mažoji (0).	Loginis	<P>, <T>
ryšys_idsąrašas	↓←	Susijusių teksto eilučių (ekrane rodomų šalia, kurios sudaro vieną sakinį, bet pavaizduotos skirtinguose valdikliuose)	Teksto eilučių masyvas	<P>, <T>, S
terminas	←	Nurodo nevienareikšmį terminą ir skliaustuose jo reikšmės numerį, pavyzdžiui, *Tab(1)* (tabuliacijos ženklas), *Tab(2)* (kortelė). Jei eilutėje yra keli tokie terminai, tai pateikiamas sąrašas skiriant sąrašo elementus kabliataškiu. Jei nėra nevienareikšmio termino, tai atributo reikšmė – tuščia eilutė. Terminų reikšmių numeriai imami iš sutarto verčiamojo-aiškinamojo žodyno, pvz., programos terminų žodyno.	Teksto eilutė	S
tikrinis	←	Nurodo, ar žodis, skirtas įrašyti vietoje parametro yra tikrinis pavadinimas ar	Loginis	<P>

Atributo vardas	Atr. tipas	Atributo aprašas	Atributo reikšmių tipas	Gramatikos simbolis
		santrumpa, ar ne. Pagal tai sprendžiama, kokia turi būti pirmoji vietoje parametro skirto įrašyti teksto eilutės raidė: didžioji ar mažoji.		
Tsąrašas	↑	Pagalbinis atributas, skirtas kaupti susijusių grafinės naudotojo sąsajos elementų grupės teksto eilučių identifikatorių sąrašui. Naudingas todėl, kad šalia esančių valdiklių tekstų vertimai gali priklausyti vienas nuo kito.	teksto eilučių masyvas	<grupė>, <h_grupė>, <v_grupė>, <valdiklis>, <tekst_užrašas>, <išskl_sąrašas>, <teksto_laukas>, <mygtukas>, <žym_langelis>, <žym_akutė>, <E>
v_tarpas	↓←	Vertikalus tarpas tarp grupės gretimų valdiklių pasirinktais matavimo vienetais. Naudojamas grupės aukščiui skaičiuoti.	Skaičius	v_grupė
v_tipas	↓	Valdiklio tipas – jo pavadinimas. Padeda tiksliau išversti tekstą, kai žinoma, kuriame valdiklyje jis bus rodomas.	Teksto eilutė	<T>, <P>, S
vald_žodis	↓←	Valdantysis eilutės žodis, pvz., „Parinkti nuostatas“ (vald_žodis = parinkti), „Naujas tinklalapis“ (vald_žodis = tinklalapis). Esant ilgai eilutei (pvz., kelių sakinių eilutei), gali būti neįmanoma išskirti valdantįjį žodį, tuomet vald_žodis = NA.	Teksto eilutė	<T>, <P>
reikšmė	↑←	Komandos arba prieigos klavišo simbolis (raidė).	Teksto eilutė	<PK>, <KK>
vietos_rodiklis	↑	Atitinkamo valdiklio tekstui skiriamos vietos išnaudojimo rodiklis (nurodo, ar pakanka vietos, ar nepakanka).	loginis	<valdiklis>, <tekst_užrašas>, <išskl_sąrašas>, <teksto_laukas>, <mygtukas>, <žym_langelis>, <žym_akutė>

Minėtas nuostatų langas gali būti aprašytas žemiau pateikta atributine gramatika. Išvedimo taisyklės (P_i), papildytos atributais ir semantikos funkcijomis, pateikiamos 17 lentelėje. Laikysimės tų pačių susitarimų, kaip meniu atributinės gramatikos atveju. Be meniu gramatikos papildomų funkcijų (žr. ankstesnį skyrelį) naudosime dar vieną funkciją:

count(x_i) – x_i elementų kiekio skaičiavimas. Rezultatas – x_i elementų kiekis. Naudojama sudėtinio valdiklio paprastiems elementams skaičiuoti.

Tie atributai, kuriems reikia priskirti pradines reikšmes (pirminiai atributai, 16 lentelė), pateikiami 18 lentelėje – anketoje. Tokią anketą užpildo gramatikos sudarytojas. Galimas

užpildymas rankiniu būdu pasirenkant vieną iš kelių atributų reikšmių (kai reikšmių aibė iš anksto žinoma) arba nurodant reikšmę (kai reikšmės negalima prognozuoti iš anksto), galimas automatizavimas kai kurioms atributų reikšmėms gauti.

17 lentelė. Programos dialogo lango atitinkančių lokalizuojamųjų išteklių atributinė gramatika

P_1	**<L> → <T> <skydelis> {<skydelis>} <mygtukas> {<mygtukas>}**
	L.aprašas = T.aprašas
	T.dait_veiksm = DAIKT
	L.Fsąrašas:
	add(L.Fsąrašas, $skydelis_i$.failo_vardas)
	$<skydelis>_i$.PKsąrašas = ()
	// išvalome skydelių prieigos klavišų sąrašą
	if $skydelis_i$.KKsąrašas ≠ ∅ OR $mygtukas_i$.KK ≠ ε then
	if isin(L.KKsąrašas, $skydelis_i$.KKsąrašas) then
	begin
	L.KKklaida = 1
	$skydelis_k$.KKklaida = 1
	// k žymi konkretų skydelį, iš kurio atėjo klaida
	end
	if isin(L.KKsąrašas, $mygtukas_i$.KK) then
	begin
	L.KKklaida = 1
	$mygtukas_k$.KKklaida = 1
	// k žymi mygtuką, iš kurio atėjo klaida
	end
	else
	add(L.KKsąrašas, $skydelis_i$.KKsąrašas ∪ $mygtukas_i$.KKsąrašas)

P_2	**<skydelis> → <T> <kortelė> {<kortelė>}**
	skydelis.aprašas = T.aprašas
	T.dait_veiksm = DAIKT
	skydelis.pav_vietos_rodiklis = skydelis.pav_plotis – T.fakt_plotis
	if skydelis.pav_vietos_rodiklis < 0 thenerror(„trumpinti“)
	$<kortelė>_i$.PKsąrašas = ()
	// išvalome kortelių prieigos klavišų sąrašą
	if $kortelė_i$.KKsąrašas ≠ ∅ then
	if isin(skydelis.KKsąrašas, $kortelė_i$.KKsąrašas) then
	begin
	skydelis.KKklaida = 1
	$kortelė_k$.KKklaida = 1
	// k žymi konkrečią kortelę, iš kurios atėjo klaida
	end
	else
	add(skydelis.KKsąrašas, $kortelė_i$.KKsąrašas)

P_3	**<skydelis> → <T> <grupė> {<grupė>}**
	skydelis.aprašas = T.aprašas
	T.dait_veiksm = DAIKT
	skydelis.pav_vietos_rodiklis = skydelis.pav_plotis – T.fakt_plotis
	if skydelis.pav_vietos_rodiklis < 0
	then error(„trumpinti“)
	if $grupė_i$.PKsąrašas ≠ ∅ then
	if isin(skydelis.PKsąrašas, $grupė_i$.PKsąrašas) then
	begin
	skydelis.PKklaida = 1
	$grupė_k$.PKklaida = 1
	// k žymi konkrečią grupę, iš kurios atėjo klaida

	end
	else
	add(skydelis.PKsąrašas, grupė$_i$.PKsąrašas
	if grupė$_i$.KKsąrašas ≠ ∅ then
	if isin(skydelis.KKsąrašas, grupė$_i$.KKsąrašas) then
	begin
	skydelis.KKklaida = 1
	grupė$_k$.KKklaida = 1
	// k žymi konkrečią grupę, iš kurios atėjo klaida
	end
	else
	add(skydelis.KKsąrašas, grupė$_i$.Kksąrašas

P_4	**<kortelė> → <T> <grupė> {<grupė>}**
	kortelė.aprašas = T.aprašas
	T.dait_veiksm = DAIKT
	kortelė.ąselės_vietos_rodiklis = kortelė.ąselės_plotis – T.fakt_plotis
	if kortelė.ąselės_vietos_rodiklis < 0
	then error(„trumpinti“)
	if grupė$_i$.PKsąrašas ≠ ∅ then
	if isin(kortelė.PKsąrašas, grupė$_i$.PKsąrašas)
	then
	begin
	kortelė.PKklaida = 1
	grupė$_k$.PKlaida = 1 // k žymi grupę, iš kurios atėjo klaida
	end
	else add(kortelė.PKsąrašas, grupė$_i$.PKsąrašas)
	if grupė$_i$.KKsąrašas ≠ ∅ then
	if isin(kortelė.KKsąrašas, grupė$_i$.KKsąrašas)
	then
	begin
	kortelė.KKklaida = 1
	grupė$_k$.KKklaida = 1 // k žymi grupę, iš kurios atėjo klaida
	end
	else add(kortelė.KKsąrašas, grupė$_i$.KKsąrašas)

P_5	**<grupė> → <T> <valdiklis> {<valdiklis>}**
	grupė.aprašas = T.aprašas
	T.dait_veiksm = DAIKT
	grupė.plotis = $\sum_{i=1}^{n}$ valdiklis$_i$.plotis + (n–1)*grupė.tarpo_plotis
	grupė.Tsąrašas:
	add(grupė.Tsąrašas, valdiklis$_i$.Tsąrašas)
	if valdiklis$_i$.PK ≠ ε then
	if isin(grupė.PKsąrašas, valdiklis$_i$.PK)
	then
	begin
	grupė.PKklaida = 1
	valdiklis$_k$.PKklaida = 1
	// k žymi konkretų valdiklį, iš kurio klaida
	end
	else add(grupė.PKsąrašas, valdiklis$_i$.PK)
	if valdiklis$_i$.KK ≠ ε then
	if isin(grupė.KKsąrašas, valdiklis$_i$.KK)
	then
	begin
	grupė.KKklaida = 1

	valdiklis$_k$.KKklaida = 1 // k žymi konkretų valdiklį, iš kurio klaida end else add(grupė.KKsąrašas, valdiklis$_i$.KK)

P_6	**<grupė> → <T> <h_grupė> {<h_grupė >}**
	grupė.aprašas = T.aprašas
	T.dait_veiksm = DAIKT
	grupė.plotis = h_grupė.plotis
	grupė.aukštis = h_grupė.aukštis
	grupė.Tsąrašas:
	add(grupė.Tsąrašas, h_grupė$_i$.Tsąrašas)
	if h_grupė$_i$.PKsąrašas ≠ ∅ then
	if isin(grupė.PKsąrašas, h_grupė$_i$.PKsąrašas)
	then
	begin
	grupė.PKklaida = 1
	h_grupė$_k$.PKklaida = 1
	// k žymi konkrečią grupę, iš kurios atėjo klaida
	end
	else add(grupė.PKsąrašas, h_grupė$_i$.PKsąrašas)
	if h_grupė$_i$.KKsąrašas ≠ ∅ then
	if isin(grupė.KKsąrašas, h_grupė$_i$.KKsąrašas)
	then
	begin
	grupė.KKklaida = 1
	h_grupė$_k$.KKklaida = 1
	// k žymi konkrečią grupę, iš kurios atėjo klaida
	end
	else add(grupė.KKsąrašas, h_grupė$_i$.KKsąrašas)

P7	**<grupė> → <T> <v_grupė> {<v_grupė >}**
	grupė.aprašas = T.aprašas
	T.dait_veiksm = DAIKT
	grupė.plotis = v_grupė.plotis
	grupė.aukštis = v_grupė.aukštis
	add(grupė.Tsąrašas, v_grupė$_i$.Tsąrašas)
	if v_grupė$_i$.PKsąrašas ≠ ∅ then
	if isin(grupė.PKsąrašas, v_grupė$_i$.PKsąrašas)
	then
	begin
	grupė.PKklaida = 1
	v_grupė$_k$.PKklaida = 1
	// k žymi konkrečią grupę, iš kurios atėjo klaida
	end
	else add(grupė.PKsąrašas, v_grupė$_i$.PKsąrašas)
	if v_grupė$_i$.KKsąrašas ≠ ∅ then
	if isin(grupė.KKsąrašas, v_grupė$_i$.KKsąrašas)
	then
	begin
	grupė.KKklaida = 1
	v_grupė$_k$.KKklaida = 1
	// k žymi konkrečią grupę, iš kurios atėjo klaida
	end
	else add(grupė.KKsąrašas, v_grupė$_i$.KKsąrašas)

P_8	**<h_grupė> → <valdiklis> {<valdiklis>}** h_grupė.plotis = $\sum_{i=1}^{n}$ valdiklis$_i$.plotis + (n−1)*h_grupė.h_tarpas

	h_grupė.Tsąrašas:
	add(h_grupė.Tsąrašas, valdiklis$_i$.Tsąrašas)
	if valdiklis$_i$.PK ≠ ε then
	if isin(h_grupė.PKsąrašas, valdiklis$_i$.PK)
	then
	begin
	h_grupė.PKklaida = 1
	valdiklis$_k$.PKklaida = 1
	// k žymi konkretų valdiklį, iš kurio atėjo klaida
	end
	else add(h_grupė.PKsąrašas, valdiklis$_i$.PK)
	valdiklis$_i$.eilė = i
	if valdiklis$_i$.KK ≠ ε then
	if isin(h_grupė.KKsąrašas, valdiklis$_i$.KK)
	then
	begin
	h_grupė.KKklaida = 1
	valdiklis$_k$.KKklaida = 1
	// k žymi konkretų valdiklį, kuriame klaida
	end
	else add(h_grupė.KKsąrašas, valdiklis$_i$.KK)

P_9	**<v_grupė> → <valdiklis> {<valdiklis>}**
	v_grupė.aukštis = $\sum_{i=1}^{n}$ valdiklis$_i$.plotis + (n−1)* v_grupė.v_tarpas
	v_grupė.plotis = max{valdiklis$_i$.plotis}
	v_grupė.Tsąrašas:
	add(v_grupė.Tsąrašas, valdiklis$_i$.Tsąrašas)
	if valdiklis$_i$.PK ≠ ε then
	if isin(v_grupė.PKsąrašas, valdiklis$_i$.PK)
	then
	begin
	v_grupė.PKklaida = 1
	valdiklis$_k$.PKklaida = 1
	// k žymi konkretų valdiklį, kuriame dubliuojama
	end
	else add(v_grupė.PKsąrašas, valdiklis$_i$.PK)
	valdiklis$_i$.eilė = i
	if valdiklis$_i$.KK ≠ ε then
	if isin(v_grupė.KKsąrašas, valdiklis$_i$.KK)
	then
	begin
	v_grupė.KKklaida = 1
	valdiklis$_k$.KKklaida = 1
	// k žymi konkretų valdiklį, kuriame klaida
	end
	else add(v_grupė.KKsąrašas, valdiklis$_i$.KK)

P_{10}	**<valdiklis> → <tekst_užrašas>** valdiklis.aprašas = tekst_užrašas.aprašas valdiklis.vietos_rodiklis = tekst_užrašas.vietos_rodiklis valdiklis.plotis = tekst_užrašas.plotis valdiklis.Tsąrašas = tekst_užrašas.Tsąrašas tekst_užrašas.eilė = valdiklis.eilė
P_{11}	**<valdiklis> → <išskl_sąrašas>** valdiklis.aprašas = išskl_sąrašas.aprašas valdiklis.vietos_rodiklis = išskl_sąrašas.vietos_rodiklis valdiklis.plotis = išskl_sąrašas.plotis

	valdiklis.Tsąrašas = išskl_sąrašas.Tsąrašas išskl_sąrašas.eilė = valdiklis.eilė
P_{12}	**<valdiklis> → <teksto_laukas>** valdiklis.aprašas = teksto_laukas.aprašas valdiklis.vietos_rodiklis = teksto_laukas.vietos_rodiklis valdiklis.plotis = teksto_laukas.plotis valdiklis.Tsąrašas = teksto_laukas.Tsąrašas teksto_laukas.eilė = valdiklis.eilė
P_{13}	**<valdiklis> → <mygtukas>** valdiklis.aprašas = mygtukas.aprašas valdiklis.vietos_rodiklis = mygtukas.vietos_rodiklis valdiklis.plotis = mygtukas.plotis valdiklis.Tsąrašas = mygtukas.Tsąrašas valdiklis.PK = mygtukas.PK mygtukas.PKklaida = valdiklis.PKklaida mygtukas.eilė = valdiklis.eilė valdiklis.KK = mygtukas.KK mygtukas.KKklaida = valdiklis.KKklaida
P_{14}	**<valdiklis> → <žym_langelis>** valdiklis.aprašas = žym_langelis.aprašas valdiklis.vietos_rodiklis = žym_langelis.vietos_rodiklis valdiklis.plotis = žym_langelis.plotis valdiklis.Tsąrašas = žym_langelis.Tsąrašas valdiklis.PK = žym_langelis.PK if valdiklis.PKklaida then žym_langelis.PKklaida = 1 žym_langelis.eilė = valdiklis.eilė
P_{15}	**<valdiklis> → <žym_akutė>** valdiklis.aprašas = žym_akutė.aprašas valdiklis.vietos_rodiklis = žym_akutė.vietos_rodiklis valdiklis.plotis = žym_akutė.plotis valdiklis.Tsąrašas = žym_akutė.Tsąrašas valdiklis.PK = žym_akutė.PK if valdiklis.PKklaida then žym_akutė.PKklaida = 1 žym_akutė.eilė = valdiklis.eilė
	... // kiti valdiklių tipai, naudojami programoje
P_{16}	**<tekst_užrašas> → <T>{<T>}** T_i.v_tipas = „teksto užrašas“ T_i.eilė = tekst_užrašas.eilė tekst_užrašas.vietos_rodiklis = tekst_užrašas.plotis – $\sum_{i=1}^{n} T_i$.fakt_plotis if tekst_užrašas.vietos_rodiklis < 0 then error("trumpinti") tekst_užrašas.Tsąrašas: add(tekst_užrašas.Tsąrašas,T_i.id)
P_{17}	**<teksto_laukas> → <T>{<T>} | ε** T_i.v_tipas = „teksto laukas“ T_i.eilė = teksto_laukas.eilė teksto_ laukas.vietos_rodiklis = teksto_ laukas.plotis – $\sum_{i=1}^{n} T_i$.fakt_plotis if teksto_ laukas.vietos_rodiklis < 0 then error(„trumpinti“) teksto_ laukas.list: add(teksto_laukas.Tsąrašas,T_i.id)
P_{18}	**<mygtukas> → <T>{<T>}[<PK>][<KK>][<PE>]** T_i.v_tipas = „mygtukas“

