Lokalizuojamųjų programinės įrangos išteklių metainformacijos formalizavimo metodas

Tatjana Jevsikova

Matematikos ir informatikos instituto doktorantė
Institute of Mathematics and Informatics, Doctoral student
Akademijos g. 4, LT-08663 Vilnius
Tel. (8 5) 272 92 07, faks. (8 5) 272 92 09
El. paštas: Tatjanaj@ktl.mii.lt

Valentina Dagienė

Matematikos ir informatikos instituto vyriausioji mokslo darbuotoja, skyriaus vadovė, profesorė, daktarė
Institute of Mathematics and Informatics, Chief Research Scientist, Head of the Department, Prof., PhD
Akademijos g. 4, LT-08663 Vilnius
Tel. (8 5) 272 92 07, faks. (8 5) 272 92 09
El. paštas: Dagiene@ktl.mii.lt

Programinės įrangos lokalizavimas – vienas svarbesnių veiksnių kompiuterių taikymo srityje. Pasaulyje beveik sutartinai išskiriami du lokalizavimo komponentai: programos adaptavimas (lokalės elementų pritaikymas) ir dialogų (pranešimų, meniu užrašų ir kt.) vertimas ir adaptavimas. Straipsnyje nagrinėjama, kaip būtų galima paspartinti dialogų tekstų vertimą ir pagerinti lokalizacijų kokybę. Tai ypač reikalinga lokalizuojant internetinę programinę įrangą, kai dažnai tenka rengti naujas programos versijas, daryti atnaujinimus. Siūloma remtis formaliosiomis atributinėmis gramatikomis ir jomis aprašyti lokalizuojamuosius išteklius, per atributus įtraukiant lokalizavimo požiūriu naudingą kontekstinę informaciją. Aptariamas programinės įrangos išteklių parengimas lokalizuoti, lokalizuojamųjų išteklių struktūra, pateikimo formatai, ypatumai. Straipsnio pabaigoje pateikiami lokalizuojamųjų išteklių formaliosios gramatikos sudarymo bendrieji principai.

Įvadas

Viena iš svarbių kultūriniu ir ekonominiu požiūriu programinės įrangos savybių yra jos sąsajos su žmogumi pateikimas naudotojo kalba kuo natūraliau, tarsi būtų sukurta jo kultūrinėje terpėje. Programinės įrangos lokalizavimo poreikis atsirado tada, kai prasidėjo masinis jos eksportas į kitas 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 (Grigas, 1998).

Daugelis lokalizavimo problemas tiriančių mokslininkų pabrėžia, kad programinės įrangos lokalizavimo darbus galima suskirstyti į dvi dideles dalis (Esselink, 2000; O'Sullivan, 20001; Dagienė, Grigas, Jevsikova, 2004, 2005; Yang, 2007 ir kt.):

- Programos adaptavimas konkrečiai kalbinei ir kultūrinei terpei (koduotės, skaičių formatai, datų ir laiko formatai, dokumentų formos ir kt.);
- Dialogo tekstų (įskaitant elektroninius žinynus, naudotojo vadovus) vertimas ir adaptavimas.

Pirmosios dalies problemos dažniausiai sprendžiamos naudojantis formaliais lokalės aprašais.

2004 metais Unikodo konsorciumas ėmėsi vykdyti CLDR (*Common Locale Data Repository*) projektą lokalės duomenų saugyklai kurti (Unicode ..., 2009). Čia pateikiamos priemonės bendriems programinėje įrangoje naudojamiems įvairių pasaulio lokalių duomenims specifikuoti, taip pat kaupiami lokalių duomenys. Lokalių duomenų mainams naudojamas XML formatas – lokalės duomenų žymėjimo kalba LDML, duomenys laisvai prieinami internete.

Gerokai daugiau problemų kyla imantis spręsti antrosios dalies – dialogo tekstų – lokalizavimą. Šių duomenų daug, programos dažnai atnaujinamos, reikia nuolatos būti pasirengus išversti po keletą eilučių, kurios ištrauktos iš konteksto neretai esti sunkiai suprantamos.

Pagrindinės problemos, su kuriomis susiduriama lokalizuojant programas, ir jų priežastys išanalizuotos straipsniuose (Dagienė, Grigas, Jevsikova, 2004; Jevsikova, 2006). Čia pateiksime galimą jų sprendimo metodą, kuris remiasi formaliosiomis atributinėmis gramatikomis.

