VYTAUTAS MAGNUS UNIVERSITY
INSTITUTE OF MATHEMATICS AND INFORMATICS

Tatjana Jevsikova

INTERNET SOFTWARE LOCALIZATION

Summary of Doctoral Dissertation

Physical Sciences (P 000)
Informatics (09 P)
Informatics, Systems Theory (P 175)

Vilnius, 2009

The dissertation will be defended at the public meeting of the Scientific Council in the field of Informatics in the Seminars Room of the Institute of Mathematics and Informatics at 1 p. m. on December 28, 2009.
Address: Akademijos str. 4, LT-08663, Vilnius, Lithuania.

The summary of the dissertation was sent-out on November 26, 2009.
A copy of the doctoral dissertation is available for review at the M. Mažvydas National Library of Lithuania, the Library of the Institute of Mathematics and Informatics and the Library of the Vytautas Magnus University.

VYTAUTO DIDŽIOJO UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS INSTITUTAS

Tatjana Jevsikova

INTERNETINĖS PROGRAMINĖS ĮRANGOS LOKALIZAVIMAS

Daktaro disertacijos santrauka

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

Vilnius, 2009

Disertacija rengta 2005–2009 metais Matematikos ir informatikos institute

Disertacija ginama viešame fizinių mokslų, informatikos mokslo krypties tarybos posėdyje 2009 m. gruodžio 28 d. 13 val. Matematikos ir informatikos instituto seminarų kambaryje.
Adresas: Akademijos g. 4, LT-08663, Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta 2009 m. lapkričio 26 d.

Disertaciją galima peržiūrėti M. Mažvydo nacionalinėje bibliotekoje, Matematikos ir informatikos instituto ir Vytauto Didžiojo universiteto bibliotekose.

1. General Characteristics of the Dissertation

1.1. Scope and Relevance

Software localization is one of important tasks to ensure successful computer users' experience. Due to wide usage of the Internet across the world, it is very important to provide user with the virtual environment (software) that does not contradict natural cultural environment he or she belongs to. Therefore, reseach on software localization, improvement of localization process and raise of quality of localized products are extremely significant. This is especially important when we deal with internet software which is frequently updated, and localizers must rapidly update their localizations, translating new text strings which usually lack information on their context of use in the graphical user interface of the program.

In this work, the term “internet software” is used as a general term to address: 1) software, used to access internet resources, and operating on the client side (e.g. web browser), 2) software, implementing internet services, and operating on the server side (e.g. virtual learning environment).

Discussions on software localization usually point out two main parts of localization (e.g. [Ess00]): 1) software adaptation to the target locale (adjustment of character encoding, number formats, dates and times, document templates, etc.); 2) translation and adaptation of user interface (menu items, button labels, check boxes, error messages, etc., including online help and documentation).

Problems, met during the first part of localization (adaptation to the target locale), are usually solved by the means of formal locale definitions. The second part (translation and adaptation of user interface) is much more problematic due to large amount of text, frequent updates, and lack of context of user interface strings. Therefore, the primary translation of user interface texts can be treated only as a draft translation, which must be gradually corrected and improved during testing and modification of translation cycle, when the context of the dialog strings is being searched running the program and, according to it, primary translation is modified. Usually, localization bugs are found after localization process is completed and the product is released.

Another problem is not sufficient level of software internationalization. During software localization process, internationalization errors are usually identified as well. These errors should be corrected by the original software developers. However, because of continuous development and frequent release of software new versions, solutions of the problems are postponed for later versions.

Therefore, research on how to raise quality of localized dialog texts and reduce testing component of localization process, is very important.

1.2. Research Object

Software localization process, localizable resources and research on formal grammars.

1.3. Aim and Objectives

The aim of the dissertational work is to present a method to formalize metainformation of textual localizable resources of internet software, which would help to systematize localization process and raise quality of localizations, as a result of analysis of localization process and its problems.

The objectives of the dissertation are the following:

1. To anlyze current scientific and experimental achievements on software localization and internationalization.
2. To investigate software localization process, its problems, formats of presentation of localizable resources, to identify and systematize internationalization errors.
3. To propose a method to formalize metainformation of internet software textual localizable resources, that would contribute to raising quality of localized products.
4. To evaluate the suggested method of formalization of textual localizable resources, using expert opinion poll.

1.4. Research Methodology

The basis of research methodology, used in this work, made up analytic, generalizing, constructive and evaluative methods.

1.5. Scientific Novelty

Localization process has been investigated, the main cultural elements of internet software have been identified and classified, the method of formalization of textual localizable resources metainformation has been proposed.

The main novel aspects of the method:

1. Attribute grammars have been used to create the method, attribute grammars has been augmented with these tools: a) internal and external

attributes are used; b) computed attributes are complemented with attributes, entered from outside; c) controlled attribute context scope. Attribute grammars are applied in other way than usual programming language compilation: to formalize and carry contextual information from original software developer to localizer by means of attributes.