	T_i.eilė = mygtukas.eilė mygtukas.vietos_rodiklis = mygtukas.plotis – $\sum_{i=1}^{n} T_i$.fakt_plotis if mygtukas.vietos_rodiklis < 0 then error(„trumpinti“) T.funkcija = mygtukas.funkcija mygtukas.list: add(mygtukas.Tsąrašas,T_i.id) mygtukas.PK = PK.reikšmė PK.PKklaida = mygtukas.PKklaida mygtukas.KK = KK.reikšmė KK.KKklaida = mygtukas.KKklaida
P_{19}	**<žym_akutė> → <T>{<T>}[<PK>][<PE>]** T_i.v_tipas = "žymimoji akutė" T_i.eilė = žym_akutė.eilė žym_ akutė.vietos_rodiklis = žym_ akutė.plotis – $\sum_{i=1}^{n} T_i$.fakt_plotis if žym_akutė.vietos_rodiklis < 0 then error(„trumpinti“) žym_akutė.list: add(žym_akutė.Tsąrašas,T_i.id) žym_akutė.PK = PK.reikšmė if žym_akutė.PKklaida then PK.PKklaida = 1
P_{20}	**<žym_langelis> → <T>{<T>}[<PK>][<PE>]** T_i.v_tipas = "žymimasis langelis" T_i.eilė = žym_langelis.eilė žym_langelis.vietos_rodiklis = žym_langelis.plotis – $\sum_{i=1}^{n} T_i$.fakt_plotis if žym_langelis.vietos_rodiklis < 0 then error(„trumpinti“) žym_langelis.list: add(žym_langelis.Tsąrašas,T_i.id) žym_langelis.PK = PK.reikšmė if žym_langelis.PKklaida then PK.PKklaida = 1
P_{21}	**<išskl_sąrašas> → <E>{<E>}** išskl_sąrašas.fakt_plotis = max{E_i.fakt_plotis} išskl_sąrašas.vietos rodiklis = išskl_sąrašas.plotis – išskl_sąrašas.fakt_plotis if išskl_sąrašas.vietos_rodiklis < 0 then error(„trumpinti“) išskl_sąrašas.elem_sk = count(E_i) išskl_sąrašas.list: add(išskl_sąrašas.Tsąrašas, E_i.Tsąrašas) E_i.eilė = išskl_sąrašas.eilė
P_{22}	**<PK> → R** PK.reikšmė = R
P_{23}	**<KK> → R** KK.reikšmė = R
P_{24}	**<PE> → <T>** PE.id = T.id
P_{25}	**<E> → <T>{<T>}** T_i.v_tipas = „išskleidžiamasis sąrašas“ T_i.eilė = E.eilė

	$E.fakt\_plotis = \sum_{i=1}^{n} T_i.fakt\_plotis$ E.Tsąrašas: add(E.Tsąrašas, T_i.id)
P_{26}	**<T> → (S | <P>) {(S | <P>)}** $T.fakt\_plotis = \sum_{i=1}^{n} (S_i.fakt\_plotis + P_i.fakt\_plotis)$ T.raidė: if T.ryšys_idsąrašas ≠ ε and T.eilė > 1 and T.tikrinis = 0 then T.raidė = 0 S_i.fakt_plotis = length(S_i, mat_vienetai) S_i.id = T.id S_i.v_tipas = T.v_tipas S_i.ryšys_idsąrašas = T.ryšys_idsąrašas P_i.vald_žodis = T.vald_žodis P_i.daikt_veiksm = T.daikt_veiksm P_i.v_tipas = T.v_tipas P_i.ryšys_idsąrašas = T.ryšys_idsąrašas

P_{27}	**<P> → T {T} | ε**
	P.fakt_plotis = Max(T_i.fakt_plotis) | x // x – apytikslė reikšmė
	P.raidė:
	if P.tikrinis = 1 then P.raidė = 1 else P.raidė = 0
	P.idsąrašas:
	add(P.idsąrašas, T_i.id)
	T_i.duom_tipas = P.duom_tipas
	T_i.forma = P.forma

P_{28}	**<T> → S** T.aprašas = S.aprašas T.fakt_plotis = length(S, mat_vienetai) S.id = T.id S.daikt_veiksm = T.daikt_veiksm S.forma = T.forma

Žemiau pateikiama anketa, skirta pirminių atributų reikšmėms priskirti, o taip pat atkreipti programos autoriaus dėmesį į formų derinimą, sudurtinių eilučių naudojimo pavojų, terminijos klausimus (18 lentelė).

Anketą užpildo gramatikos sudarytojas. Stulpelyje „Atsakymas arba atsakymų variantai“ jis nurodo pirminio atributo reikšmę (jei langelis tuščias) arba pasirenka vieną iš galimų reikšmių. Paskutiniame stulpelyje pateikiami reikšmių pavyzdžiai, pastabos.

18 lentelė. Nuostatų lango gramatikos pirminių atributų priskyrimo anketa

Išv. tais.	**Atributo vardas**	**Klausimas**	**Atsakymas arba ats. variantai**	**Reikšmių pavyzdžiai, pastabos**
P_1	L.failo_vardas	Koks dialogo lango lokalizuojamųjų išteklių failo vardas ir kelias? (Kelias – santykinis, nuo lokalizuojamųjų išteklių šakninio katalogo.) Jei lango ištekliai pateikiami keliuose failuose, jų keliai pateikiami atskiriant juos kabliataškiu.		locale/browser/properties.dtd locale/browser/dialog.dtd; locale/browser/prefs.dtd

Išv. tais.	Atributo vardas	Klausimas	Atsakymas arba ats. variantai	Reikšmių pavyzdžiai, pastabos
P_1	L.plotis	Koks dialogo lango plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 400
P_1	L.aukštis	Koks dialogo lango aukštis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 600
P_1	L.eilutė	Lango pavadinimo teksto eilutė iš lokalizuojamųjų išteklių arba eilučių, skiriamų kabliataškiu, sąrašas.		Preferences
P_2	skydelis.pav_ plotis	Koks dialogo lango skydelio pavadinimui skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 60
P_2	skydelis.failo_ vardas	Koks skydelio lokalizuojamųjų išteklių failo vardas ir kelias? (Kelias – santykinis, nuo lokalizuojamųjų išteklių šakninio katalogo.)		lokale/prefs/privacy.dtd
P_3	skydelis.pav_ plotis	Koks dialogo lango skydelio pavadinimui skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 60
P_3	skydelis.failo_ vardas	Koks skydelio lokalizuojamųjų išteklių failo vardas ir kelias? (Kelias – santykinis, nuo lokalizuojamųjų išteklių šakninio katalogo.)		lokale/prefs/privacy.dtd
P_4	kortelė.ąselės_ plotis	Koks dialogo lango kortelės ąselėje tekstui parašyti skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 50
P_5	grupė.h_tarpas	Koks grupės horizontalaus tarpo tarp valdiklių plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 3
P_5	grupė.aukštis	Koks valdiklių grupei skirtas aukštis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 50
P_8	h_grupė.aprašas	Koks horizontaliosios valdiklių grupės aprašas?		Download preferences controls group
P_8	h_grupė.aukštis	Koks horizontaliai valdiklių grupei skirtas aukštis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 50
P_8	h_grupė.h_ tarpas	Koks horizontaliosios valdiklių grupės horizontalaus tarpo tarp valdiklių plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 3

Išv. tais.	Atributo vardas	Klausimas	Atsakymas arba ats. variantai	Reikšmių pavyzdžiai, pastabos
P_9	v_grupė.v_ tarpas	Koks vertikaliai išdėstytų valdiklių grupės horizontalaus tarpo tarp valdiklių plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 5
P_{16}	tekst_užrašas. aprašas	Koks teksto užrašo valdiklio aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		„Use up to“ appears in one line with text box and „MB of space for the cache“
P_{16}	tekst_užrašas. plotis	Koks teksto užrašui skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 40
P_{17}	teksto_laukas. aprašas	Koks teksto lauko valdiklio aprašas? (Pildoma, jei lauke rodyti skirtoje eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		Folder name or path as a default place for all downloads
P_{17}	teksto_laukas. plotis	Koks teksto lauke tekstui įrašyti skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 40
P_{18}	mygtukas. dialogas	Ar spustelėjus mygtuką iškviečiamas dialogo langas?	Taip Ne	
P_{18}	mygtukas.plotis	Koks mygtuko tekstui įrašyti skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 30
P_{18}	mygtukas. funkcija	Koks vardas vidinės programos funkcijos, susietos su mygtuku?		openFontDialog()
P_{19}	žym_akutė. aprašas	Kokia žymimosios akutės paskirtis?		If checked, the browser will as where to save every download
P_{19}	žym_akutė. plotis	Koks žymimosios akutės tekstui įrašyti skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 40
P_{20}	žym_langelis. aprašas	Kokia žymimojo langelio paskirtis?		If checked, closes the download dialog when all downloads are comlete

Išv. tais.	Atributo vardas	Klausimas	Atsakymas arba ats. variantai	Reikšmių pavyzdžiai, pastabos
P_{20}	žym_langelis. plotis	Koks žymimojo langelio tekstui įrašyti skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 40
P_{21}	išskl_sąrašas. aprašas	Pateikite išskleidžiamojo sąrašo paskirties aprašą.		A list of actions to do on browser start up. List elements continue sentence, started via text label.
P_{21}	išskl_sąrašas. plotis	Koks išskleidžiamojo sąrašo laukui skirtas maksimalus plotis pasirinktais matavimo vienetais?		Skaičius, pavyzdžiui, 40
P_{22}	PK.id	Koks prieigos klavišo identifikatorius? Jei nėra, tai palikite lauką tuščią.		Browse.Accesskey
P_{23}	KK.id	Koks komandos klavišo identifikatorius? Jei nėra, tai palikite lauką tuščią.		Browse.Commandkey
P_{24}	PE.aprašas	Koks paaiškinančiosios etiketės aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		Hint for Customize button
P_{25}	E.aprašas	Koks išskleidžiamojo sąrašo elemento aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		Opens a blank page on browser startup
P_{26}	S_i.aprašas	Koks kiekvieno eilutės segmento S_1, S_2, ..., S_n aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		Saves a web page in a text file
P_{26}	T.aprašas	Koks eilutės aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas		Title for browsing history options

Išv. tais.	Atributo vardas	Klausimas	Atsakymas arba ats. variantai	Reikšmių pavyzdžiai, pastabos
		paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		
P_{26}	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	VEIKSM DAIKT NA	*Pastaba*. Šio atributo reikšmė naudojama kaip numatytoji, bet gali būti pakeičiama kita taikant gramatikos semantikos taisykles.
P_{26}	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę (NA).		Save NA Page
P_{26}	T.ryšys_ idsąrašas	Nurodykite eilučių, kartu su šia eilute formuojančių vieną frazę ar sakinį, identifikatorius, atskirdami juos kabliataškiu. Jei nėra, palikite tuščią lauką.		useCache1.label; useCache2.label *Pastaba*. Tai susijusios eilutės, išdėstytos skirtinguose valdikliuose.
P_{26}	T.tikrinis	Ar eilutės tekstas yra tikrinis pavadinimas ar santrumpa arba prasideda tikriniu pavadinimu ar santrumpa?	Taip Ne	
P_{26}	T.id	Koks teksto eilutės identifikatorius?		0125
P_{26}	S_i.terminas	Jei eilutės segmente pavartotas nevienareikšmis terminas, nurodykite jį ir jo reikšmės numerį. Jei viename segmente yra keli tokie terminai, tai nurodykite juos skirdami kabliataškiu. Jei minėto tipo terminų nėra, palikite lauką tuščią.		call(1) certificate(2); key(4)
P_{27}	P.duom_tipas	Koks parametro duomenų tipas?	ASM_VARDAS TEKST N_SKAIT T_SKAIT DATA LAIK DAT_LAIK	*Pastaba.* Jeigu parametro reikšmė yra skaitinė (P.duom_tipas = N_SKAIT), tai reikia derinti toliau einančio segmento S formą (žr. pastabą po lentele). *Pastaba.* Jei P.duom_tipas = ASM_VARDAS, tai reikia numatyti šalia esančio segmento giminės (pvz., Jonas yra *prisijungęs*), kurią galima gauti iš naudotojo registracijos duomenų), linksnio (pvz., *Jono* laiškas) derinimą.

Išv. tais.	Atributo vardas	Klausimas	Atsakymas arba ats. variantai	Reikšmių pavyzdžiai, pastabos
P_{27}	P.tikrinis	Ar vietoje parametro skirtas įrašyti tekstas yra tikrinis pavadinimas arba santrumpa?	Taip Ne	
P_{27}	P.forma	Į kurį klausimą atsako parametro (galima) reikšmė? Pavyzdžiui, Who (subject)? Who (object)?, Where?, What (subject)? What (object)?, When?, Why? How many? ir t.t. Kuriuo prielinksniu galima apibūdinti žodžio formą?	Who (object) Who (subject) Where What (object) What (subject) When Why How many Whom To From In Kita: _____	
P_{28}	S.aprašas	Koks eilutės segmento, atitinkančio visą eilutę, aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		Saves web page in a text file
P_{28}	S.terminas	Jei eilutės segmente pavartotas nevienareikšmis terminas, tai koks jis ir koks jo reikšmės numeris? Jei viename segmente yra keli tokie terminai, tai nurodykite juos skirdami kabliataškiu. Jei minėto tipo terminų nėra, palikite lauką tuščią.		call(1) certificate(2); key(4)
P_{28}	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	VEIKSM DAIKT NA	*Pastaba*. Šio atributo reikšmė naudojama kaip numatytoji, bet gali būti pakeičiama kita taikant gramatikos semantikos taisykles.
P_{28}	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę (NA).		Save Find NA
P_{28}	T.id	Koks teksto eilutės identifikatorius?		Browse1
P_{28}	T.ryšys_ idsąrašas	Kokios eilutės kartu su šia eilute formuoja vieną frazę ar sakinį? Nurodykite jų identifikatorius, atskirdami kabliataškiais. Jei nėra, palikite tuščią lauką.		useCache1.label; useCache2.label *Pastaba*. Tai susijusios eilutės, išdėstytos skirtinguose valdikliuose.

Pastaba. Derinant skaitvardžio ir daiktavardžio formas, naudojamos įvairių kalbų formų scenarijai, pvz., lietuvių turi 3 formas: n%10==1 && n%100!=11 ? 0 : n%10>=2 && (n%100<10 or n%100>=20) ? 1 : 2; arabų – 6 formas: n==0 ? 0 : n==1 ? 1 : n==2 ? 2 : n%100>=3 && n%100<=10 ? 3 : n%100>=11 && n%100<=99 ? 4 : 5.

Apibendrinsime, kas atliekama lango atributinėje gramatikoje įvestų atributų ir semantikos taisyklių pagalba (17 ir 18 lentelės).

- Skaičiuojama vieta, skirta tam tikrai eilutei grafinės sąsajos elemente rodyti. Jei viršijamas vietos limitas, tai pranešama, kad eilutę reikia trumpinti. Tam tarnauja grupė atributų (16 lentelė): *plotis*, *aukštis*, *pav_plotis*, *pav_vietos_rodiklis*, *ąselės_plotis*, *ąselės_vietos_rodiklis*, *fakt_plotis*, *vietos_rodiklis*. Ši savybė leidžia parinkti teisingus eilučių ilgius ir aptikti klaidas dar iki lokalizacijos testavimo.
- Tikrinama, ar valdiklio prieigos klavišas (komandos pabrauktoji raidė, kuri suaktyvina valdiklį paspaudus tos raidės klavišą kartu su *Alt* klavišu), nėra dubliuojamas tokių klavišų galiojimo srityje. Kadangi prieigos klavišų galiojimo sritis yra aktyvaus lango dalis, kuri vienu metu rodoma ekrane, t. y. tas pats skydelis (jei jis nėra išskirstytas į korteles) arba ta pati kortelė, tai tikrinama tik tos srities parinkti valdymo klavišai. Tam naudojami atributai *PKsąrašas* (panaudotiems prieigos klavišams kaupti), *PKklaida* (pranešti apie padubliuotą prieigos klavišą srityje). Atributas *PKklaida* taip pat padeda fiksuoti, iš kuriuo grafinės sąsajos elemento atėjo klaida (kur buvo padubliuotas prieigos klavišas).
- Tikrinama, ar nėra padubliuotas komandos klavišas visame lange. Šiam tikslui įvesti ir naudojami atributai *KKsąrašas* (komandų klavišų sąrašui kaupti) ir *KKklaida* (klaidai apie pasikartojantį komandos klavišą pažymėti). Visų komandos klavišų sąrašas kaupiamas pradinio lango gramatikos simbolio atribute ir gali pasitarnauti jungiant lango gramatiką su kitomis programos dalinėmis gramatikomis, pvz., meniu.
- Kaupiami susijusių eilučių identifikatoriai išteklių sistemoje, pavyzdžiui, jei parametro vietoje įrašoma viena iš daugelio eilučių, tai visa tų eilučių grupė greičiausiai turės panašią formą, dėl to patogu jas nagrinėti ir versti kartu. Analogiškai kaupiami visų suduriamų eilučių identifikatoriai, tam tikro valdiklio eilučių alternatyvų grupės identifikatoriai, visos valdiklių grupės eilučių identifikatoriai. Taip galima sužinoti, kurios eilutės bus rodomos viena šalia kitos ir spręsti apie jų formų derinimą.
- Fiksuojamas eilutės (frazės) valdantysis žodis. Jei eilutė turi parametrų, tai tas žodis perduodamas ir vietoje parametro skirtoms įrašyti eilutėms, kadangi visos eilutės valdantysis žodis gali įtakoti parametro reikšmės formą.
- Naudojamas daiktavardinės ar veiksmažodinės frazės atributas, leidžiantis tiksliau išversti eilutę. Šis atributas perduodamas ir parametrui, jei eilutė jį turi, kadangi visa eilutės forma įtakoja ir parametro eilutės formą. Jei eilutė skirta rodyti lango, skydelio, kortelės ar valdiklių grupės pavadinime, tai automatiškai priskiriamas daiktavardinės frazės atributas, nes lietuvių kalboje pavadinimas dažniausiai turi daiktavardinę formą. Be abejo, ši reikšmė yra tik rekomendacinio pobūdžio, kiekvienu konkrečiu atveju sprendimą priima lokalizuotojas.
- Kiekviena eilutė ir daugelis naudotojo grafinės sąsajos elementų turi aprašo atributus, kurie prilygtų lokalizavimo komentarams ir leistų paaiškinti eilutės ar grafinio elemento paskirtį.
- Tikrinama, ar vietoje parametro įrašoma eilutė turi prasidėti didžiąja ar mažąja raide.