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

Metodo veiksmai (principai) – lokalizavimui skirti programos ištekliai – pateikiami hierarchiniu pavidalu, atspindinčiu lokalizuojamų eilučių ryšius su programos grafinės sąsajos elementais. Hierarchinei struktūrai išreikšti naudojamos formaliosios gramatikos, o semantikai aprašyti – atributai. Bendras išteklių modelis aprašomas modifikuota atributine gramatika. Tokia gramatika sukuriama konkrečiai programai ir gali padėti ne tik sumažinti internacionalizavimo ir lokalizavimo klaidų skaičių programoje, bet ir tvarkingiau projektuoti grafinę programos naudotojo sąsają.

Lokalizuojamųjų išteklių struktūra, jų atskyrimo metodai ir formatai

Planuojamos lokalizuoti programinės įrangos kūrėjai programą turi tam parengti, t. y. ją internacionalizuoti. Vienas svarbiausių šio parengiamojo darbo etapų – lokalizuojamųjų išteklių atskyrimas nuo pirminių programos tekstų. Tai visų tekstų, grafikos, garsų, lokalės elementų, pagalbinių parametrų ir kt. elementų, pateikiamų kompiuterio ekrane programai veikiant, iškėlimas į atskirus failus.

Lokalizuojamieji programos ištekliai gali būti tekstiniai arba dvejetainiai – jų pateikimas priklauso nuo programavimo kalbos, kuria parašyta programinė įranga, naudojamo kompiliatoriaus, platformos, kuriai projektuojama programinė įranga, taip pat naudoto išteklių atskyrimo metodo. Apžvelgsime keletą pagrindinių programinės įrangos internacionalizacijos tipų – išteklių atskyrimo metodų.

R. Laucius (2007) disertacijoje skiria tris internacionalizacijos tipus (jie kartu gali būti laikomi ir programinės įrangos išteklių atskyrimo metodais): (1) internacionalizacija kompiliavimo metu; (2) internacionalizacija susaistymo metu; (3) internacionalizacija vykdymo metu. Iš jų saistymo ir vykdymo tipai pastaruoju metu laikomi vienu tipu.

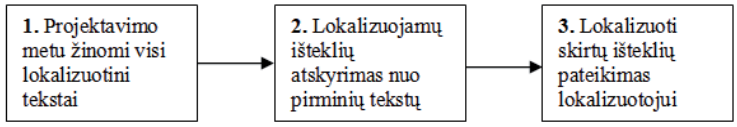
Internacionalizacija kompiliavimo metu – tai programos projektavimo būdas, kai lokalizavimui skirti ištekliai neatskiriami nuo pirminio programos teksto: teksto eilutės, kurios bus matomos kompiuterio ekrane, įkompiliuojamos į pirminį programos tekstą. Tuomet lokalizuojant programą daromos atskiros kopijos kiekvienai lokalei, pirminiame tekste randamos lokalizuotinos eilutės ir programa perkompiliuojama. Perkompiliuojant programą atsiranda pavojus pažeisti pirminį tekstą. Taigi tokį metodą vadinti internacionalizacija yra daugiau nei simboliška.

Internacionalizacija susaistymo ir vykdymo metu pasižymi tuo, kad lokalizavimui skirti ištekliai yra atskiriami nuo programos pirminio teksto. Norint įtraukti lokalizuotus išteklius, programos pirminio teksto nereikia perkompiliuoti. Lokalizavimui skirti ištekliai yra įkompiliuojami į vykdomąsias programas arba į vykdymo metu prie programų prijungiamas bibliotekas, išteklių paketus, duomenų bazes. Tokiu būdu parengtas programas galima suskirstyti į lokalizuotas ir daugiakalbes (Kokkots, Spyropoulos, 1997, p. 15). Lokalizuotos programos realizuojamos saistant atitinkamas išteklių bibliotekas, kurios pakeičia originalias programos funkcijas, neatitinkančias

lokalės. Daugiakalbės programos naudoja išorinius pranešimų ir išteklių failus, kurie gali būti platinami atskirai ir pakeičiami ar prijungiami prie programos ją vykdant.

Kiekvienas didesnis programinės įrangos gamintojas sukuria savo išteklių atskyrimo metodą ir išteklių pateikimo lokalizavimui formatą. Atviruųjų programų kūrėjai taip pat kuria ir naudoja savo formatus. Tačiau šiuo metu naudojami formatai pateikia lokalizuotinus išteklius be konteksto arba tik su menkomis užuominomis apie kontekstą.