- Teksto eilutei, jos sudarančioms dalims – segmentams ir parametrams – yra priskiriamas valdiklio tipo atributas, nusakantis, kuriame valdiklyje tas tekstas bus rodomas. Tai yra svarbu lokalizavimui, kadangi ta pati eilutė, rodoma skirtinguose valdikliuose, gali būti verčiama skirtingai.
- Skaičiuojami sudėtinių valdiklių (pvz., išskleidžiamojo sąrašo) elementai.
- Eilutės parametrui ir vietoje parametro įrašomoms visoms eilutėms priskiriamas duomenų tipas, kuris leidžia geriau suderinti vietoje parametro įrašomos eilutės formą su tekstu, kuris bus rodomas šalia.
- Nurodoma sąsaja su komandos vidine funkcija (pvz., kuri funkcija programos pirminiame tekste realizuoja komandą, atliekamą spustelėjus mygtuką). Tai leidžia giliau išnagrinėti atliekamos komandos specifiką, parinkti taiklesnį pavadinimą lokalizavimo kalba.
- Atributas *dialogas* nurodo, ar spustelėjus mygtuką bus iškviečiamas dialogo langas, ar ne. Jei iškviečiama – šioje vietoje bus numatomas gramatikos dalių jungimas.
- Valdiklių eilei nurodyti įvestas atributas *eilė*. Naudingas tuomet, kai iš kelių valdiklių tekstų sudaromas bendras sakinys ar frazių grupė, pvz., 1) teksto užrašas: *Slapukus laikyti iki* 2) šalia esantis išskleidžiamasis sąrašas: *jų galiojimo laiko pabaigos*, *naršyklės seanso pabaigos*, *klausti prieš priimant slapuką*.
- Fiksuojamas ryšys eilučių, kurios bus rodomos šalia formuojant vieną sakinį ar frazių grupę. Tam naudojamas atributas *ryšys_idsąrašas*. Priklausomai nuo esamo ryšio ir valdiklių eilės nustatoma eilutės pirmoji raidė (didžioji ar mažoji). Ši reikšmė yra rekomendacinio pobūdžio.
- Atributas *forma* skirtas iš dalies nusakyti eilutės, įrašomos vietoje parametro, linksnį ar formą. Tai trumpas klausimas, į kurį jis turi atsakyti (*When? Where? Whom?* ir kt.) arba prielinksnis (*to*, *from* ir kt.).

Kadangi gramatika rašoma konkrečiai programai, tai programuotojas (lokalizavimo ir internacionalizavimo inžinierius) parenka konkrečiai sąsajos ir jos projektavimo specifikai tinkamesnius simbolius (pvz., jei naudojamos užklojamos valdiklių grupės, verta įtraukti tokią grupę atitinkantį pomedį vietoje atskirų valdiklių).

Pastebėsime, kad lokalizacijos atributinė gramatika skirsis nuo originalo ne tik terminaliniais simboliais, bet ir neterminalinių simbolių P ir terminalinių simbolių S eilės tvarka, kadangi tam tikroje lokalizacijoje parametro vieta gali būti perkelta dėl kitos žodžių tvarkos. Tokiu būdu gramatika yra nevienareikšmė, bet tai nėra kliūtis, kadangi lokalizuojamųjų eilučių ir jos dalių tvarka nėra reikšminė, kaip, pvz., programavimo kalbos konstrukcijų atpažinime, kai nuo terminalinių simbolių eilės keičiasi programos semantika.

4.3.3. Dalinių atributinių gramatikų jungimas

Kadangi mūsų nagrinėjamos atributinės gramatikos atributai nėra visi perduodami iki šakninio mazgo ir jame nėra renkama bendra visų struktūros (mūsų atvejų lokalizuojamųjų išteklių, aprašytų atributine gramatika) semantinė reikšmė (kaip buvo numatyta originaliame Knutho apibrėžime [Knu68]), o dalis atributų perduodama tik per keletą mazgų arba nėra perduodama, o priskiriama prie gramatikos simbolių ir yra skirta tam, kad teiktų informaciją lokalizuotojui, tai galime komponuoti bendrąją atributinę gramatiką iš smulkesnių dalių – dalinių atributinių gramatikų. Taip galima palaipsniui formalizuoti lokalizuojamuosius išteklius ir jų metainformaciją, o projektuojant naują programos versiją arba panašios paskirties programą, pasinaudoti anksčiau sudarytomis lokalizuojamųjų išteklių atributinių gramatikų dalimis – komponentais.

Kaip ir komponentiniame programavime, komponentinėse atributinėse gramatikose naudojamos komponentų sąsajos, paslėptoji realizacija ir atskiras apdorojimas (kompiliavimas). Literatūroje galima rasti keletą komponentinių (kai kurie autoriai jas vadina modulinėmis) atributinių gramatikų sudarymo principų, naudojamų programavimo kalboms projektuoti [Paa95]:

1. *Neterminalinis simbolis = komponentas*. Šiuo atveju neterminalinis simbolis pateikiamas kaip atributinės gramatikos komponentas. Tai leidžia aprašyti ir atskirai laikyti lingvistinius elementus, kurie yra panašūs skirtingose programose. Tokie komponentai gali būti importuojami į naujų programų specifikaciją arba tos pačios programos naujas versijas. Neterminalinio modulio sąsają sudaro neterminalinio simbolio atributai, o semantikos taisyklės yra paslėptos realizacijos dalyje. Taip galima pakeisti neterminalinio simbolio semantiką nekeičiant jį naudojančių komponentų. Neterminalinio simbolio išvedimo taisyklės taip pat pateikiamos realizacijos dalyje.
2. *Atributas = komponentas*. Taikant šį būdą laikoma, kad pagrindiniai semantiniai aspektai atributinėje gramatikoje yra atributai, o ne neterminaliniai simboliai. Tuomet komponentas yra sudarytas iš atributo visų semantikos taisyklių. Komponente taip pat pateikiamos ir visos išvedimo taisyklės, kuriose figūruoja atributas. Tačiau taikant šį būdą sunku realizuoti tikrą informacijos slėpimą ir komponentų realizacijos autonomiškumą.
3. *Semantinis aspektas = komponentas*. Šiuo atveju į komponentus skirstoma pagal skirtingus semantinius aspektus [Wat85; RT89; FMJ92]. Komponentas turi visas atributinės gramatikos dalis, kurios yra susijusios su tam tikru semantiniu aspektu. Taip paprastai keletas neterminalinių simbolių, išvedimo taisyklių ir semantikos taisyklių yra priskiriami vienam komponentui.

Formaliai aprašant lokalizuojamuosius programos išteklius, remiantis programos grafinės naudotojo sąsajos struktūra, patogu skirstyti atributines gramatikas į dalis pagal autonomiškus grafinės sąsajos komponentus (pvz., dialogo langas, meniu, priemonių juosta) bei pagal programos komponentus (pvz., naršyklės adresynas, žurnalas, pagrindinis langas ir t. t.), todėl natūralu taikyti atributinių gramatikų komponavimo principą *semantinis aspektas = komponentas*.

Kadangi programos ištekliai, atitinkantys programos komponentus ir jų dalis, ir aprašyti dalinėmis atributinėmis gramatikomis, turi autonomiškumo savybes (dauguma atributų yra perduodama dalinių gramatikų viduje), tai galima jungti tokias dalines atributines gramatikas nekeičiant dalinių gramatikų atributų ir semantikos taisyklių. Konteksto (atributų) perdavimas lokalizavimo požiūriu gali būti svarbus dalinių gramatikų sujungimo vietoje, pavyzdžiui, tam, kad būtų galima palyginti atributus ir pakoreguoti arba patikrinti išteklių eilutes ar jų vertimą. Taigi jungiant dalines atributines gramatikas reikėtų numatyti sujungimo vietos atributus, kuriuos svarbu analizuoti lokalizavimo metu.

Bendroji visos programos atributinė gramatika (AG_p) gaunama sujungus visų jos atskirų komponentų k_i (jei yra) dalių atributines gramatikas:

$$AG_p = AG_{k1} \cup AG_{k2} \cup ... \cup AG_{kn},$$

čia AG_{ki} – i-ojo programos komponento atributinė gramatika, i = 1, 2, ..., n.

$$AG_{ki} = AG_1 \cup AG_2 \cup ... \cup AG_m,$$

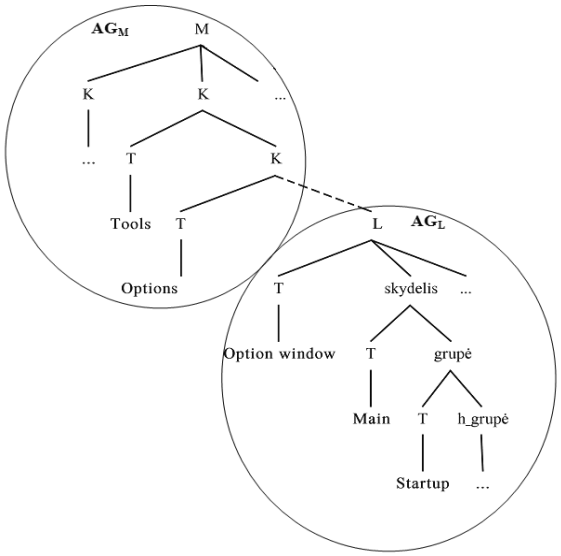
čia AG_i – tai k_i komponento j-osios dalies programos komponento atributinė gramatika, j = 1, 2, ..., m.

Norint apjungti, pavyzdžiui, pagrindinio meniu atributinę gramatiką su nuostatų langu atributine gramatika, reikia papildyti meniu gramatika dar viena išvedimo taisykle:

```
K -> T {T} [PK] [KK] [PE] L
```

Be papildomo L neterminalinio simbolio, ši taisyklė pasikeitė tuo, kad nėra įdėtinių meniu komandų {K}, nes komanda iškviečia dialogo langą o ne atveria žemesnio lygio meniu.

Dalinių gramatikų jungimas grafiškai pavaizduotas 25 paveiksle, čia AG_M – meniu atributinė gramatika, AG_L – lango atributinė gramatika.



25 pav. Atributinių gramatikų jungimo pavyzdžio schema

Langas iškviečiamas meniu komanda, todėl lango gramatika prijungiama prie komandos neterminalinio simbolio.

Be turimų išvedimo taisyklės `K -> T {T} [PK] [KK] [PE]` atributų (žr. AG_M) sujungimo vietoje lokalizavimo požiūriu svarbu patikrinti meniu komandos ir lango pavadinimo eilutes ir jose vartojamus terminus. Kadangi tokios eilutės gali būti pateikiamos lokalizuojamųjų ištelių failų sistemos skirtingose vietose, tai dažna lokalizavimo klaida yra parinkimas skirtingų terminų šioms eilutėms. Atributai padėtų tokias eilutes pamatyti vieną greta kitos – taip, kaip bus rodoma ekrane. Taip pat būtų galima palyginti sujungiamų neterminalinių simbolių aprašus, ar nebuvo klaidos sudarant gramatiką. Taigi gramatikų AG_M ir AG_L sujungimo vietą – išvedimo taisyklę – vertėtų papildyti tokiomis semantikos taisyklėmis:

```
K -> T {T} [PK] [KK] [PE] L
Print("Lyginti aprašus", K.aprašas, L.aprašas)
Print("Lyginti terminiją", K.eilutė, L.eilutė)
```

4.3.4. Lokalizuojamųjų išteklių metainformacijos formalizavimo metodo veiksmai

Apibendrindami ankstesnius skyrelius, skirtus lokalizuojamiems ištekliams ir jų metainformacijai formalizuoti, pateiksime metodo veiksmų seką.

1. Programinės įrangos komponentai (arba visa programa, jei ji neturi komponentų) skaidomi į autonomiškų dalių, atitinkančių grafinės naudotojo sąsajos susijusių elementų rinkinius, aibę. Komponentų dydis ir skaičius nėra esminis, svarbu, kad tai būtų toks komponentas, kurio lokalizuojamąsias eilutes būtų patogu analizuoti kartu.

2. Kiekvienai programinės įrangos grafinės naudotojo sąsajos komponento daliai (ar visam komponentui) sudaroma bekontekstė gramatika, imant pagrindu programinės įrangos grafinės sąsajos elementų, rodomų kompiuterio ekrane, struktūrą, o gramatikos neterminalinius ir terminalinius simbolius parenkant 4.2.2 skyriuje aprašytu būdu. Šiuo žingsniu formaliai aprašoma programinės įrangos grafinės sąsajos su naudotoju dalis, siejant ją su lokalizuojamųjų tekstinių išteklių eilutėmis bei skaidant tokias eilutes į dalis 4.2.3 skyriuje aprašytu būdu.
3. Kiekviena bekontekstė gramatika išplečiama iki atributinės gramatikos priskiriant kiekvienam gramatikos simboliui atributus ir aprašant semantikos taisykles, taip pat pagalbines išorines funkcijas. Atributų sąrašas, reikšmės bei semantikos taisyklės pateiktos 4.2.4 sk. Jie koreguojami ir parenkami atsižvelgiant į konkrečios PĮ specifiką.
4. Dalinės atributinės gramatikos jungiamos į visumą, numatant atributus, kurie bus perduodami dalinių gramatikų sujungimo vietose. Jei reikia, koreguojami atributai ir jų semantikos taisyklės, sudarytos ankstesniame žingsnyje.
5. Naudojantis sudaryta atributine gramatika lokalizuojamieji programos tekstiniai ištekliai yra transliuojami ir rezultatai įrašomi į failus.

4.4. Išvados

1. Tekstinių lokalizuojamųjų išteklių struktūros apibendrinimas bei internetinės programinės įrangos išteklių nagrinėjimo pavyzdžiai patvirtino, kad tokiuose ištekliuose nėra pateikiama kontekstinės informacijos apie eilutes, dominuoja trumpos eilutės (apie 50% eilučių ilgis yra nuo 1 iki 4 žodžių, apie 20% eilučių sudarytos iš 1–2 žodžių). Tokias eilutes taikliai ir kokybiškai išversti be papildomos kontekstinės informacijos yra beveik neįmanoma.
2. Pasiūlytas metodas, skirtas lokalizuojamųjų išteklių dviejų rūšių kontekstą nusakančiai metainformacijai formalizuoti:
   2.1. Vietai, kurioje tam tikra eilutė bus rodoma programos grafinėje sąsajoje, nurodyti ir ryšiams tarp susijusių eilučių nusakyti.
   2.2. Išteklių eilutės teksto semantiką nusakantiems atributams priskirti. Pateiktas išteklių eilučių pagrindinių atributų sąrašas, kuris yra išreiškiamas anglų kalbai, kuria kuriamos daugumos programinės įrangos originalai, būdingomis konstrukcijomis.
3. Pasiūlytas metodas turėtų prisidėti prie lokalizuojamųjų išteklių pateikimo sistemingumo gerinimo, kas padėtų pastebėti internacionalizavimo klaidas ankstyvojoje – programinės įrangos specifikavimo, projektavimo – stadijoje.
4. Siūlomas atributinių gramatikų taikymas lokalizuojamųjų išteklių kontekstui ir semantinei informacijai nurodyti, kuris skiriasi nuo atributinių gramatikų taikymų realizuojant programavimo kalbų transliatorius tuo, kad:
   4.1. Atributinės gramatikos naudojamos lokalizuojamųjų išteklių metainformacijai, nusakančiai kontekstą, formalizuoti ir iš programos autoriaus perduoti lokalizuotojams naudojant atributus.
   4.2. Naudojami išoriniai ir vidiniai atributai. Išoriniai atributai priskiriami gramatikos sudarytojo naudojantis anketa. Vidiniai atributai skaičiuojami automatiškai naudojant atributų semantikos funkcijas.
   4.3. Išoriniai atributai priskiriami ne tik terminaliniams simboliams, bet ir neterminaliniams simboliams.

4.4. Atributais perduodamos kontekstinės metainformacijos perdavimo sritis yra valdoma: metainformacija nėra perduodama visu atributiniu medžiu ir renkama šakniniame išvedimo medžio mazge, ji yra paskirstyta aktualių medžio mazgų aplinkoje.

4.5. Nebūtinas griežtas kalbos konstrukcijos (transliatorių atveju analogas būtų programavimo kalba parašytos programos, mūsų atveju – programos lokalizuojamųjų išteklių) atpažinimas. Akcentuojamas formalus aprašas su atributais, kuriuos lokalizuotojas gali pasiekti pagal užklausą, o ne transliavimas ir jo rezultatas.

4.6. Atributinė gramatika sudaroma ne tam, kad automatizuotume vertimą, bet tam, kad būtų galima analizuoti lokalizuojamuosius išteklius: matyti atributų nurodomą kontekstą, lyginti. Galutinį sprendimą, kaip lokalizuoti ar išversti tam tikrą eilutę, priima žmogus, dirbantis su šiais ištekliais.

5. EKSPERIMENTINIS LOKALIZUOJAMŲJŲ IŠTEKLIŲ METAINFORMACIJOS FORMALIZAVIMO METODO VERTINIMAS

Ankstesniame skyriuje pasiūlytas lokalizuojamųjų tekstinių išteklių metainformacijos formalizavimo metodas skirtas:

1. Mažinti lokalizavimo klaidų skaičių, perkeliant jų šalinimą į ankstyvąją lokalizavimo stadiją (potencialių klaidų aptikimas dar iki lokalizacijos testavimo).
2. Kuo daugiau internacionalizavimo klaidų aptikimo ir taisymo darbų perkelti į ankstyvąją (programinės įrangos projektavimo) stadiją.

Šio skyriaus pagrindinis tikslas – atlikti metodo ekspertinį vertinimą.

Prieš atliekant ekspertinį vertinimą metodas buvo pritaikytas tekstinių lokalizuojamųjų išteklių fragmento kontekstui sudaryti, o gautas pavyzdys naudotas ekspertų supažindinimo su metodo esme pradžioje, prieš pradedant išsamiau nagrinėti 4 skyriaus rezultatus.

5.1. Metodo taikymo pavyzdys

Pailiustruosime pasiūlyto metodo taikymą pavyzdžiu. Pavyzdys turėtų būti nedidelis ir vaizdus, kadangi jo sudarymo tikslas yra dvejopas:

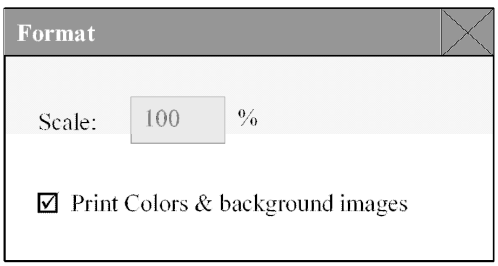
1. Pritaikyti metodą programinės įrangos fragmento lokalizuojamiems ištekliams.
2. Panaudoti tolesniam ekspertinio vertinimo tyrimui supažindinant ekspertus su metodo esme.

Paprastumo dėlei čia naudosime tik tuos atributus ir gramatikos simbolius, kurie reikalingi šio pavyzdžio elementams. Į pavyzdį neįeina automatizuoto eilutėms skirtos vietos ekrane tikrinimo, prieigos ir komandos klavišų parinkimo korektiškumo tikrinimo, kurie yra aprašyti ankstesniame skyriuje.

Tarkime, kad lokalizuojamųjų išteklių failas turi septynias eilutes, išdėstytas dažnai pasitaikančia lokalizavimo praktikoje forma (eilutės identifikatorius, lygybės ženklas ir eilutės tekstas):

```
01=Format
02=Scale:
03=&#037;
04=Print #1
05=Colors
06=Background images
07=Colors &amp;amp; background images
```

Programos dialogo lango fragmentas, kurį atitinka lokalizuotinos eilutės pavaizduotas 26 paveiksle.



26 pav. Dialogo lango fragmentas, kuriame naudojamos lokalizuotinos eilutės

Pastebėsime, kad lokalizuotojas, pradedantis lokalizuoti programą, paprastai mato tik aukščiau pateiktą lokalizuojamųjų išteklių failą, be 26 paveiksle pavaizduotos grafinės naudotojo sąsajos fragmento. Taip dirbdamas net su tokiu nedideliu išteklių failu gali padaryti nemažai klaidų, kurias galėtų aptikti tik testavimo metu.

Parinkime gramatikos simbolius ir sudarykime bekontekstę gramatiką.

G_L = <N_L, T_L, P_L, L_fragmentas>, čia

N_L = {<L_fragmentas>, <valdiklis>, <teksto_užrašas>, <teksto_laukas>, <žym_langelis>, <T>, <P>} – neterminalinių simbolių aibė;

T_L = {S}, S ∈ F, F = {Format; Scale:; %; Print; Colors; Background images; Colors & background image} – terminalinių simbolių aibė:

P_L – išvedimo taisyklių aibė:

P_L= {<L_fragmentas> → <T> <valdiklis> {<valdiklis>}
<valdiklis> → <teksto_užrašas>
<valdiklis> → <teksto_laukas>
<valdiklis> → <žym_langelis>
<teksto_užrašas> → <T>{<T>}
<teksto_laukas> → <T>{<T>} | ε
<žym_langelis> → <T>{<T>}
<T> → (S | <P>) {(S | <P>)}
<P> → T {T} | ε
<T> → S}

Priskirkime gramatikai atributus (atributų sąrašas pateikiamas 19 lentelėje). Naudojame tuos pačius žymėjimus, kaip ir ankstesniuose skyriuose.

19 lentelė. Pavyzdžio atributų sąrašas.

Atributo vardas	Atr. tipas	Atributo reikšmių tipas
aprašas	↑←	Teksto eilutė
daikt_veiksm	↓←	Teksto eilutė
duom_tipas	↓←	Teksto eilutė
eilė	↓←	Skaičius
forma	↓←	Teksto eilutė
id	↑←	Teksto eilutė
idsąrašas	↑	Teksto eilučių masyvas
raidė	↑←	Loginis
ryšys_idsąrašas	↓←	Teksto eilučių masyvas
terminas	←	Teksto eilutė
tikrinis	←	Loginis
vald_žodis	↓←	Teksto eilutė

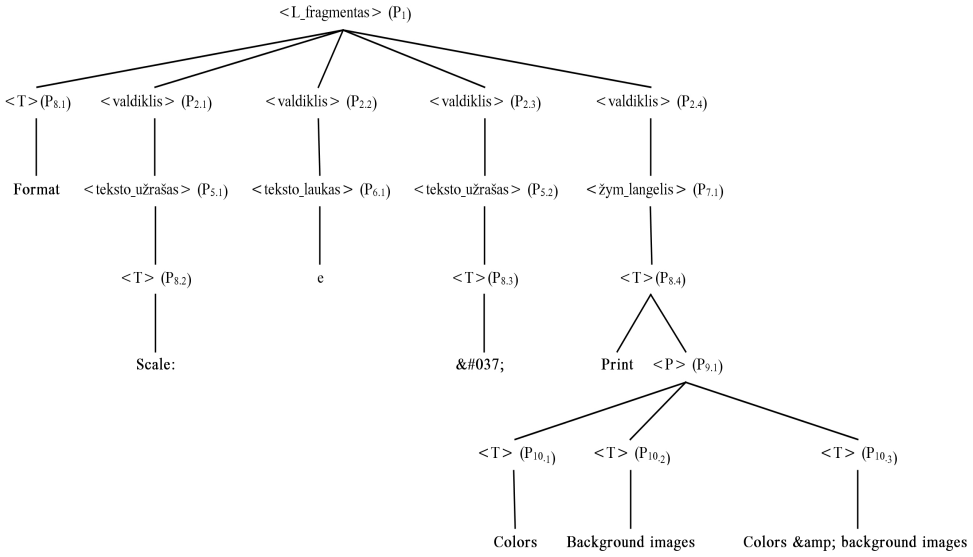
Atributinė gramatika minėtam programos fragmentui pateikiama 20 lentelėje.

20 lentelė. Lango fragmento atributinė gramatika

P_1	**<L_fragmentas> → <T> <valdiklis> {<valdiklis>}** L_fragmentas.aprašas = T.aprašas T.dait_veiksm = DAIKT T.raidė = 1
P_2	**<valdiklis> → <teksto_užrašas>** teksto užrašas.eilė = valdiklis.eilė
P_3	**<valdiklis> → <teksto_laukas>** teksto laukas.eilė = valdiklis.eilė

P_4	**<valdiklis> → <žym_langelis>** žym_langelis.eilė = valdiklis.eilė
P_5	**<teksto_užrašas> → <T>{<T>}** T_i.v_tipas = „teksto užrašas“ T_i.eilė = tekst_užrašas.eilė
P_6	**<teksto_laukas> → <T>{<T>} | ε** T_i.v_tipas = „teksto laukas“ T_i.eilė = teksto_laukas.eilė
P_7	**<žym_langelis> → <T>{<T>}** T_i.v_tipas = „žymimasis langelis“ T_i.eilė = žym_langelis.eilė
P_8	**<T> → (S | <P>) {(S | <P>)}** P_i.vald_žodis = T.vald_žodis P_i.daikt_veiksm = T.daikt_veiksm T.raidė: if T.eilė > 1 and T.tikrinis = 0 then T.raidė = 0 else T.raidė = 1 S_i.id = T.id S_i.v_tipas = T.v_tipas P_i.v_tipas = T.v_tipas
P_9	**<P> → T {T} | ε** P.raidė: if P.tikrinis = 1 then P.raidė = 1 else P.raidė = 0 P.idsąrašas: add(P.idsąrašas, T_i.id) T_i.duom_tipas = P.duom_tipas T_i.forma = P.forma
P_{10}	**<T> → S** S.id = T.id S.daikt_veiksm = T.daikt_veiksm S.forma = T.forma

Kadangi pirminių atributų reikšmės priskiriamos užpildant anketą, tai prieš tai reikia žinoti, kiek kartų kuri gramatikos išvedimo taisyklė bus pritaikyta. Išvedimo medis su išvedimo taisyklių ir jų pritaikymo numeriais pateikiamas 27 paveiksle.



27 pav. Išvedimo medis su taisyklių ir jų taikymų numeriais.

Pirminių atributų reikšmių priskyrimo anketa pateikiama 21 lentelėje, čia $P_{i,j}$: i = 1, 2, ..., 9, j = 1, 2, ..., n, i – taisyklės numeris, j – taisyklės taikymo eilės numeris, n – i-osios išvedimo taisyklės taikymų skaičius.

21 lentelė. Pavyzdžio pirminių atributų priskyrimo anketa

Išv. taisyklė	**Atributo vardas**	**Klausimas**	**Atsakymas arba ats. variantai**	**Reikšmių pavyzdžiai, pastabos**
P_1	valdiklis$_i$.eilė	Koks kiekvieno valdiklio eilės numeris? 1, 2, ..., n – jei valdikliai yra susiję, 0 – jei valdiklis yra atskiras.		0 1 2
P_1	L_fragmentas. plotis	Koks lango fragmento plotis (*em* vienetais)?		40
P_1	L_fragmentas. aukštis	Koks lango fragmento aukštis (*em* vienetais)?		30
P_1	T.aprašas	Koks eilutės aprašas? (Pildoma, jei eilutėje vartojama santrumpa ar specialieji ženklai, arba eilutei reikalingas paaiškinimas. Priešingu atveju laukas paliekamas tuščias.)		„Use up to“ appears in one line with text box and „MB of space for the cache“
$P_{8,j}$	T.daikt_ veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	VEIKSM DAIKT NA	*Pastaba*. Šio atributo reikšmė naudojama kaip numatytoji, bet gali būti pakeičiama kita taikant gramatikos semantikos taisykles.
$P_{8,j}$	T.vald_žodis	Koks eilutės valdantysis žodis (jei yra)? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.		Save NA Page
$P_{8,j}$	T.tikrinis	Ar eilutės tekstas yra tikrinis pavadinimas ar santrumpa arba prasideda tikriniu pavadinimu ar santrumpa?	Taip Ne	
$P_{8,j}$	T.id	Koks teksto eilutės identifikatorius?		0125
$P_{8,j}$	S_i.terminas	Jei eilutės segmente pavartotas nevienareikšmis terminas, koks jo reikšmės numeris? Jei viename segmente yra keli tokie terminai, tai nurodykite juos skirdami kabliataškiu. Jei minėto tipo terminų nėra, palikite lauką tuščią.		call(1) certificate(2); key(4)
$P_{9,j}$	P.duom_tipas	Koks parametro duomenų tipas?	ASM_VARD AS TEKST N_SKAIT T_SKAIT DATA LAIK DAT_LAIK	*Pastaba.* Jeigu parametro reikšmė yra skaitinė (P.duom_tipas = N_SKAIT), tai reikia derinti toliau einančio segmento S formą (žr. pastabą po lentele). *Pastaba.* Jei

				P.duom_tipas = ASM_VARDAS, tai reikia numatyti šalia esančio segmento giminės (pvz., Jonas yra *prisijungęs*), kurią galima gauti iš naudotojo registracijos duomenų), linksnio (pvz., *Jono* laiškas) derinimą.
$P_{9.j}$	P.tikrinis	Ar vietoje parametro skirtas įrašyti tekstas yra tikrinis pavadinimas arba santrumpa?	Taip Ne	
$P_{9.j}$	P.forma	Į kurį klausimą atsako parametro (galima) reikšmė? Pavyzdžiui, Who (object)?, Who (subject)?, Where?, What (object)?, What (subject)?, When?, Why? How many? ir t. t. Kuriuo prielinksniu galima apibūdinti žodžio formą?	Who (object) Who (subject) Where What (object) What (subject) When Why How many Whom To From In Kita:_____	
$P_{10.j}$	S.terminas	Jei eilutės segmente pavartotas nevienareikšmis terminas, koks jo reikšmės numeris? Jei viename segmente yra keli tokie terminai, tai nurodykite juos skirdami kabliataškiu. Jei minėto tipo terminų nėra, palikite lauką tuščią.		Call (1)
$P_{10.j}$	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	VEIKSM DAIKT NA	*Pastaba*. Šio atributo reikšmė naudojama kaip numatytoji, bet gali būti pakeičiama kita taikant gramatikos semantikos taisykles.
$P_{10.j}$	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.		Save
$P_{10.j}$	T.id	Koks teksto eilutės identifikatorius?		0023

Pastaba. Derinant skaitvardžio ir daiktavardžio formas, naudojamos įvairių kalbų formų scenarijai, pvz., lietuvių turi 3 formas: n%10==1 && n%100!=11 ? 0 : n%10>=2 && (n%100<10 or n%100>=20) ? 1 : 2; arabų – 6 formas: n==0 ? 0 : n==1 ? 1 : n==2 ? 2 : n%100>=3 && n%100<=10 ? 3 : n%100>=11 && n%100<=99 ? 4 : 5.

Užpildyta anketa pateikiama tolesnėje lentelėje. Išvedimo taisyklių ir jų taikymų numeriai atitinka numerius 27 paveikslo medyje.

Nurodant atributo *terminas* reikšmę, laikoma, kad naudojamasi žodynu, kuriame yra tokie arba analogiški įrašai:

Format

1. *Formatas*. Dokumento arba jo dalies (rašmens, skaičiaus, datos, pastraipos, lentelės ir kt.) vaizdavimo ir apipavidalinimo būdas.
2. *Formatas*. Popieriaus lapo matmenys. Nurodomas ilgio vienetais, pvz., 210×297 mm, 21×29,7 cm arba standartinių formatų pavadinimais, pvz., A4, A5.
3. *Ženklinti (diską)*. Komanda diską darbui paruošti. Yra dviejų lygių ženklinimas: žemo ir aukšto. Žemo lygio – paruošia takelius, suskirsto juos į sektorius, sužymi sektorius. Aukšto lygio – adresuoja sektorius ir suformuoja lenteles, reikalingas konkrečiai failų išdėstymo sistemai.

Scale

1. *Skalė*. Matavimo prietaiso rodmenų įtaisas – linija su padalomis ir šalia jų parašytomis matavimo vertėmis.
2. *Skalė*. Skaičių, spalvų ar kitokių ženklų, kurių vertės gali būti vartojamos matavimui arba palyginimui, surikiuota eilė.
3. *Mastelis*. Lange rodomo dokumento vaizdo padidinimas arba sumažinimas, išreikštas procentais (normalus 100%).

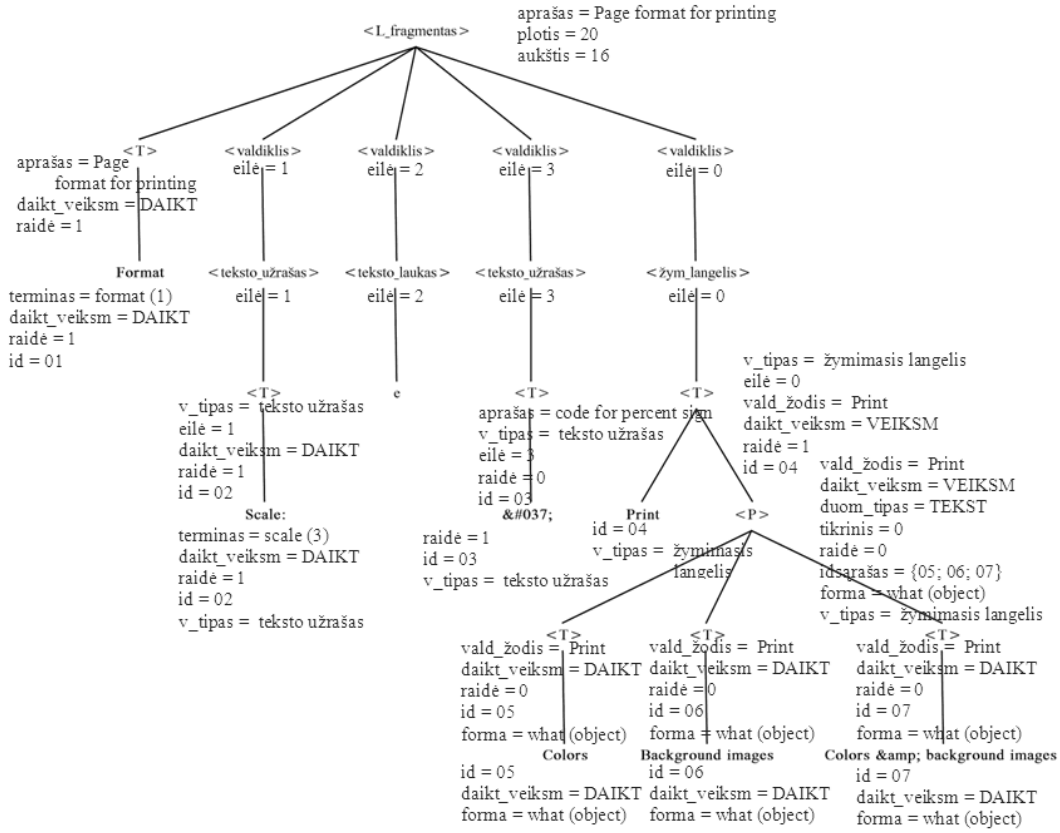
22 lentelė. Pirminių atributų reikšmės

Išv. tais.	**Atributo vardas**	**Klausimas**	**Atsakymas arba atsakymų variantai**
P_1	L_fragmentas.plotis	Koks lango fragmento plotis (em)?	20
P_1	L_fragmentas.aukštis	Koks lango fragmento aukštis (em)?	16
P_1	$valdiklis_i$.eilė	Koks kiekvieno valdiklio eilės numeris? 1, 2, ..., n – jei valdikliai yra susiję, 0 – jei valdiklis yra atskiras.	$valdiklis_1$.eilė = 1 $valdiklis_2$.eilė = 2 $valdiklis_3$.eilė = 3 $valdiklis_4$.eilė = 0
P_1	T.aprašas	Pateikite eilutės aprašą.	Page format for printing.
$P_{8.3}$	T.aprašas	Pateikite eilutės aprašą.	Code for percent sign
$P_{8.2}$	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	DAIKT
$P_{8.3}$	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	NA
$P_{8.4}$	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	VEIKSM
$P_{8.1}$	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.	NA
$P_{8.2}$	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.	NA
$P_{8.3}$	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.	NA

$P_{8.4}$	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.	Print
$P_{8.1}$	T.tikrinis	Ar eilutės tekstas yra tikrinis pavadinimas ar santrumpa arba prasideda tikriniu pavadinimu ar santrumpa?	Ne
$P_{8.2}$	T.tikrinis	Ar eilutės tekstas yra tikrinis pavadinimas ar santrumpa arba prasideda tikriniu pavadinimu ar santrumpa?	Ne
$P_{8.3}$	T.tikrinis	Ar eilutės tekstas yra tikrinis pavadinimas ar santrumpa arba prasideda tikriniu pavadinimu ar santrumpa?	Ne
$P_{8.4}$	T.tikrinis	Ar eilutės tekstas yra tikrinis pavadinimas ar santrumpa arba prasideda tikriniu pavadinimu ar santrumpa?	Ne
$P_{8.5}$	T.tikrinis	Ar eilutės tekstas yra tikrinis pavadinimas ar santrumpa arba prasideda tikriniu pavadinimu ar santrumpa?	Ne
$P_{8.6}$	T.tikrinis	Ar eilutės tekstas yra tikrinis pavadinimas arba prasideda tikriniu pavadinimu?	Ne
$P_{8.7}$	T.tikrinis	Ar eilutės tekstas yra tikrinis pavadinimas ar santrumpa arba prasideda tikriniu pavadinimu ar santrumpa?	Ne
$P_{8.1}$	T.id	Koks teksto eilutės identifikatorius?	01
$P_{8.2}$	T.id	Koks teksto eilutės identifikatorius?	02
$P_{8.3}$	T.id	Koks teksto eilutės identifikatorius?	03
$P_{8.4}$	T.id	Koks teksto eilutės identifikatorius?	04
$P_{8.1}$	S.terminas	Jei eilutės segmente pavartotas nevienareikšmis terminas, koks jo reikšmės numeris? Jei viename segmente yra keli tokie terminai, tai nurodykite juos skirdami kabliataškiu. Jei minėto tipo terminų nėra, palikite lauką tuščią.	Format (1)
$P_{8.2}$	S.terminas	Jei eilutės segmente pavartotas nevienareikšmis terminas, koks jo reikšmės numeris? Jei viename segmente yra keli tokie terminai, tai nurodykite juos skirdami kabliataškiu. Jei minėto tipo terminų nėra, palikite lauką tuščią.	Scale (3)
$P_{9.1}$	P.duom_tipas	Koks parametro duomenų tipas?	TEKST
$P_{9.1}$	P.tikrinis	Ar vietoje parametro skirtas įrašyti tekstas yra tikrinis pavadinimas ar santrumpa?	Ne
$P_{9.1}$	P.forma	Į kurį klausimą atsako parametro (galima) reikšmė? Pavyzdžiui, Who (subject)? Who (object)?, Where?, What (subject)? What (object)?, When?, Why? How many? ir t. t. Kuriuo prielinksniu galima apibūdinti žodžio formą?	What (object)
$P_{10.1}$	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	DAIKT
$P_{10.2}$	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	DAIKT

$P_{10.3}$	T.daikt_veiksm	Ar teksto eilutė yra veiksmažodinė (VEIKSM), ar daiktavardinė (DAIKT)? Jei negalima nustatyti, tai nurodoma NA.	DAIKT
$P_{10.1}$	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.	Print
$P_{10.2}$	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.	Print
$P_{10.3}$	T.vald_žodis	Koks eilutės valdantysis žodis? Jei negalima išskirti, nurodykite neapibrėžtą reikšmę NA.	Print
$P_{10.1}$	T.id	Koks teksto eilutės identifikatorius?	05
$P_{10.2}$	T.id	Koks teksto eilutės identifikatorius?	06
$P_{10.3}$	T.id	Koks teksto eilutės identifikatorius?	07

Išvedimo medis, papildytas atributais, pateikiamas 28 paveiksle. Jame žymime ne visus atributus: praleidžiame tuos atributus, kuriems priskirtos reikšmės NA (neapibrėžta).