Pirmame paveiksle pavaizduotas programinės įrangos parengimo lokalizuoti procesas: projektavimo metu žinomas visų tekstų ir kt. nuo lokalės priklausomų elementų kontekstas programoje, tada lokalizuojamieji ištekliai atskiriami nuo programos pirminio teksto ir tam tikru formatu pateikiami lokalizuotojams. Tarp 2 ir 3 žingsnio prarandamas lokalizuojamųjų išteklių ryšys su kontekstu programoje.



1 pav. ***Programinės įrangos parengimo lokalizuoti proceso schema***

Apibendrinti lokalizuojamųjų išteklių analizės rezultatai, įvardijant pagrindinius šiuo metu naudojamus lokalizuojamųjų išteklių atskyrimo metodus ir atitinkamus failų formatus, pateikiami lentelėje.

Lentelė. ***Lokalizuojamųjų išteklių atskyrimo metodai ir atitinkami failų formatai***

Išteklių atskyrimo metodas	Pagrindiniai 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

Lokalizuojamųjų išteklių ypatumai ir konteksto svarba

Ankstesniame skyrelyje apžvelgti programinės įrangos lokalizuojamų išteklių atskyrimo metodai ir pateikimo formatai yra panašūs tuo, kad tekstiniai lokalizuojamieji ištekliai pateikiami 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. Teksto eilutės – tai ne tik ekrane rodomi tekstai programai veikiant, bet ir kai kurių funkcijų parametrų reikšmės, šriftų, koduočių ir kt. vardai, nuorodos į dvejetainius objektus, 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) sąryšių su kitomis susijusiomis eilutėmis ir naudotojo grafinės sąsajos elementais, kuriuose eilutės bus vaizduojamos programai veikiant (atsižvelgiant į kontekstą).

Programinės įrangos tekstai (aibės E elementai) yra lakoniški, atsieti nuo konteksto, juose gausu naujų terminų (kurių gali dar nebūti kalboje, į kurią lokalizuojama programa). Todėl programos lokalizuotojui tenka spręsti įvairias problemas.

Viena pagrindinių problemų ta, kad lokalizuotojas mato tik atskirus žodžius ar frazes be konteksto, t. y. lokalizuodamas programą, žmogus dirba su dialogo eilučių duomenų baze (aibės L elementais, o tam tikrais atvejais – tik su aibės E elementais, kadangi V aibė gali būti sudaryta iš skaitinių identifikatorių, neteikiančių jokios lokalizavimo požiūriu svarbios informacijos).

Lokalizuojamų tekstų kontekstas pamatomas tik programai veikiant arba nagrinėjant programos pirminius tekstus (jeigu tai leidžia daryti programos licencija). Dalis programos dialogo tekstų (kai kurie ekspertai mano, kad jų yra apie

10–15 proc.) atsiranda tik esant ypatingoms situacijoms (klaidoms, kitų programų poveikiui), kurias sudėtinga arba neįmanoma sumodeliuoti testuojant lokalizuotą programą. Dėl minėtų veiksnių programos dialogo tekstų vertimo ir adaptavimo sąnaudos yra kelis kartus didesnės (kai kurie autoriai nurodo, jog net tris kartus), palyginti su rišlaus teksto vertimu.

Tik atpažinus ir įvertinus lokalizuojamųjų išteklių eilučių kontekstą galima tinkamai išversti ar adaptuoti. Lokalizuojamųjų išteklių kontekstas gali būti kelių lygių. Išskirsime du lygius:

- Visos lokalizuojamos eilutės (aibės E elemento) kontekstas (pavyzdžiui, programos komponento, kuriame vartojama eilutė, pavadinimas; lango, kuriame vartojama eilutė, identifikavimas; konkretus lango elementas, kuriame rodoma eilutė; ryšiai su kitomis eilutėmis; situacijos, kuriai esant eilutė pateikiama naudotojui, įvardijimas; valdantysis eilutės-frazės žodis; vidinė programos funkcija, kuri realizuoja eilute pavadintą komandą).
- Lokalizuojamos eilutės dalies (segmento, parametro, sąvokos) kontekstas (pavyzdžiui, eilutėje pavartoto parametro ryšiai su kitomis eilutės dalimis; eilutės dalių tarpusavio priklausomybė ir formų derinimas; konkrečių žodžių semantikos paaiškinimai: ar tai veiksmažodis, ar daiktavardis ir pan.).