28 pav. Pavyzdžio atributinis išvedimo medis

Lokalizuotojui svarbu pateikti simbolių <T> (visos eilutės), terminalinių simbolių (eilučių segmentų) bei šakninio simbolio atributus.

Pastebėsime, kad čia pateiktas pavyzdys skirtas pailiustruoti metodo esmę, tačiau neapima visų metodo taikymo aspektų. Pavyzdys skirtas pradėti ekspertų susipažinimą su metodu, o toliau remiamasi šio darbo teorine dalimi (4 skyrius).

5.2. Metodo ekspertinis vertinimas

Žmogaus ir kompiuterio sąveikos tyrimų literatūroje pažymima, kad ekspertų apklausos metodas yra aktualus vertinant naujus šios srities metodus ir projektavimo rezultatus.

Šiame darbe pasiūlytas metodas paspartina programos lokalizavimo procesą sumažinant lokalizuotinų eilučių konteksto paieškos ir lokalizacijos testavimo laiko sąnaudas, tačiau reikalauja papildomų sąnaudų sudarant lokalizuojamųjų išteklių atributinę gramatiką ir priskiriant atributų reikšmes.

Siekiant ištirti, kiek sumažėja programos lokalizavimo dalies sąnaudos ir padidėja projektavimo dalies sąnaudos, buvo atlikta ekspertų nuomonių apklausa – interviu.

Apklausai buvo pasirinkti labai aukštos kvalifikacijos programinės įrangos lokalizavimo srities ekspertai iš Lietuvos, tiek dirbantys lokalizavimo mokslinių tyrimų srityje, tiek programų projektavimo ir lokalizavimo verslo srityje. Susitikus su kiekvienu iš jų buvo pristatyta metodo esmė, pagrindiniai veiksmai, pateiktas metodo taikymo pavyzdys (5.1 sk.), skirtas pradiniam susipažinimui su metodo taikymu, supažindinta su šio darbo 4 dalimi.

Supažindinus su metodu, kiekvienam ekspertui buvo pateikti iš anksto parengti interviu klausimai (5.2.3 skyrelis). Į interviu klausimus ekspertai turėjo atsakyti iš turimos patirties programų internacionalizavimo ir lokalizavimo srityse.

Apklausos tipas (interviu) buvo pasirinktas todėl, kad ekspertų apklausos atveju:

- respondentų skaičius nėra didelis, dėl to svarbu užtikrinti, kad kiekvienas respondentas vienareikšmiškai suprato klausimą,
- atsakant į klausimą ekspertui gali kilti papildomų klausimų apie metodą ir jo taikymą, į kuriuos tuoj pat galėtų gauti atsakymą iš apklausos gavėjo.

5.2.1. Ekspertų skaičius ir reikalavimai ekspertų kvalifikacijai

Taikant ekspertų apklausos metodą svarbu pasirinkti pakankamą ekspertų skaičių ir tinkamos kvalifikacijos ekspertus.

Ekspertų nuomonių agregavimo patirtis ir tyrimai rodo, kad tikslių rezultatų taikant ekspertų apklausos metodą galima pasiekti, kai pasirenkamas ekspertų skaičius apklausai yra nuo trijų iki šešių ekspertų [Ash86; AA85; Cle89]. Ekspertų skaičiaus didinimas taikant ekspertų apklausos metodą nedidina rezultatų tikslumo dėl didesnės tikimybės ekspertų tarpusavio priklausomybės [BR00]. Yra tyrimų, kuriuose nagrinėjamas vieno eksperto nuomonės apibendrinimas [YF09]. Svarbiausias rezultatų patikimumą lemiantis veiksnys yra ne ekspertų skaičius, bet ekspertų kvalifikacija.

Šiame darbe siekiant patikimų ir tikslių pasiūlyto metodo vertinimo rezultatų, ekspertų kvalifikacijai keliami tokie reikalavimai:

1. Ne mažesnė kaip septynerių metų patirtis programinės įrangos lokalizavimo arba (ir) internacionalizavimo srityse.
2. Lokalizavo ar dalyvavo lokalizavimo grupėje lokalizuojant ne mažiau kaip penkias kompiuterines programas.
3. Paskelbė bent tris mokslines publikacijas iš lokalizavimo ar internacionalizavimo temų.

Šiame darbe pasiūlytam metodui vertinti buvo atrinkti 3 ekspertai, kurie atitiko iškeltus kvalifikacinius reikalavimus.

5.2.2. Klausimų ekspertams sudarymas

Vertinant darbe pasiūlytą metodą svarbu išskirti du programinės įrangos lokalizavimo būdus:

1. Įprastasis (kai lokalizuojama be kontekstinės informacijos, o kontekstas išsiaiškinamas testuojant lokalizuotą programą ir kartojant vertimo, testavimo ir vertimų koregavimo ciklą).
2. Įtraukiant kontekstą (kai lokalizuojamieji ištekliai parengiami remiantis šiame darbe pasiūlytu metodu, o lokalizuojama turint kontekstinę informaciją).

   Antrojo lokalizavimo būdo galima išskirti du atvejus:

   1. Kontekstinę informaciją (gramatiką ir atributus) rengia programos autorius.
   2. Kontekstinę informaciją (gramatiką ir atributus) rengia kvalifikuotas lokalizuotojas arba patyręs programos naudotojas.

   Interviu klausimai, pateikiami ekspertams, turėtų tenkinti tokius reikalavimus:

1. Apimti aukščiau aprašytus lokalizavimo būdus, kai kontekstą rengia programos autorius ir lokalizuotojas arba patyręs naudotojas.
2. Duomenys, gauti atsakant į klausimus, turi leisti palyginti lokalizavimo darbo sąnaudas, netaikant šiame darbe pasiūlyto metodo ir pritaikius metodą.
3. Duomenys, gauti atsakant ekspertams į klausimus, turi leisti nuspręsti, kokioms sąlygoms esant verta taikyti metodą.
4. Klausimai turi būti trumpi ir aiškiai suformuluoti. Formuluotėse gali būti vartojamos sąvokos iš metodo ir ankstesnės disertacijos dalies be išplėstinių paaiškinimų, kadangi šios sąvokos ir metodo principai buvo pristatyti ekspertams ir aptarti akivaizdinio susitikimo metu.
5. Klausimų neturi būti daug (ne daugiau kaip 10 klausimų).
6. Atsakymai į klausimus neturėtų užimti daug laiko ir reikalauti papildomų tyrimų ekspertams, dirbantiems programinės įrangos lokalizavimo (internacionalizavimo) srityje.

5.2.3. Interviu klausimynas

Remiantis iškeltais reikalavimais prieš atliekant apklausą buvo sudarytas interviu klausimynas. Klausimų numeriai žymimi K1–K8.

K1–K7 klausimuose įvertinamas apibrėžtų darbų sąnaudų santykis. Siekiant klausimų formuluočių paprastumo, darbai, kurių sąnaudas reikia palyginti atsakant į klausimus, įvardyti ir pateikti atskirai prie kiekvieno klausimo.

K8 klausimas yra atvirasis, skirtas bendroms pastaboms apie šiame darbe pasiūlytą metodą išreikšti.

Visuose klausimuose, kuriuose minimas konteksto parengimas, turima omenyje, kad kontekstas rengiamas taikant šiame darbe pasiūlytą metodą. Kalbant apie lokalizavimo darbo sąnaudas turimos omenyje tik kokybiškai atlikto darbo sąnaudos.

K1. Prašom įvertinti, kiek kartų *1-ojo darbo* sąnaudos yra didesnės už *2-ojo darbo* sąnaudas. Atsakymą pakomentuokite.

1-asis darbas: lokalizuojamos programos testavimas ir vertimo koregavimas nagrinėjant kontekstą programai veikiant.

2-asis darbas: tekstinių lokalizuojamųjų išteklių vertimas, kai lokalizuojamieji programos ištekliai pateikiami be konteksto.

K2. Kiek kartų *1-ojo darbo* sąnaudos yra didesnės (mažesnės) už *2-ojo darbo* sąnaudas? Atsakymą pakomentuokite.

1-asis darbas: programos lokalizuojamųjų išteklių konteksto parengimas (gramatikos sudarymas, pirminių atributų reikšmių priskyrimas). Darbą atlieka programos autorius.

2-asis darbas: programos lokalizuojamųjų išteklių vertimas neturint konteksto.

K3. Kiek kartų *1-ojo darbo* sąnaudos yra mažesnės (didesnės) už *2-ojo darbo* sąnaudas? Atsakymą pakomentuokite.

1-asis darbas: lokalizuojamos programos testavimas ir vertimo koregavimas, kai lokalizuojama su išteklių kontekstine informacija.

2-asis darbas: lokalizuojamos programos testavimas ir vertimo koregavimas, kai lokalizuojama neturint išteklių kontekstinės informacijos.

K4. Kiek kartų padidėtų (ar sumažėtų) programos tekstinių lokalizuojamųjų išteklių vertimo darbo sąnaudos, jei būtų pateiktas kiekvienos lokalizuojamos eilutės kontekstas? Atsakymą pakomentuokite.

K5. Kokią dalį *1-ojo darbo* sąnaudų sudarytų *2-ojo darbo* sąnaudos? Atsakymą pakomentuokite.

1-asis darbas: programos lokalizuojamųjų išteklių konteksto parengimas (gramatikos sudarymas, pirminių atributų reikšmių priskyrimas). Darbą atlieka programos autorius.

2-asis darbas: konteksto parengimas kitai (papildomai) fleksinei abėcėlinei kalbai. Darbą atlieka programos autorius.

K6. Kiek kartų *1-ojo darbo* sąnaudos būtų didesnės (mažesnės) už *2-ojo darbo* sąnaudas? Atsakymą pakomentuokite.

1-asis darbas: programos lokalizuojamųjų išteklių konteksto parengimas (gramatikos sudarymas, pirminių atributų reikšmių priskyrimas). Kontekstą rengia programos autorius.

2-asis darbas: programos lokalizuojamųjų išteklių konteksto parengimas (gramatikos sudarymas, pirminių atributų reikšmių priskyrimas). Kontekstą rengia kvalifikuotas lokalizuotojas ar patyręs programos naudotojas.

K7. Kiek kartų *1-ojo darbo* sąnaudos būtų didesnės (mažesnės) už *2-ojo darbo* sąnaudas? Atsakymą pakomentuokite.

1-asis darbas: konteksto parengimas papildomai fleksinei abėcėlinei kalbai. Darbą atlieka kvalifikuotas lokalizuotojas ar patyręs programos naudotojas.

2-asis darbas: konteksto parengimas papildomai fleksinei abėcėlinei kalbai. Darbą atlieka programos autorius.

K8. Prašome pateikti Jūsų pastabas ir siūlymus dėl metodo ir jo taikymo (laisva forma).

5.2.4. Atsakymų analizės principai

Analizuojant įvairių lokalizavimo etapų darbų sąnaudų santykius, matavimo vienetu paimtas programos lokalizuojamųjų išteklių vertimui reikalingos darbo sąnaudos, kai verčiama neturint kontekstinės informacijos. Kitiems etapams (kontekstinei informacijai parengti, išversti tekstinius išteklius turint kontekstinę informaciją ir pan.) reikalingos darbo sąnaudos skaičiuojamos minėto pasirinkto vieneto pagrindu.

23 lentelėje pateikiami svarbiausi tiriamieji aspektai, juos atitinkantys klausimai iš ekspertų interviu, koeficientas, kurį gauname iš eksperto jam atsakius į atitinkamą klausimą, bei formulė, pagal kurią galima apskaičiuoti kiekvieną aspektą atitinkančias darbo sąnaudas. Lentelės paskutiniame stulpelyje naudojamas daugiklis 100 atitinka darbo sąnaudas, reikalingas verčiant programos tekstinius lokalizuojamuosius išteklius neturint konteksto (dabartinė programinės įrangos lokalizavimo praktika). Toks vienetas buvo pasirinktas dėl to, kad jis yra nepriklausomas nuo konkrečios lokalizuojamos programos bei programos (ar jos dalies) dydžio.

23 lentelė. Atsakymų naudojimas darbo sąnaudoms skaičiuoti taikant metodą ir netaikant metodo

Klausimas	Tiriamasis aspektas	Koeficientas	Formulė
K1	Testavimas ir vertimo koregavimas neturint konteksto	a_1	$a_1 \times 100$
K2	Konteksto parengimas (programos autorius)	b_1	$b_1 \times 100$
K3	Testavimas ir vertimo koregavimas (turint kontekstą)	b_2	$a_1 b_2 \times 100$
K4	Vertimas turint kontekstą	b_3	$b_3 \times 100$
K5	Konteksto parengimas kitai abėcėlinei fleksinei kalbai (programos autorius)	b_4	$b_1 b_4 \times 100$
K6	Konteksto parengimas (lokalizuotojas ar naudotojas)	c_1	$b_1 c_1 \times 100$
K7	Konteksto parengimas kitai abėcėlinei fleksinei kalbai (lokalizuotojas ar naudotojas)	c_2	$b_1 b_4 c_2 \times 100$

K8 klausimas yra atvirasis, skirtas eksperto pastaboms apie metodą ir jo naudingumą išreikšti.

Remiantis gautais iš ekspertų įvertinimais, galima apskaičiuoti darbo sąnaudas, reikalingas programai lokalizuoti netaikant metodo bei palyginti jas su darbo laiko sąnaudomis, reikalingomis programai lokalizuoti pritaikius metodą vienai kalbai ir kiekvienai kitai papildomai kalbai, kai kontekstą rengia programos autorius arba kvalifikuotas lokalizuotojas ar patyręs programos naudotojas (24 lentelė).

24 lentelė. Lokalizavimo darbo sąnaudų įvairių atvejų skaičiavimas taikant metodą ir netaikant metodo

Tiriamasis aspektas	Darbų sąrašas	Formulė
Programos lokalizavimas netaikant metodo	Vertimas, testavimas ir vertimo koregavimas.	$100a_1 + 100$
Programos lokalizavimas pritaikius metodą vienai kalbai (kontekstą rengia autorius)	Konteksto parengimas (autorius), vertimas turint kontekstą, testavimas ir vertimo koregavimas turint kontekstą.	$(b_1+b_3+a_1b_2)\times 100$
Programos lokalizavimas pritaikius metodą kiekvienai papildomai n-ajai kalbai ($n \geq 2$) (kontekstą rengia autorius)	Konteksto perkėlimas (autorius), vertimas turint kontekstą, testavimas ir vertimo koregavimas turint kontekstą.	$(b_1b_4+b_3+a_1b_2)\times 100$
Programos lokalizavimas pritaikius metodą vienai kalbai (kontekstą rengia kvalifikuotas lokalizuotojas ar patyręs programos naudotojas)	Konteksto parengimas (kvalifikuotas lokalizuotojas ar patyręs programos naudotojas), vertimas turint kontekstą, testavimas ir vertimo koregavimas turint kontekstą.	$(b_1c_1+b_3+a_1b_2)\times 100$
Programos lokalizavimas pritaikius metodą kiekvienai papildomai n-ajai kalbai ($n \geq 2$) (kontekstą rengia kvalifikuotas lokalizuotojas ar patyręs programos naudotojas)	Konteksto perkėlimas (kvalifikuotas lokalizuotojas ar patyręs programos naudotojas), vertimas turint kontekstą, testavimas ir vertimo koregavimas turint kontekstą.	$(b_1b_4\,c_2+b_3+a_1b_2)\times 100$

5.2.5. Ekspertų atsakymų analizė

25 lentelėje yra pateikti klausimuose minimų darbų santykių ekspertų įvertinimai.

25 lentelė. Ekspertų atsakymai į klausimus (nurodant, kiek kartų didesnės ar mažesnės sąnaudos) ir atitinkamų koeficientų reikšmės

Kl. Nr.	Atsakymas					
	Ekspertas A		**Ekspertas B**		**Ekspertas C**	
K1	3 k. didesnės	$a_1 = 3$	2,5 k. didesnės	$a_1 = 2,5$	3,5 k. didesnės	$a_1 = 3,5$
K2	1,2 k. mažesnės	$b_1 = 0,8$	maždaug vienodos	$b_1 = 1$	maždaug vienodos	$b_1 = 1$
K3	2,5 k. mažesnės	$b_2 = 0,4$	3 k. mažesnės	$b_2 = 0,3$	3 k. mažesnės	$b_2 = 0,3$
K4	padidėtų 1,5 karto	$b_3 = 1,5$	1,2 k. sumažėtų	$b_3 = 0,8$	2 k. sumažėtų	$b_3 = 0,5$
K5	vieną dešimtąją	$b_4 = 0,1$	vieną dešimtąją	$b_4 = 0,1$	trečdalį	$b_4 = 0,3$
K6	3 kartus didesnės	$c_1 = 3$	2,5 k. didesnės	$c_1 = 2,5$	3 kartus didesnės	$c_1 = 3$
K7	3 kartus didesnės	$c_2 = 3$	2,5 k. didesnės	$c_2 = 2,5$	maždaug vienodos	$c_2 = 1$

Toliau analizuosime ekspertų komentarus dėl atsakymo į kiekvieną pateiktą klausimą ir galimas priežastis, kodėl buvo gauti labiau besiskiriantys atsakymai į kai kuriuos klausimus.

5.2.5.1. Testavimas ir vertimo koregavimas netaikant metodo

Iš ekspertų atsakymų (25 lentelė) matyti, kad testavimo ir vertimo koregavimo darbo sąnaudas, palygintas su vertimo sąnaudomis, kai lokalizuojama įprastu būdu (neturint kontekstinės informacijos) ekspertai vertina panašiai: santykis svyruoja nuo 2,5 iki 3,5.

Ekspertas A, nurodęs, kad testavimo darbo sąnaudos yra 3 kartus didesnės už vertimo sąnaudas, pakomentavo, kad toks vertinimas buvo nustatytas eksperimentiškai iš jo darbo patirties lokalizuojant programas ir naudotas paskirstant atlyginimą už atskirus, šiame klausime minimus, programų lokalizavimo darbus.

Ekspertas B nurodė, kad testavimo ir vertimo koregavimo darbo sąnaudos yra maždaug 2–3 kartus didesnės, nes dirbant su programa jos grafinėje sąsajoje reikia ieškoti kiekvienos lokalizuojamuose ištekliuose esančios frazės. Be to, radus klaidą ir ją pataisius, reikia vėl tikrinti, ar būtent ta frazė buvo ištaisyta. O kai kurias programas, kuriose lokalizuojamieji ištekliai nėra interpretuojami, tenka iš naujo perkompiliuoti norint daryti kiekvieną pakartotiną eilučių tikrinimą.

Ekspertas C atsakė, kad testavimo ir vertimo koregavimo darbo sąnaudos mažiausiai 3–4 kartus yra didesnės už lokalizuojamųjų išteklių vertimą. Iš savo darbo patirties pažymėjo, kad jei nepavyksta rasti išverstos eilutės testuojant programą arba ta pati eilutė pasirodo keliose programos grafinės sąsajos vietose, arba yra dažnai pasikartojančių eilučių, tai tada tenka tokias eilutes numeruoti (žymėti), kad vėliau būtų galima stebėti, kurioje programos grafinės sąsajos vietoje jos atsiranda ir taisyti vertimą. Baigus testavimą, pagalbinė numeracija pašalinama.

5.2.5.2. Lokalizuojamųjų išteklių konteksto parengimas

Darbo sąnaudų, reikalingų lokalizuojamųjų išteklių kontekstui parengti remiantis šiame darbe pasiūlytu metodu, santykis su įprastu lokalizuojamųjų išteklių vertimu neturint konteksto, buvo įvertintas panašiai visų trijų ekspertų (25 lentelė).

Visi apklausti ekspertai atsakė, kad lokalizuojamųjų išteklių metainformacijai (kontekstui) parengti reikia įdėti maždaug tiek pat darbo, kiek ištekliams išversti neturint konteksto.

Ekspertas A atsakė, kad intelektualaus darbo sąnaudos maždaug vienodos, o savo atsakymą pagrindė tuo, kad programos autorius gerai žino programą ir formalias kalbas, o vertėjas (lokalizuotojas) gerai žino kalbą. Tačiau kontekstui užrašyti (techniniam darbui) laiko sąnaudų reikia šiek tiek mažiau, kadangi formalūs užrašai trumpesni ir vienareikšmiai, nereikia galvoti apie kalbos stilių. Dėl to galima įvertinti, kad konteksto parengimo sąnaudos mažesnės 1,2 karto.

Ekspertas B atsakė, kad darbo sąnaudos maždaug vienodos. Autorius gali parengti kontekstą gana sparčiai, nes jis žino, kur kuri eilutė pasirodys programos grafinėje sąsajoje. Maždaug tiek pat sąnaudų reikia profesionaliam lokalizuotojui pirminį eilučių vertimą atlikti.

Ekspertas C atsakė, kad darbo sąnaudos maždaug vienodos, nes kai programos autorius rengia kontekstą jam reikia apgalvoti kiekvieną eilutę, priskirti jai atributus, panašiai ir lokalizuotojui, kuris atlieka pirminį eilučių vertimą: jam reikia apgalvoti kiekvieną eilutę ir išversti.

5.2.5.3. Testavimas ir vertimo koregavimas esant kontekstui

Testavimo ir vertimo koregavimo darbų, kai lokalizuojamieji ištekliai pateikiami su kontekstu ir be konteksto, sąnaudų santykį visi apklausti ekspertai įvertino panašiai: su kontekstu darbo sąnaudos mažesnės nuo 2,5 iki 3 karto (25 lentelė).

Ekspertai A ir B pateikė testavimo sąnaudų turint kontekstą sumažėjimo panašų pagrindimą: „verčiant eilutes su kontekstu iš karto parenkamas taiklesnis ir kokybiškesnis vertimas, todėl tenka mažiau darbo skirti testavimui: kai eilutės išverstos teisingos, nereikia kartoti taisymo, perkompiliavimo ir tolesnio testavimo darbų“.

Ekspertas C atsakė, kad turint kontekstą testavimo ir vertimo koregavimo sąnaudos sumažėtų mažiausiai tris kartus, ir pateikė pagrindimą, analogišką ekspertų A ir B komentarui. Taip pat nurodė ir antrą testavimo darbų sąnaudų sumažėjimo priežastį: „Testuojant atskiras eilutes, pvz., programos klaidų pranešimus, jų vietą programoje rasti bus daug paprasčiau turint metodo numatomą išteklių formalųjį aprašą. „Aklas“ ieškojimas reikalautų labai daug laiko“.

5.2.5.4. Išteklių vertimas turint kontekstą

Lyginant darbo sąnaudas, reikalingas išversti išteklių eilutes, pateiktas su kontekstu, su darbo sąnaudomis, reikalingomis išversti išteklių eilutes, pateiktas be konteksto, ekspertų nuomonės išsiskyrė (25 lentelė).

Ekspertas A nurodė, kad darbo sąnaudos verčiant su kontekstu padidėtų 1,5 karto, paaiškindamas savo atsakymą tuo, kad vertėjui (lokalizuotojui) reikia papildomai nagrinėti kiekvienos eilutės kontekstą: atributų reikšmes, išvedimo medžio elementus, vietoj to, kad tiesiog išverstų (nors ir verčiant tenka kiekvieną eilutę apgalvoti).

Ekspertas B atsakė, kad vertimo su kontekstu darbo sąnaudos maždaug 1,2 karto sumažėtų apibendrinus tai, kad reikia peržiūrėti pateiktą kiekvienos eilutės kontekstą, tačiau nereikia jos ieškoti pačiam grafinėje programos sąsajoje.

Ekspertas C įvertino, kad su verčiant su kontekstu darbo sąnaudos būtų 2 kartus mažesnės, palyginus su vertimu be konteksto. Ekspertas pažymėjo, kad „žinoma, kontekstą skaitant kiek laiko būtų sugaišta, failai būtų didesni, tačiau patyręs lokalizuotojas ne viską ir skaitytų, o skaitytų tas vietas, kur jam neaišku – jam pagreitintų darbą dar labiau. Verčiant reikėtų rečiau žiūrėti į žodynus, lyginti daugiareikšmių terminų reikšmes, bandyti atspėti, kuriame programos lange pasirodys eilutė, tai padėtų nustatyti atributai. Nereikėtų numeruoti eilučių – tai irgi sutaupytų lokalizuotojui laiko“.

Skirtingus ekspertų įvertinimus visų pirma galima paaiškinti ekspertų darbo patirtimi skirtingose lokalizavimo darbo grupėse, skirtingu lokalizavimo darbo stiliumi. Vieni laiko, kad įprastas pirminis išteklių eilučių vertimas yra tiesioginis vertimas be lokalizuotojo papildomų pastangų atspėti kontekstą, kiti laiko, kad net pirminio išteklių vertimo metu lokalizuotojas bando ieškoti konteksto remiantis ir intuicija, ir paieška programoje, išskirti įvairius termino atvejus, esančius žodynuose, taiko pseudovertimą (pvz., eilučių numeraciją). Remiantis antruoju darbo stiliumi akivaizdu, kad vertimo sąnaudos sumažėtų, nes viskas, ko bando ieškoti lokalizuotojas savarankiškai, būtų pateikta su eilutės kontekstine informacija – atributais.

5.2.5.5. Konteksto parengimas kitai abėcėlinei fleksinei kalbai (rengia autorius)

Įvertinant, kokią dalį konteksto parengimo darbo sąnaudų sudarytų konteksto parengimas kitai abėcėlinei fleksinei kalbai, kai kontekstą rengia autorius, ekspertų A ir B nuomonės sutapo, o eksperto C atsakymas skiriasi nuo A ir B atsakymų 3 kartus (25 lentelė).

Ekspertas A pakomentavo savo atsakymą taip: „Reikėtų tik peržiūrėti atributus ir gal kai kuriuos pašalinti, kad be reikalo jų nereikėtų skaityti vertėjui, o gal vieną kitą ir pridėti. Dešimtadalį reikėtų laikyti atsargiu vertinimu. Tiksliau būtų ne daugiau, kaip vieną dešimtadalį“.

Eksperto B atsakymas ir paaiškinimas iš esmės sutapo su eksperto A atsakymu ir komentaru.

Ekspertas C įvertino, kad papildomos kalbos konteksto parengimas sudarytų „maždaug trečdalį buvusio darbo rengiant kontekstą: rengiant kontekstą jau bus daug padaryta, reikės tik sutikrinti, ar naujai fleksinei kalbai nereikia papildomų atributų ar komentarų. Kadangi sunku pasakyti, kokia yra ta kita kalba, be to, autorius gali neturėti tos kalbos žinių, tai pateikiu įvertinimą su atsarga. Gal tam tikroms kalboms rengiant kontekstą iš viso nereikės nieko papildomo daryti, tik peržiūrėti“.

Galima daryti išvadą, kad į šį klausimą ekspertams buvo sunkiausia atsakyti, kadangi neišvardijamos konkrečios kalbos. Dėl to buvo bandoma „apsidrausti“ ir šiek tiek padidinti darbo sąnaudas dėl tam tikrų nenumatytų atvejų, kurių gali būti rengiant kontekstą nepažįstamai kalbai.

5.2.5.6. Konteksto parengimas (rengia lokalizuotojas arba naudotojas)

Įvertinant, konteksto parengimo darbo sąnaudų santykį, kai rengia lokalizuotojas arba patyręs naudotojas, palyginus su atveju, kai kontekstą rengia autorius, gauti panašūs ekspertų atsakymai (25 lentelė).

Ekspertas A pakomentavo savo įvertinimą taip: „Atliekant šį darbą reikėtų išsiaiškinti tuos pačius dalykus, kaip ir lokalizuojant programą nenaudojant šio metodo (vertimą, intensyvų testavimą, vertimo koregavimą), tik rezultatą užrašyti kitu pavidalu – ne lokalizuotas eilutes, o atributinį medį. Todėl vertinu taip, kaip ir K1“.

Ekspertas B įvertino, kad lokalizuotojui arba patyrusiam naudotojui prireiktų maždaug 2–3 kartus daugiau darbo sąnaudų. Paaiškinimas panašus į K1 atsakymo paaiškinimą. Konteksto parengimo darbas būtų panašus į vertimo derinimo (kiekvienos eilutės paieškos ir konteksto išsiaiškinimo) darbą.

Ekspertas C įvertino panašiai, kaip ir ekspertai A ir B (darbo sąnaudos padidėtų 3 kartus), tačiau pastebėjo, kad jei vyktų lokalizuotojo ir autoriaus bendradarbiavimas rengiant kontekstinę informaciją, tai kontekstą būtų galima parengti žymiai sparčiau.

5.2.5.7. Konteksto parengimas kitai abėcėlinei fleksinei kalbai (rengia lokalizuotojas arba naudotojas)

Ekspertų A ir B įvertintas darbo sąnaudų santykis buvo toks pats, kaip ir vertinant K6 klausime minimų darbų santykius (25 lentelė). Įvertinimų pagrindimas yra analogiškas 5.2.5.6 skyrelyje pateiktiems atitinkamų ekspertų komentarams.

Eksperto C įvertinimas skiriasi daugiau kaip 2,5 karto nuo vidutinio ekspertų A ir B įvertinimo. Jis pažymėjo, kad darbo sąnaudos rengiant kontekstą kitai abėcėlinei fleksinei kalbai autoriui arba lokalizuotojui (patyrusiam naudotojui) yra maždaug vienodos. Atsakymo pagrindimas yra tas, kad „kontekstas yra jau parengtas vienai iš kalbų, t. y. formalių gramatikų kurti iš naujo nereikia, bet reikia įsigilinti į formalųjį aprašą, sužiūrėti atributus. Peržiūrėti esamą formalųjį aprašą autoriui būtų sparčiau negu lokalizuotojui, o peržiūrėti atributų reikšmes būtų lengviau padaryti žmogui, kuris gerai moka tą kalbą, kuriai perkeliamas kontekstas, negu autoriui. Dėl to darbo sąnaudos maždaug vienodos“.

Matome, kad ekspertai A ir B daugiau dėmesio skyrė formaliojo aprašo nagrinėjimui, o šį darbą būtų paprasčiau atlikti programos autoriui. Ekspertas C daugiau reikšmės suteikė atributų reikšmių peržiūrai ir kalbos, kuriai perkeliamas kontekstas, žinojimui, dėl to darbo sąnaudos buvo įvertintos kaip beveik vienodos.

5.2.5.8. Ekspertų pastabos ir siūlymai dėl metodo ir jo taikymo

Šiame skyrelyje pacituosime ekspertų atsakymus į K8 klausimą (žr. 5.2.3 sk.), kuriame reikėjo pateikti bendras pastabas ir siūlymus dėl šiame darbe siūlomo metodo ir jo taikymo.

Ekspertas A: „Kontekstinių sąlygų išsiaiškinimas ir formalizavimas neabejotinai prisidės prie lokalizavimo kultūros – sudarys sąlygas iš karto daryti gerą produktą, užuot pradžioje darius prastą ir po to jį taisyti (netgi po to, kai jis patenka pas naudotoją). Manau, kad pasiūlytas metodas vertingas ne tik tuo, kad naudą duoda tiesiogiai jį naudojant (t. y. gamybiniu aspektu), bet ir tuo, kad jis duoda pradžią tolesniam kontekstinių sąlygų formalizavimui, ypač sąlytyje su dialogo formalizavimu, o taip pat dialogo terminijos sisteminimui (t. y. moksliniu aspektu). Tai įvertinti kiekybiškai vargu ar įmanoma, bet kokybiškai – pasakyti, kad apčiuopiama nauda bus, galima drąsiai.“.

Ekspertas B: „Manau, ateityje metodas gali turėti geras perspektyvas išplėsti jį ir pritaikyti automatizuotam programinės įrangos sąsajos tekstų vertimui. Konteksto atributų, reikalingų automatizuoto vertimo priemonių pritaikymui programinės įrangos vertimui, man atrodo, dar niekas nėra nagrinėjęs“.

Ekspertas C: „Be abejonės, išteklių konteksto pateikimo metodo pritaikymas palengvintų lokalizuotojų darbą ir pagerintų lokalizuotų produktų kokybę: nereikėtų daugelį kartų kartoti vertimo, eilučių paieškos grafinėje sąsajoje, eilučių taisymo ir testavimo ciklo. Nereikėtų taikyti pseudovertimo arba pagalbinės eilučių numeracijos. Teisingą vertimą būtų galima parinkti nuo pat pradžių“.

Taigi ekspertų pastabos apie metodą yra iš esmės teigiamos. Yra nusakytos galimos metodo plėtros ateityje kryptys: pritaikyti formalizuojant dialogą, sisteminant dialogo su kompiuteriu terminiją, automatizuojant išteklių eilučių vertimą.

5.2.5.9. Metodo tiriamųjų aspektų įvertinimas remiantis ekspertų atsakymais

Remiantis ekspertų atsakymais į K1–K7 klausimus (žr. 5.2.3 sk.) ir 23 lentelėje pateiktomis įvairių tiriamųjų aspektų įvertinimo formulėmis gauti šie galutiniai tiriamųjų aspektų įvertinimai (26 lentelė).

26 lentelė. Metodo tiriamųjų aspektų įvertinimai remiantis ekspertų atsakymais

Tiriamasis aspektas	Įvertinimas		
	Ekspertas A	**Ekspertas B**	**Ekspertas C**
Testavimas ir vertimo koregavimas neturint konteksto	*300*	*250*	*350*
Konteksto parengimas (programos autorius)	*80*	*100*	*100*
Testavimas ir vertimo koregavimas (turint kontekstą)	*120*	*75*	*105*
Vertimas turint kontekstą	*150*	*80*	*50*
Konteksto parengimas kitai abėcėlinei fleksinei kalbai (programos autorius)	*8*	*10*	*30*
Konteksto parengimas (lokalizuotojas ar naudotojas)	*240*	*250*	*300*
Konteksto parengimas kitai abėcėlinei fleksinei kalbai (lokalizuotojas ar naudotojas)	*24*	*25*	*30*

Remiantis 24 lentelėje pateiktais skaičiavimo principais ir ekspertų vertinimais, suformuota metodo taikymo pagrindinių atvejų lentelė (27 lentelė).

27 lentelė. Dviejų metodo taikymo variantų ir lokalizavimo netaikant metodo įvertinimas remiantis ekspertų atsakymais

Tiriamasis aspektas	Įvertinimas		
	Ekspertas A	**Ekspertas B**	**Ekspertas C**
Programos lokalizavimas netaikant metodo	*400*	*350*	*450*
Programos lokalizavimas pritaikius metodą vienai kalbai (kontekstą rengia autorius)	*350*	*255*	*255*
Programos lokalizavimas pritaikius metodą antrai ir kiekvienai kitai fleksinei abėcėlinei kalbai (pritaiko autorius)	*278*	*165*	*185*
Programos lokalizavimas pritaikius metodą vienai kalbai (kontekstą rengia lokalizuotojas ar naudotojas)	*510*	*405*	*455*
Programos lokalizavimas pritaikius metodą antrai ir kiekvienai kitai kalbai (kontekstą rengia lokalizuotojas arba naudotojas)	*294*	*180*	*185*

Išskirkime 3 pagrindinius metodo taikymo atvejus:

M0 – lokalizavimas netaikant metodo.

M1 – lokalizavimas taikant metodą, kai kontekstą rengia programos autorius.

M2 – lokalizavimas taikant metodą, kai kontekstą rengia lokalizuotojas arba patyręs programos naudotojas.

Remiantis 27 lentelėje pateiktais įvertinimais, gauname darbo sąnaudas, reikalingas programai lokalizuoti M0, M1 ir M2 atvejais vienai ir dviems fleksinėms abėcėlinėms kalboms.

Taikant metodą lokalizuojamieji ištekliai pateikiami su kontekstine informacija, kuri padeda iš karto parinkti taiklų kokybišką vertimą, todėl nereikia atskirų sąnaudų kontekstui ieškoti ir tuo atveju, kai skirtingoms kalboms programą lokalizuoja skirtingi lokalizuotojai

(kaip dažniausiai ir yra dabartinėje lokalizavimo praktikoje), darbo sąnaudų kiekvienai papildomai fleksinei abėcėlinei kalbai prisideda maždaug po lygiai.