Lokalizuojamųjų išteklių metainformacija, nusakanti kontekstą ir struktūrą

Kai kuriose programose dalį kontekstą nusakančių atributų galima išskirti iš V aibės elementų – vardo struktūros, kitą dalį – iš failų ir katalogų vardų ir lokalizavimo komentarų.

Eilučių suskirstymas į lokalizavimui skirtus failus gali teikti naudingos kontekstinės informacijos lokalizuotojui: failas ar jų grupė atitinka tam tikros programų teminės dalies eilučių rinkinį. Taip yra „Mozilla“ šeimos programose, virtualiojoje mokymo aplinkoje „Moodle“, tačiau kai kurių programų autoriai ištraukia visas lokalizavimui skirtas eilutes į vieną failą (pvz., „LeMill“ aplinka). Taigi bendros tendencijos nėra.

Šiek tiek informacijos galima gauti iš eilutės vardo (identifikatoriaus), jame kartais (tai priklauso nuo programuotojo ir vardų sudarymo susitarimo) nurodoma, kuriame sąsajos elemente bus panaudota eilutė. Tačiau toks nurodymas nėra sistemingas net toje pačioje programoje arba nėra numatytas išteklių atskyrimo metodu, pvz., .RC formato eilučių sekcijoje eilutės numeruojamos, „gettext“ metodu atskirtuose ištekliuose eilutės vardas paprastai sutampa su pačios eilutės tekstu.

Kitas metainformacijos šaltinis – tai komentarai lokalizuotojams. Jie paprastai skirti tam tikrai eilutei paaiškinti, nurodant sudėtingesnes lokalizavimui vietas, susitarimus ar ribojimus. Tačiau komentavimas taip pat priklauso nuo programos projektuotojų, komentarams trūksta sistemingumo.

Minėtos metainformacijos nepakanka net jeigu ji ir yra nurodoma. Lokalizuotojui būtų daug paprasčiau parinkti tinkamą eilutės atitikmenį lokalizacijos kalba, jeigu būtų žinoma eilutės 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. Šias problemas iš dalies bandoma spręsti XLIFF lokalizuojamų išteklių formatu, tačiau šio formato specifikacijoje priemonės išsamesniam kontekstui nurodyti yra ribotos. Dėl to siūlomas lokalizuojamųjų išteklių formalizavimo metodas, kad eilučių pateikimas atspindėtų ryšius tarp jų ir programos grafinės naudotojo sąsajos, o eilutės turėtų atributus, kurie leistų pagerinti lokalizacijų kokybę.

Lokalizuojamųjų išteklių atributinės gramatikos sudarymo principai

Atributines gramatikas kaip priemonę programavimo kalbų semantikai formalizuoti pasiūlė D. E. Knuth (1968). Mes naudosime modifikuotas atributines gramatikas lokalizuojamiesiems ištekliams aprašyti. Pateiksime pagrindinius principus, kurie nusako lokalizuojamųjų išteklių formalizavimo metodą.

- Laisvojo konteksto gramatika G = <N, T, P, S> (N – neterminalinių simbolių aibė, T – terminalinių simbolių aibė, P – išvedimo taisyklių aibė, S – pradinis simbolis) sudaroma konkrečiai programai, atspindint jos grafinės sąsajos struktūrą ir siejant ją su lokalizuotinomis eilutėmis.
- Gramatika pagal jos simbolių vartojimą sudaroma iš dviejų pagrindinių dalių: tai programos grafinės naudotojo sąsajos struktūra ir lokalizuojamos eilutės bei jų struktūra.
- Neterminalinių simbolių naudojimas: programos grafinės naudotojo sąsajos kiekvienam elementui įvardyti neterminaliniai simboliai parenkami remiantis programos grafinės naudotojo sąsajos specifika; įvedamas neterminalinis simbolis visai eilutei iš lokalizuojamųjų išteklių įvardyti; 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 dinamiškai. Šis principas buvo pasirinktas todėl, kad parametrai kelia nemažai problemų lokalizavimo metu ir jiems būdingi atskiri atributai. Tokiu būdu įgyvendinamas antro lygio konteksto įvedimas (žr. ankstesnį skyrelį). Jei meniu ar grafinės sąsajos elemente (valdiklyje) rodomas tekstas, suduriamas iš kelių eilučių, tai gramatikos medyje jos atsiduria šalia. Tam įvedami atskiri neterminaliniai simboliai grafinės sąsajos elementui ir visai eilutei.
- Terminaliniai simboliai – tai lokalizuojamos eilutės ar eilučių dalys (segmentai). Jei eilutėje nėra parametrų, tai visa eilutė atitinka terminalinį simbolį. Jei eilutėje yra parametrų, tai eilutė skaidoma į segmentus, kuriuos skiria parametrai.
- Visiems gramatikos simboliams priskiriami atributai, skirti lokalizavimo požiūriu svarbiai semantinei informacijai nusakyti, pvz., visos eilutės aprašas, valdantysis frazės žodis, veiksmažodinė ar daiktavardinė frazė, grafinio elemento plotis, tipas ir t. t.
- Apibrėžiamos semantinės taisyklės kiekvieno gramatikos simbolio atributų reikšmėms skaičiuoti.