Kaip pasekmė, supaprastintos lokalizavimo darbo sąnaudų (y) priklausomybės nuo kalbų skaičiaus (x), kurioms lokalizuojama programa, M0–M2 atvejais gali būti išreikštos šiomis lygtimis:

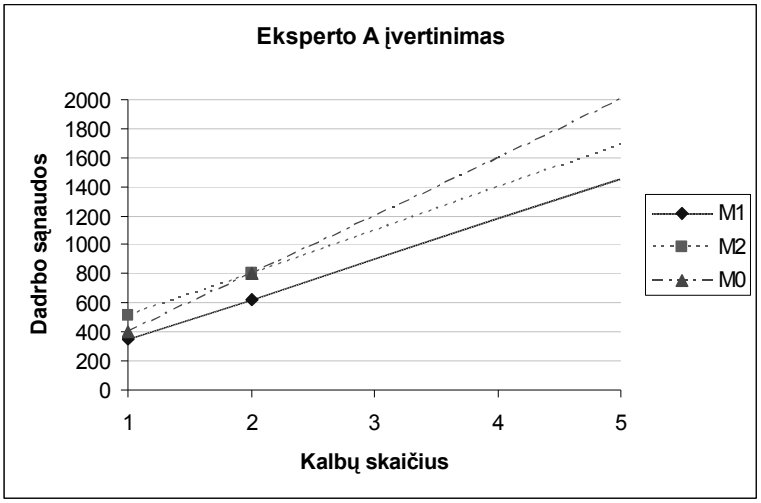
M0: $100(a_1+1)\boldsymbol{x} - \boldsymbol{y} = 0$

M1: $100(b_1b_4+b_3+a_1b_2)\boldsymbol{x} - \boldsymbol{y} + 100(b_1 - b_1b_4) = 0$

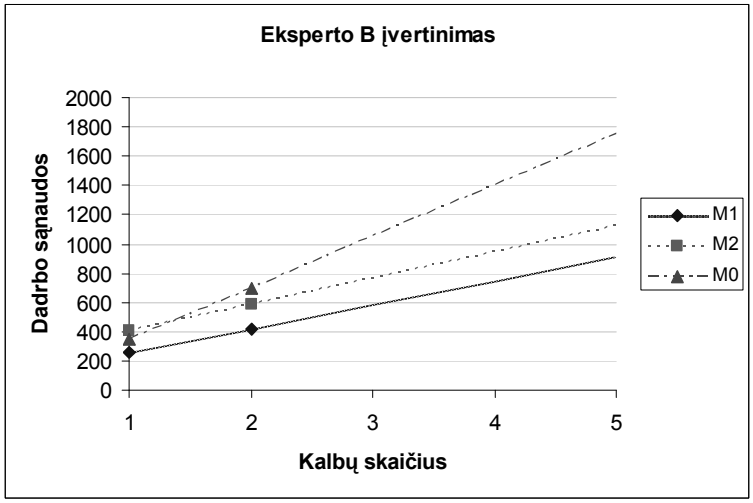
M2: $100(b_1b_4c_2+b_3+a_1b_2)\boldsymbol{x} - \boldsymbol{y} + 100(b_1c_1 + b_3 + a_1b_2) = 0,$

čia a_1, b_i, c_j, ($i = 1, 2, \ldots, 4$; $j = 1, 2$) atitinka 23 lentelėje pateiktus koeficientus.

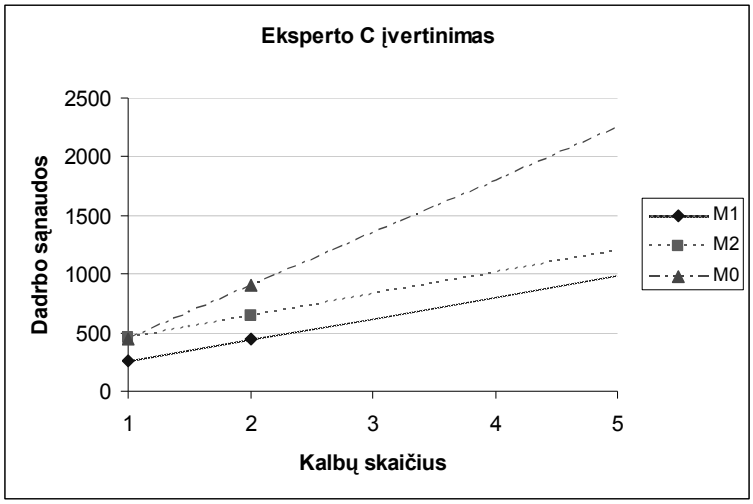
Kiekvieno eksperto įvertinimais paremtos lokalizavimo darbo sąnaudų priklausomybė nuo kalbų skaičiaus, kurioms taikomas metodas, pavaizduotos grafiškai 29, 30 ir 31 paveiksluose. Visais atvejais skaičiuojamos kokybiško lokalizuojamųjų išteklių vertimo darbo sąnaudos.



29 pav. Lokalizacijų kalbų skaičiaus ir darbo sąnaudų priklausomybė, remiantis eksperto A įvertinimais



30 pav. Lokalizacijų kalbų skaičiaus ir darbo sąnaudų priklausomybė, remiantis eksperto B įvertinimais



31 pav. Lokalizacijų kalbų skaičiaus ir darbo sąnaudų priklausomybė, remiantis eksperto C įvertinimais

Jei programa lokalizuojama tik vienai kalbai, tai metodą taikyti yra prasminga tik tuo atveju, jei kontekstą rengia programos autorius.

Darant išvadą, kada yra prasminga taikyti metodą, imsime M1 ir M2 darbo sąnaudų maksimalią reikšmę ir ją lyginsime su M0 darbo sąnaudų atitinkama reikšme.

Remiantis eksperto A įvertinimais, metodą apsimoka taikyti jei programa lokalizuojama 2,03 kalboms ($x_A = 2{,}03$). Remiantis eksperto B įvertinimais, metodą apsimoka taikyti, jei programa lokalizuojama 1,32 kalboms ($x_B = 1{,}32$). Remiantis eksperto C įvertinimais, metodą apsimoka taikyti, jei programa lokalizuojama 1,02 kalboms ($x_C = 1{,}02$).

Siekdami patikimo rezultato, apibendrindami įvertinimus remsimės ne vidutiniais ekspertų įvertinimų reikšmėmis, bet imsime iš ekspertų įvertinimų gautą pesimistiškiausią variantą:

$$\lceil \max(x_A, x_B, x_C) = 3 ,$$

čia $\lceil$ – apvalinimo iki artimiausio didesnio sveikojo skaičiaus operacija.

Remiantis apklaustų ekspertų įvertinimais, šiame darbe pasiūlytą metodą verta taikyti, kai programa lokalizuojama ne mažiau kaip 3 kalboms. Tokiu atveju lokalizavimo darbo sąnaudos žymiai sumažėja palyginus su lokalizavimo darbo sąnaudomis, kai šis metodas netaikomas.

5.3. Išvados

Šioje darbo dalyje pritaikėme pasiūlytą lokalizuojamųjų išteklių metainformacijos (konteksto) formalizavimo metodą lokalizuotinos programos fragmentui bei atlikome metodo ekspertinį vertinimą. Remiantis ekspertų programinės įrangos lokalizavimo ir internacionalizavimo srityse apklausos metu gautais atsakymais nustatyta, kad:

1. Lokalizuojamos programinės įrangos testavimo ir išteklių vertimo koregavimo darbai reikalauja maždaug tris kartus didesnių darbo sąnaudų negu tekstinių lokalizuojamųjų išteklių vertimas.

2. Lokalizuojamos programinės įrangos testavimo ir išteklių vertimo koregavimo darbų sąnaudos sumažėtų tris kartus, jeigu lokalizuojamieji tekstiniai ištekliai būtų pateikiami su metainformacija, nusakančia kontekstą.
3. Metainformacijos formalizavimas (konteksto parengimo) darbas, remiantis šiame darbe pasiūlytu metodu, reikalautų maždaug tokių pačių darbo sąnaudų, kaip pirminis lokalizuojamųjų tekstinių išteklių vertimas neturint kontekstinės informacijos.
4. Metainformacijos formalizavimo (konteksto parengimo) darbo sąnaudos būtų 2,5–3 kartus mažesnės, jei šį darbą atliktų lokalizuojamos programos autorius, negu lokalizuotojas ar patyręs programos naudotojas.
5. Jeigu programinę įrangą planuojama lokalizuoti tik vienai kalbai, tai pritaikius darbe pasiūlytą metodą, lokalizavimo darbo sąnaudos sumažėtų nežymiai ir tik tuo atveju, jei kontekstinę lokalizuojamųjų išteklių informaciją formalizuotų programos autorius, o ne lokalizuotojas ar patyręs naudotojas.
6. Remiantis apibendrintais pagal maksimalią lokalizavimo darbo sąnaudų reikšmę rezultatais, lokalizuojamųjų išteklių metainformacijos formalizavimo metodą yra prasmė taikyti tada, kai programinę įrangą planuojama lokalizuoti ne mažiau kaip trims fleksinėms abėcėlinėms kalboms. Tokiu atveju metainformaciją gali formalizuoti arba programos autorius (nuo 3 kalbos bendrosios lokalizavimo darbo sąnaudos pradeda mažėti bent 24%), arba lokalizuotojas ar patyręs programos naudotojas (nuo 3 kalbos bendrosios lokalizavimo darbo sąnaudos pradeda mažėti bent 8%).
7. Remiantis ekspertų bendrosiomis pastabomis apie lokalizuojamųjų išteklių metainformacijos formalizavimo metodą, metainformacijos (kontekstinių sąlygų išsiaiškinimas) pateikimas ir išsiaiškinimas prisidėtų prie programinės įrangos lokalizavimo kultūros, sudarytų sąlygas iš karto daryti gerą produktą, palengvintų lokalizuotojų darbą. Prie ateities darbų būtų galima priskirti metodo išplėtimą iki programinės įrangos išteklių tekstų vertimo automatizavimo.

BENDROSIOS IŠVADOS IR REZULTATAI

1. Remiantis programinės įrangos lokalizavimo analizės ir lokalizavimo proceso tyrimo bei praktinio internetinės programinės įrangos lokalizavimo metu sukauptomis ir susistemintomis žiniomis sukurtas lokalizuojamųjų išteklių metainformacijos formalizavimo metodas, kuris:
   1.1. Išplečia formaliąsias atributines gramatikas papildydamas trimis pagrindinėmis naujovėmis: 1) gramatika papildoma nauju atributų tipu, 2) į gramatiką įtrauktos priemonės skaičiuojamiesiems medžio mazgų atributams papildyti įvedamais iš išorės, 3) valdoma konteksto galiojimo sritis.
   1.2. Leidžia sistemingai pateikti lokalizuojamuosius išteklius, kas sudaro galimybę pastebėti internacionalizavimo klaidas ankstyvojoje – programinės įrangos specifikavimo, projektavimo – stadijoje.
   1.3. Skirtas įtraukti į išteklius juos aprašančią kontekstinę informaciją, reikalingą lokalizavimo kokybei gerinti ir formaliai apibrėžiančią vietą, kurioje tam tikra eilutė bus rodoma programos grafinėje sąsajoje, ryšius tarp susijusių eilučių, kiekvienos eilutės teksto semantiką nusakančią informaciją.
2. Patobulintų formaliųjų atributinių gramatikų metodas yra reikšmingas ir teoriniu informatikos kaip mokslo aspektu, nes pateikia naują požiūrį į formaliąsias atributines gramatikas ir išplečia jas naujomis priemonėmis, ir praktiniu taikomumu ne tik sparčiai kintančiai internetinei programinei įrangai lokalizuoti, bet ir kitoms kalbinėms sritims modeliuoti.
3. Išanalizavus ir palyginus lokalizuojamųjų išteklių atskyrimo metodus, pateikimo formatus ir lokalių modelius, numatyti pagrindiniai sprendimai, kurie galėtų pagerinti programinės įrangos lokalizacijų kokybę:
   3.1. Lokalizuojamųjų tekstinių išteklių pateikimo formatų išplėtimas įvedant: a) išteklių eilučių papildymą kontekstine metainformacija, b) statiškai ir dinamiškai valdomų dialogo tekstų vietos grafinėje programos sąsajoje žymėjimus, c) parametrizuotų ir komponuojamų eilučių valdymą, d) kalbinę semantinę informaciją apie išteklių eilutes.
   3.2. Programinės įrangos kultūros elementų, kurių nėra visuotinai naudojamuose lokalių formaliuosiuose aprašuose, atskiras nagrinėjimas: naudotojo sąsajos tekstų formų, mažųjų ir didžiųjų raidžių derinimas, frazių komponavimas, nevienareikšmių terminų reikšmių, kalbos dalių parinkimas ir kt.
   3.3. Lokalizuojamųjų tekstinių išteklių pateikimo formatų ir kultūros elementų problemų sprendimą tikslinga perkelti į programinės įrangos projektavimo ar net specifikavimo stadiją užuot ieškant jų sprendimo lokalizavimo metu.
4. Ištyrus programinės įrangos lokalizavimo procesą ir internetinės programinės įrangos kultūros elementus:
   4.1. Parengta internetinės programinės įrangos kultūros elementų klasifikacija, skirta: a) programų projektuotojams – atkreipti dėmesį į jų projektuojamos programinės įrangos kultūros elementų adaptavimo procesą, b) programų lokalizuotojams – adaptuoti daugiau kultūros elementų, aptikti internacionalizavimo klaidas, c) tyrėjams ir testuotojams – vertinti programinę įrangą lokalizavimo požiūriu.
   4.2. Nustatyta, kad dabartinėje lokalizavimo praktikoje dominuoja lokalizavimo būdas, kai dėl nepakankamos kontekstinės informacijos apie lokalizuojamuosius išteklius tokių išteklių vertimas sudaro tik nedidelę dalį visų lokalizavimo darbų: vertimo

metu padarytos klaidos ir netikslumai koreguojami testavimo metu, kas reikalauja maždaug tris kartus didesnių sąnaudų negu išteklių vertimas.

5. Atlikus pasiūlyto metodo ekspertinį vertinimą, nustatyta, kad:
   5.1. Pritaikius patobulintų formaliųjų gramatikų metodą lokalizuojamų išteklių kontekstui formalizuoti, testavimo ir lokalizuotų išteklių koregavimo darbų sąnaudos sumažėtų tris kartus.
   5.2. Lokalizuojamųjų išteklių metainformacijos formalizavimo metodą prasminga taikyti tada, kai programinę įrangą planuojama lokalizuoti ne mažiau kaip trims fleksinėms abėcėlinėms kalboms.

LITERATŪRA

[AA85] Ashton, A. H., & Ashton, R. H. Aggregating subjective forecasts: some empirical results. Management Science, 31, 1985, p. 1499–1508.

[AD07] Auer, S. Dick, E. When does a difference make a difference? A snapshot on global icon comprehensibility. In: Jacko, J.A. (Ed.) Human-computer interaction, Pt 2, proc. of 12th International Conference on Human-Computer Interaction. Lecture Notes in Computer Science, 4551, 2007, p. 3–12.

[AFO08] Ågerfalk, P. J., Fitzgerald, B., Olsson, H. H., Conchúir, O. Benefits of Global Software Development: The Known and Unknown. In: Wang, Q., Pfahl, D., Raffo, D. (Eds.) Making Globally Distributed Software Development a Success Story, proc. of International Conference on Software Process‘ 2008, Leipzig. Lecture Notes in Computer Science, 5007, 2008, p. 1–9.

[AGH90] Alexin, Z., Gyimóthy, T., Horváth, T., Fábricz, K. Attribute grammar specification for a natural language understanding interface. Attribute Grammars and their Applications, LNCS, Vol. 461/1990, 1990, p. 313–326.

[App92] Apple Computer. Guide to Macintosh Software Localization. Addison Wesley, Reading, Mass, 1992.

[Ash86] Ashton, R. H. Combining the judgments of experts: how many and which ones? Organizational Behavior and Human Decision Processes, 38, 1986, p. 405–414.

[Atk01] Atkin, S. E. A Framework for Multilingual Information Processing. Doctoral dissertation. Florida Institute of Technology, Melbourne, Florida, 2001.

[Bar06] Bargary, K. Translation Web Services – an Implementation for the IGNITE Project. Localisation focus. The international journal for Localisation. Vol. 5, Issue 1, 2006, p. 22–23.

[BB07] Batra, S., Bishu, R. R. Web usability and evaluation: Issues and concerns. In: Aykin, N. (Ed.) Usability and Internationalization, pt 1, proc. Global and Local User Interfaces 4559. Lecture Notes in Computer Science, 2007, p. 243–249.

[BKO02] Boswell, D., King, B., Oescbger, I., Collins, P., Murphy, E. Creating Applications with Mozilla. O‘Reilly & Associates, 20002.

[Boi05] Boitet, Ch. Message Automata for messages with variants, and methods for their translation. LNCS 3406, 2005

[Bow05] Bowker, L. Productivity vs Quality? A pilot study on the impact of translation memory systems. Localisation focus. The international journal for Localisation. Vol. 4, Issue 1, 2005, p. 13–20.

[BR00] Budescu, D., Rantilla, A. K. Confdence in aggregation of expert opinions. Acta Psychologica, 104, 2000, p. 371–398.

[BT89] Van de Burgt, S. P., Tilanus, P. A. J. Attributed ASN.1. In: Proceedings of the 2nd International Conference on Formal Description Techniques, FORTE 89, Amsterdam, 1989, p. 298–310.

[Ca98] Carey, J. Creating global software: A conspectus and review. Interacting with Computers, Special Issue: Shared values and shared interfaces, 9, 4, 1998, p. 449–465.

[CGP90] Crimi, C., Guercio, A., Pacini, G., Tortora, G., Tucci, M. Automating visual language generation. IEEE Transactions on Software Engineering, Vol. 16, Iss. 10, 1990, p. 1122–1135.

[Cha07] Chavan, A. L. Smart strategies for creating culture friendly products and interfaces. In: Aykin, N. (Ed.) Usability and Internationalization, pt 1, proc. Global and Local User Interfaces 4559. Lecture Notes in Computer Science, 2007, p. 27–32.

[Cle89] Clemen, R. T. Combining forecasts: a review and annotated bibliography. International Journal of Forecasting, 5, 1989, p. 559–583.

[Co02] Collins, R.W. Software localization for Internet software, issues and methods. IEEE software, 19 (2): 74, 2002.

[CT07] Chen, C.-H., Tsai, C.-Y. Designing user interfaces for mobile entertaining devices with cross-cultural considerations. In: Aykin, N. (Ed.) Usability and Internationalization, pt 1, proc. Global and Local User Interfaces 4559. Lecture Notes in Computer Science, 2007, p. 37–46.

[Dav07] Davis, M. Locale data markup language (LDML). Unicode Technical Standard #35. Unicode, Inc., 2007, http://unicode.org/reports/tr35/tr35-8.html [žiūrėta 2009-11-20]

[DC01] Deitsch A, Czarnecki D. Java Internationalization. O'Reilly and Associates, 2001.

[DG06] Dagienė, V., Grigas, G. Quantitative evaluation of the process of open source software localization. Informatica vol. 17, no. 1, 2006, p. 3–12.

[DGJ04] Dagienė, V., Grigas, G., Jevsikova, T. Programinės įrangos lietuvinimas: patirties analizė. Informacijos mokslai. 31, 2004, p. 171–185.

[DGJ08] Dagienė, V., Grigas, G., Jevsikova, T. Enciklopedinis kompiuterijos žodynas. Vilnius: TEV, 2008.

[DJ05] Dagienė, V., Jevsikova, T. Virtualiosios mokymosi aplinkos lokalizavimo požiūriu Lietuvos matematikos rinkinys. ISSN 0132-2818. T. 45, spec. Nr., 2005, p. 197–202.

[DJ08] Dagienė, V., Jevsikova, T. Cultural Elements in Internet Software Localization. In: Brezillon, P., Coppin, G., Lenca, P. (Eds.) HCP-2008 – Third International Conference on Human Centered Processes. Delft, Netherlands, June 8–12, 2008, p. 275–288.

[DJ09] Dagienė, V., Jevsikova, T. Cultural Elements in Internet Software Localization. In: Lenca, P., Brézillon, P., Coppin, G. (eds.) Revue d'intelligence artificielle. Human-centered processes – Current trends. Volume 23 – no 4/2009. Hermes – Lavoisier, 2009, p. 485–501.

[DL04] Dagienė, V., Laucius, R. Internationalization of open source software: Framework and some issues. In: Boyle, T., Oriogun, P., Pakstas, A. (Eds.) Proc. 2nd International Conference Information Technology – Research and Education (ITRE 2004), 2004, p. 204–207.

[DM93] Deransart, P. Maluszynski, J. A Grammatical View of Logic Programming. The MIT Press, Cambridge, Mass, 1993.

[DMP07] Drepper, U. , Meyering, J., Pinard, F., Haible, B.: GNU gettext tools, version 0.17. Native Language Support Library and Tools. Free Software Foundation, 2007.

[Dür03] Dürst M. J. Internationalization of XML – Past, Present, Future. XML Conference&Exposition. Pensylvania, December 7–12, 2003.

[ED97] Evers, V. and Day, D. The role of culture in interface acceptance, In Human Computer Interaction: INTERACT‘97, Sydney, Chapman and Hall, 1997.

[Emb77] Ember C.R., Ember M., Anthropology. Prentice-Hall, Englewood Cliffs, NJ, 1977.

[Ess00] Esselink, B. A practical guide to localization. John Benjamins, 2000.

[Ev01a] Evers, V. Cultural aspects of user interface understanding. Doctoral dissertation. Institute of Educational Technology, The Open University, 2001.

[Ev01b] Evers V., “Cross-Cultural Understanding of Graphical Elements on the DirectED Website”. Smith, A (Ed) Proc. of Ann. Workshop on Cultural Issues on HCI, Univ. of Luton, 2001.

[FG03] Ford, G., Gelderblom, J. H. The effects of culture on performance achieved through the use of human-computer interaction. Proceedings of SAICSIT, 2003, p. 218–230.

[FH05] Frimannsson, A.; Hogan, J. M. Adopting Standards-based XML File Formats in Open Source Localisation. Localisation Focus. The International Journal for Localisation. Vol. 4, Issue 4, 2005, p. 9–23.

[FHC03] Faltstrom P., Hoffman P., Costello, A., Internationalizing Domain Names in Application. Network Working Group, Request for Comments: 3490, 2003.

[FMJ92] Farrow, R., Marlowe, T. J., Yellin, D. M. Composable attribute grammars: Support for modularity in translator design and implementation. In: Conference Record of the 19th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM, New York, 1992, p. 223–234.

[Fro92] Frost, R.A. Constructing programs as executable attribute grammars. The Computer Journal, Vol. 35, Iss. 4. Special issue on models and architectures, 1992, p. 376–389.

[Gas07] Gaspari, F. The role of online MT in Webpage translation. A thesis for a degree of Doctor of Philosophy. Universtity of Manchester, 2007.

[GP07] Grigas G., Pedzevičienė S. Errors in Personal and Cities Names in Users Registration Data of Chat Applications. Information sciences, issue: 42–43, 2007, p. 141–144.

[Gri03] G. Grigas. Programinės įrangos vertimo į lietuvių kalbą dabartinė situacija, problemos ir jų sprendimai. Informacijos mokslai, ISSN 1392-0561, 26, 2003, p. 251–256.

[Gri08] G. Grigas. Kompiuterijos leksika ir terminija. Santalka, t. 16, Nr. 2, 2008, p. 31–39.

[Gri98] Grigas, G. Lietuviškų rašmenų panaudojimo kompiuteriuose ir jų tinkluose problemos. Iš: Lituanistika pasaulyje šiandien: darbai ir problemos. Baltos lankos, t. 3, p. 65–72, Vilnius, 1998.

[Guz04] Guzman, R. Tools Review: Catalyst and Terminology Wizard. Localisation focus. The international journal for Localisation. Vol. 3, Issue 1, 2004, p. 18–19.

[GZ00] Grigas, G., Zalatorius, J. (2000) Cultural and economic aspects of software localization. Baltic IT Review, nr. 6, p. 61–64.

[Hal07] Hall, B. Resources in Microsoft .NET. Ccaps Newsletter, 2007, p. 1–6.

[Hal90] Hall E., Hall M., Understanding Culture Differences. Maine, Intercultural Press, 1990.

[HH05] Hofstede, G., Hofstede, G. J. Cultures and organization. McGraw Hill, New York, 2005.

[HH90] Hall, E., Hall, M. Understanding Culture Differences. Maine, Intercultural Press, 1990.

[HHP04] Hogan J. M., Ho-Stuart C., Pham B., Key challenges in software internationalisation. In: Proceedings of the Australasian Workshop on Software Internationalisation (AWSI 2004). ACSW Frontiers 2004: Conferences on Research and Practice in Information Technology Volume 32, Sydney: Australian Computer Society, 2004, p. 187–194.

[HK87] Hudson, S. E., King, R. Implementing a user interface as a system of attributes. In: Proceedings of the ACM SIGSOFT/SIGPLAN Software Engeneering Symposium on Practical Software Development Environments. ACM SIGPLAN Notes, 22, 1, 1987, p. 143–149.

[Hof01] Hofstede, G. Culture's Consequences: Comparing Values, Behaviors, Institutions, and Organizations across Nations, 2nd edn. McGraw-Hill, New York, 2001.

[Hof07] Hofmann, P. Localising and internationalising graphics and visual information. IEEE Transactions on Professional Communication vol. 50, no. 2, 2007, p. 91–92.

[Hof91] Hofstede, G. Cultures and Organization. McGraw Hill, New York, 1991.

[HR92] Horwitz, S., Reps, T. The use of program dependence graphs in software engineering. In Proceedings of the 14th international conference on Software engineering. ACM New York, NY, USA, 1992, p. 392–411.

[IBM94] IBM, National Language Design Guide Volume 1, National Language Support Reference Manual, 4th Ed., 1994.

[Ya07] Yang, Y. X. Extending the user experience to localized products. In: Aykin, N. (Ed.) Usability and Internationalization, pt 2, proc. Global and Local User Interfaces 4560. Lecture Notes in Computer Science, 2007, p. 285–292.

[YF09] Yang, S. L., Fu, Ch. Constructing confidence belief functions from one expert. Expert Systems with Applications. 36, 2009, p. 8537–8548.

[YJM05] Yijun Yu; Jianguo Lu; Mylopoulos, J.; Weiwei Sun; Jing-Hao Xue; D'Hollander, E.H. Making XML document markup international. Software - Practice and Experience vol. 35, no. 1, 2005, p. 1–14.

[Jev03] Jevsikova, T. Programų adaptavimas lietuviškai lokalei. Informacinės technologijos 2003, Konferencijos pranešimų medžiaga, ISBN 9955-09-335-8, Kaunas: Technologija, 2003, p. I-(8–14).

[Jev06] Jevsikova T. Internationalization and Localization of Web-based Learning Environment. In: R. Mittermeir (Ed.) Informatics Education – the Bridge Between Using and Understanding Computers. LNCS, 4226, 2006, p. 310–319.

[JS86] Jones, L.G., Simon, J. Hierarchical VLSI design systems based on attribute grammars. In: Proceedings of the 13th ACM SIGACT-SIGPLAN symposium on Principles of programming languages, 1986, p. 58–69.

[Kan95] Kano, N. Developing International Software: For Windows 95 and Windows NT, Microsoft, Redmond, WA, 1995.

[KK93] Kaiser, G., Kaplan, S. M. Parallel and distributed incremental attribute evaluation algorithms for multiuser software development environments. ACM Transactions on Software Engineering and Methodology. Vol. 2, Iss. 1, 1993, p. 47–92.

[Knu68] Knuth, D. E. Semantics of context-free languages. Theory of Computing Systems, vol. 2, no. 2, 1968, p. 127–145.

[KS97] Kokkots, S., Spyropoulos, C.D. An architecture for designing internationalized software. Software Technology and Engineering Practice. Proceedings. 8th IEEE International Workshop on incorporating Computer Aided Software Engineering. London, 1997, p. 13–21.

[Lan05] Lanier, C.R. Linux and the appeal to cultural values. IEEE Technology and Society Magazine vol. 24, no. 4, 2005, p. 12–17.

[Lau03] Laucius, R. Lokalės, jų sandara ir ypatumai. Informacinės technologijos 2003, Konferencijos pranešimų medžiaga, ISBN 9955-09-335-8, Kaunas: Technologija, 2003, p. I-(1–7).

[Lau07] Laucius, R. Kompiliatorių internacionalizacija. Daktaro disertacija. Technologijos mokslai, informatikos inžinerija (07 T). Vilniaus Gedimino technikos universitetas, Matematikos ir informatikos institutas, Vilnius, 2007.

[LD03] Laucius, R., Dagienė, V.. Raštinės programinės įrangos „OpenOffice.org“ adaptavimas lokalės normoms. Informacijos mokslai, 26, 2003, p. 240–245.

[LRS74] Lewis, P., Rosenkrantz, D., Stearns, R. Attributed translations. Journal of Computer and System Sciences, 9, 1974, p. 279–307.

[LVS92] Lewi, J., De Vlaminck, K., Steegmans, E., Van Horebeek, J. Software Development by LL(1) Syntax Description. John Wiley & Sons, New York, 1992.

[LW03] Lepouras, G.; Weir, G.R.S. Subtitled interaction: complementary support as an alternative to localization. International Journal of Human-Computer Studies vol. 59, no. 6, 2003, p. 941–957.

[MG01] Marcus, A., Gould, E. W. Cultural dimensions and global Web design. Experience Intelligent design. AM+A, 2001.

[Mic09] Microsoft Corporation. Go Global Development Center. http://msdn.microsoft.com/en-us/goglobal/bb688117.aspx [žiūrėta 2009-09-12]

[Mic90] Microsoft. Microsoft windows international handbook for software design, Microsoft Corporation, Redmond, Washington, 1990.

[Mmv09a] Mokslas, mokslininkai, visuomenė. Lietuviškų kompiuterinių programų sąvadas. Klausimynas. Lokalizuotų programų ekspertizė, 2009. http://mokslasplius.lt/lkps/?q=node/3 [žiūrėta 2009-09-19]

[Mmv09b] Mokslas, mokslininkai, visuomenė. Lietuviškų kompiuterinių programų sąvadas. Lokalizuotos programos (ekspertizės), http://mokslasplius.lt/lkps/?q=node/2 [žiūrėta 2009-09-19]

[Moo06] Moore, L. CLDR: The common locale data repository – locales for the world. In: Kelly, K., Schaler, R. (Eds.) The 11th annual internationalisation and Localisation

conference organised by the Localisation research centre. The Localisation factory. 25–26 October, 2006, Dublin, Ireland. p. 31–43, 2006.

[MP00] Manousopoulou, A. G., Papakonstantinou, G. A Grammatical Approach to User Model Resolution for Electronics. In: Proceedings of the Third Workshop on Attribute Grammars and their Applications, 2000, p. 117–124.

[MT05] Mushtaha, A., De Troyer, O. Towards Localising e-Learning Websites. Localisation Focus. The International Journal for Localisation. Vol. 4, Issue 4, 2005, p. 6–8.

[Mus04] Musale, S. Getting more from translation memory. Localisation focus. The international journal for Localisation. Vol. 3, Issue 1, 2004, p. 9–10.

[Nev00] Neven, F. Extensions of Attribute Grammars for Structured Document Queries. Research Issues in Structured and Semistructured Database Programming, LNCS, Vol. 1949/2000, 2000, p. 99–117.

[Nie90] Nielsen, J. (Ed.) Designing User Interfaces for International Use, Elsevier Science, Amsterdam, 1990.

[NWT05] Nichols, D. M.; Witten, I. H.; Te Taka Keegan; Bainbridge, D.; Dewsnip, M. Digital libraries and minority languages. New Review of Hypermedia and Multimedia vol. 11, no. 2, 2005, p. 139–55.

[Oas08] OASIS. XLIFF version 1.2. OASIS Standard, 2008.

[Paa95] Paaki, J. Attribute Grammar Paradigms – A High-Level Methodology in Language Implementation. ACM Computmg Surveys, Vol. 27, No, 2, 1995.

[Pal85] Palionis, J. Kalbos mokslo pradmenys. Vilnius, 1985.

[PC99] Psaila, G., Crespi-Reghizzi, S. Adding Semantics to XML. In: D. Parigot and M. Mernik (eds.), Procedings of the Second Workshop on Atribute Grammars and their Applications, Amsterdam, The Netherlands, 1999, p. 113–132.

[Qin07] Qingxin Shi. Cultural Usability: The Effects of Culture on Usability Testing. Human-Computer Interaction – INTERACT 2007, LNCS, 4663, 2007, p. 611–616.

[RB07] Reinecke, K., Bernstein, A. Culturally adaptive software: Moving beyond internationalization. In: Aykin, N. (Ed.) Usability and Internationalization, pt 2, proc. Global and Local User Interfaces 4560. Lecture Notes in Computer Science, 2007, p. 201–210.

[RB93] Russo, P. and Boor, S. How fluent is your interface? Designing for international users, In INTERCHI‘93 Human Factors in Computing Systems, Amsterdam, NY: ACM, 1993.

[Reg09] Reghizzi, S. C. Formal Languages and Compilation. Texts in Computer Science. Springer, 2009.

[RLH98] Ramalho, J.C., Lopes, A.R., Henriques, P.R. Generating SGML specific editors: from DTDs to Atribute Grammars. In: Markup Technologies Conference, Chicago, 1998. p.

[RT89] Reps, T, W., Teitelbaum, T. The Synthesizer Generator – A System for Constructing Language-Based Editors. Springer-Verlag, New York, 1989.

[Sca02] Scattergard, D. Documentation Localisation Costs. Localisation Focus, vol. 1, issue 2, 2002.

[Sch02] Schäler, R.: The Cultural Dimensions in Software localization. Localisation Focus, vol. 1, isssue 2, 2002.

[ShK88] Shinoda, Y, Katayama, T. Attribute grammar based programming and Its environment. In: proceedings of the 21st Hawaii International Conference on System Sciences. IEEE Computer Society Press, 1988, p. 612–620.

[SJ07] Schadewitz, N., Jachna, T. Introducing new methodologies for identifying design patterns for internationalization and localization. In: Aykin, N. (Ed.) Usability and Internationalization, pt 2, proc. Global and Local User Interfaces 4560. Lecture Notes in Computer Science, 2007, p. 228–237.

[Sul01] O'Sullivan, P. A Paradigm for Creating Multilingual Interfaces. Doctoral Dissertation. University of Limerick, 2001.

[Sul99] O'Sullivan, P. User Interface and Translation. Results of a survey completed for Lotus Development Communication Products group, 1999.

[SV03] Souchon, N., Vanderdonckt, J. A Review of XML-compliant User Interface Description Languages, LNCS 2844, 2003.

[Tay92] Taylor, D. Global Software: Developing Applications for the International Market. Springer-Verlag New York, 1992.

[TLB94] Teasley, B., Leventhal, L., Blumenthal, B., Instone, K. and Stone, D. Cultural diversity in user interface design: Are intuitions enough? SIGCHI Bulletin, 26, 1, 1994, p. 36–40.

[Tro97] Trompenaars, F. Riding the waves of culture – Understanding cultural diversity in business. Nicholas Brealey Publishing, London, 1997.

[UHP93] Uren, E., Howard, R., Perinotti, T. Software internationalization and localization. An Introduction. Van Nostrand Reinhold, 1993.

[Uni03] Unicode. The Unicode Standard, Version 4.0. Boston, MA, Addison-Wesley, 2003.

[Uni07] Unicode, Inc. Unicode CLDR Project: Common Locale Data Repository, 2007. http://unicode.org/cldr/ [žiūrėta 2009-09-27]

[VG90] Vaske J., Grantham C. E., Socializing the Human-computer Environment. Ablex, Norwood, NJ, 1990.

[Was04] Wassmer, T. Comparing Tools Used in Software Localization. Localisation reader, 2004, p. 17–21.

[Wat85] Watt, D. A. Modular description of programming languages. Rep. A-8 1-734, Computer Science Division-EECS, Univ. of California, Berkeley, Calif, 1985.

[WM07] Wan Mohd Isa, W. A. R., Md Noor, N. L. Incorporating the cultural dimensions into the theoretical framework of website information architecture. In: Aykin, N. (Ed.) Usability and Internationalization, pt 1, proc. Global and Local User Interfaces 4559. Lecture Notes in Computer Science, 2007, p. 212–221.

STANDARTAI

ISO 31-1:1992. Quantities and units – Part 1: Space and time.

ISO 3166-1:1997. Codes for the representation of names of countries and their subdivisions – Part 1: Country codes

ISO 639-1:2002. Codes for the representation of names of languages – Part 1: Alpha-2 code

ISO 8601: 2004. Data elements and interchange formats – Information interchange – Representation of dates and times.

ISO 8879:1986: Information processing. Text and office systems. Standard Generalized Markup Language (SGML), (Geneva: ISO, 1986)

ISO/IEC 14652:2004. Information technology – Specification method for cultural conventions.

ISO/IEC 9899:1999. Programming languages. C.

ISO/IEC 9945-1:2003 Information technology – Portable Operating System Interface (POSIX) – Part 1: Base Definitions.

ISO/IEC 9945-2:2003. Information technology – Portable Operating System Interface (POSIX). Part 2: System Interfaces.

LST ISO/IEC 15897:2001: Lietuvos standartas. Informacijos technologija. Kultūros elementų registravimo procedūros (tapatus ISO/IEC 15897:1999).

Tatjana Jevsikova

INTERNETINĖS PROGRAMINĖS ĮRANGOS LOKALIZAVIMAS

Daktaro disertacija
Fiziniai mokslai (P 000)
Informatika (09 P)
Informatika, sistemų teorija (P 175)

_____

Išleido ir spausdino AB MINTIS,
Z. Sierakausko g. 15, LT-03105 Vilnius
Interneto svetainė http://www.mintis.eu/
Tiražas 20 egz.