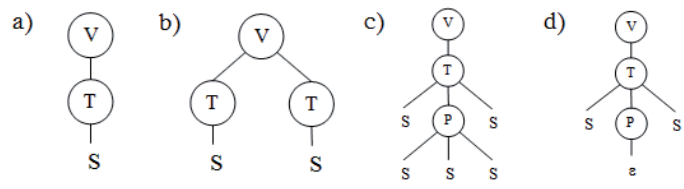
Parodysime, kaip formuojamos lokalizuojamųjų išteklių eilutės, pavyzdžiui, rodomos grafinės naudotojo sąsajos elementuose – valdikliuose.

1. Ištisa eilutė paprastajame valdiklyje. 2a paveiksle pavaizduota eilutė be parametro (vienas segmentas S, mazgas V atitinka valdiklio neterminalinį simbolį, mazgas T – visos teksto eilutės neterminalinį simbolį).

2. Viename valdiklyje rodomos kelios sudurtos eilutės. Kiekviena tokia eilutė turi savo vardą lokalizuojamuose ištekliuose, t. y. eilutės pateikiamos atskirai (2b pav.).

3. Lokalizavimui skirtoje eilutėje pavartotas vienas ar keli parametrai. Galimi atvejai:

- 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 (2c pav.).
- Parametro vietoje įrašoma reikšmė, kurios nėra lokalizuojamuosiuose ištekliuose, ji nežinoma iš anksto, bet įrašoma dinamiškai, vykdant programą. Pavyzdžiui, kokių nors objektų skaičius, naudotojo vardas ir



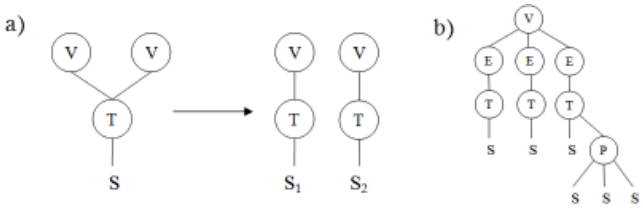
2 pav. ***Gramatikos simbolių parinkimas atsižvelgiant į lokalizuojamų eilučių pateikimo valdikliuose atvejus***

pavardė ir pan. Atvejis, kai eilutės viduje yra vienas parametras su dinamiškai parenkama reikšme, pavaizduotas 2d paveiksle. Simbolis P turi atributą – parametro duomenų tipą.

1. Valdiklyje rodoma viena iš kelių eilučių, atsižvelgiant į kontekstą, kuris išaiškėja vykdant programą (vaizdavimas sutampa su 2 punktu, 2b pav.).

2. Ta pati lokalizuojamųjų išteklių eilutė naudojama keliuose naudotojo grafinės sąsajos elementuose. Tokia eilutė būtų dubliuojama (3a pav.), 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.

3. Sudėtinis valdiklis – valdiklis, kuriame pagal jo paskirtį rodomos kelios eilutės, pavyzdžiui, išskleidžiamasis sąrašas. Atvejis, kai sudėtinis valdiklis turi tris paprastus elementus (E), o viename iš jų pavartota eilutė su parametru, pavaizduotas 3b paveiksle.



3 pav. ***Gramatikos simbolių parinkimas, kai ta pati eilutė naudojama keliuose valdikliuose ir sudėtinio valdiklio atveju***

Kaip įprasta projektuojant šiuolaikinę programinę įrangą, naudojamas komponentinis gramatikos kūrimo principas. Galima atskirai nagrinėti tam tikrą programos komponentą, kuris turi savo autonominę grafinę naudotojo sąsają (pvz., el. pašto programos adresų knygos tvarkymo komponentas). Tada programos komponento grafinę naudotojo sąsają galima skaidyti į dalis, pavyzdžiui, nuostatų langą, pagrindinį meniu ir pan. Bendroji visos programos atributinė gramatika (AG_p) gaunama sujungus visų jos atskirų komponentų (jei yra) dalių atributines gramatikas, o dalinių gramatikų sąsaja pateikiama per jungiamų simbolių atributus.

Išvados

Lokalizavimo kokybę galėtų pagerinti konteksto – metainformacijos – įtraukimas į lokalizuojamuosius išteklius. Siūlomas atributinių gramatikų metodas lokalizuojamųjų išteklių kontekstui ir semantinei informacijai nurodyti skiriasi nuo atributinių gramatikų taikymų realizuojant programavimo kalbų transliatorius tuo, kad:

- naudojami išoriniai ir vidiniai atributai; išoriniai atributai priskiriami interaktyviu būdu;
- 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;
  - atributinė gramatika šiuo atveju kuriama 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.

LITERATŪRA

DAGIENĖ, V.; GRIGAS, G.; JEVSIKOVA, T. (2004). Programinės įrangos lietuvinimas: patirties analizė. *Informacijos mokslai*, t. 31, p. 171–185.

ESSELINK, B. (2000). *A practical guide to localization*. John Benjamins, 2000.

GRIGAS, G. (1998). Lietuviškų rašmenų panaudojimo kompiuteriuose ir jų tinkluose problemos. Iš: *Baltos lankos*, t. 3. Lituanistika pasaulyje šiandien: darbai ir problemos. Vilnius, p. 65–72.

YANG, Y. X. (2007). Extending the user experience to localized products. Iš: Aykin, N. (Ed.) *Usability and Internationalization*. Proc. Global and Local User Interfaces. Lecture Notes in Computer Science, 4560, p. 285–292.

JEVSIKOVA, T. (2006). Internationalization and Localization of Web-based Learning Environment. Iš: R. Mittermeir (Ed.) *Informatics Education – the Bridge Between Using and Understanding Computers*. Proc. ISSEP 2006, Lecture Notes in Computer Science, 4226, p. 310–319.

KNUTH, D. E. (1968). Semantics of context-free languages. *Theory of Computing Systems*, vol. 2, no. 2, p. 127–145.

KOKKOTS, S.; SPYROPOULOS, C.D. (1997). An architecture for designing internationalized software. Iš: *Software Technology and Engineering Practice*. Proc. 8th IEEE International Workshop on incorporating Computer Aided Software Engineering. London, p. 13–21.

LAUCIUS, R. (2007). *Kompiliatorių internacionalizacija*: daktaro disertacija. VGTU, MII, Vilnius.

O'SULLIVAN, P. A. (2001). *Paradigm for Creating Multilingual Interfaces*: Doctoral Dissertation. University of Limerick.

UNICODE, Inc (2009). Unicode CLDR Project [žiūrėta 2009 m. liepos 14 d.]. Prieiga per internetą: http://unicode.org/cldr/

FORMALIZATION OF SOFTWARE LOCALIZABLE RESOURCES' METAINFORMATION

Tatjana Jevsikova, Valentina Dagienė

S u m m a r y

Software localization is one of important tasks to ensure successful computer users' experience. Many experts identify two main components of software localization: 1) software adaptation (locale items and their adjustment to suit target language and culture) and 2) translation and adaptation of the dialog elements (program's messages, menu items, dialog boxes and their controls, etc.). The paper discusses how translation and adaptation of the dialog's text can be accelerated and how to raise the quality of software product localization. This is especially important when we deal with internet software which is frequently updated, and localizers must rapidly update their localization, translating new text strings which usually lack information on their context in the program's graphical user interface. We also discuss the main features and common structure of localizable software resources, their formats and preparation for localization. As a result, we suggest to apply a modified formalism of attribute grammars to describe localizable resources, taking graphical user interface as a basic grammar structure, localizable strings and their parts as terminal symbols, and using attributes to add important metainformation and context to the resources. The main principles of creation of such attribute grammars are presented.