2. Formalization of information on the context in the graphical user interface and semantic linguistic information.
3. The method is suitable not only for static user interface strings, but for dynamic strings as well.

1.6. Defended Propositions

1. Existing locale models and localizable resources formats do not allow bringing enough information, needed to correctly translate those resources to other language (therefore, the quality of primary translation is low or expenditures of localization grow several times due to testing and correction of translation).
2. The method, based on modified attribute grammars, allows formalizing metainformation of textual localizable resources.
3. Additional work, needed to formalize metainformation of textual localizable resources, is covered if software is being localized to 3 or more languages.

1.7. Practical Importance

The main practical importance of the work is that the proposed method is designed to include the main contextual information into localizable resources, and therefore to raise quality of primary translation of localizable resources, as well as to reduce the expenditure of efforts, involved during localization testing and correction cycle. Moreover, the detection and correction of internationalization errors is moved to an earlier phase of software development. The evaluation of method, based on multicriteria evaluation by experts, has shown that applying the method, overall localization expenditures start considerably decreasing when software is localized (planned to be localized) into three or more languages.

1.8. Publication and Approbation

The results of the dissertational work were published in 12 scientific publications (6 of them in periodical peer-reviewed journals, 3 publications in ISI Proceedings, 3 publications in other scientific conferences proceedings).

The full list of publications on the topic of the dissertation is presented in the end of this Summary.

The results of the dissertation were presented via 14 presentations, given during 12 international and national conferences and seminars:

1. 3d International Conference on Human Centered Processes (HCP 2008). Delft, the Netherlands, June 8–12, 2008.
2. International Conference: Informatics in Secondary Schools: Evolution and Perspectives' 2006, Vilnius, Lithuania, November 7–11, 2006.
3. IFIP international conference "Informatics, Mathematics and ICT: a Golden Triangle", Boston, USA, June 27–29, 2007.
4. 2nd International Conference "Information Technology: Research and Education", London, UK, June 28 – July 1, 2004.
5. XLVIII Conference of Lithuanian Mathematicians (the conference was held by the Society of Lithuanian Mathematicians). Vilnius, Lithuania, June 27–28, 2007.
6. XLVI Conference of Lithuanian Mathematicians (the conference was held by the Society of Lithuanian Mathematicians). Vilnius, Lithuania, June 15–16, 2005.
7. XLIII Conference of Lithuanian Mathematicians (the conference was held by the Society of Lithuanian Mathematicians). Vilnius, Lithuania, June 17–18, 2002.
8. Information Technologies '2003, Kaunas, Lithuania, January 28–29, 2003.
9. "Kompiuterininkų dienos '2005" (the conference was held by the Lithuanian Computer Society). Klaipėda, Lithuania, Augus, 15–17, 2005.
10. Seminar at the Institute of Mathematics and Informatics: "On localization of Mozilla family software". Vilnius, Lithuania, January 24, 2007.
11. Seminar at the Institute of Mathematics and Informatics: "Mozilla family software internationalization level and localization". Vilnius, Lithuania, February 22, 2006.
12. "An importance of usage of Lithuanian language in information society". A seminar, held by the Information society development committee under the Government of Republic of Lithuania. Vilnius, December 11, 2007.

1.9. Structure of the Dissertation

The dissertation consists of five chapters, general conclusions and results, references, and glossary. The work includes 149 pages of text, 31 figures, and 27 tables. The dissertation is written in Lithuanian.

In the **first**, an introductory chapter, the statement of the problem, its relevance, research aim and objectives, its scientific novelty and practical importance, as well as work's approbation and publication are presented.

The **second** chapter is analytical. It is aimed to present results on the main concepts of software localization analysis, locale models, localizable resources separation methods and formats.

The **third** chapter deals with software localization process and its systematization, classification of cultural elements, met in internet software.

The **fourth** chapter presents a method of representation of software localizable resources, including metainformation on context, important during localization.

The **fifth** chapter is aimed to evaluate the method, proposed in the dissertation. The results, derived from experts' opinions, are presented.

2. ANALYSIS OF SOFTWARE LOCALIZATION

In this section, the results of analytical part of the dissertation (Chapter 2) are briefly presented.

2.1. Software localization and internationalization

Research on software localization and internationalization has been carried since the last decade of XX century. The concept of *software localization* is defined differently by different authors. In this work we will use such definition of software localization: a process of modifying software to be suitable for a particular cultural and linguistic environment. Software localization should be considered from the start of the software development process. Only properly developed (i.e. internationalized) software can be localized well. Therefore, internationalization and localization are closely related concepts.

Many authors, working in the field of software localization, usually point out two main parts of localization (e.g. [Ess00]): 1) software adaptation to the target locale (adjustment of character encoding, number formats, date and time formats, document templates, etc.); 2) translation and adaptation of user interface (menu items, button labels, check boxes, error messages, etc., including online help and documentation).

The features, related to these two main parts of software localization have been implemented gradually. Some authors point out 7 levels of software localization: 1) translate nothing; 2) translate documentation and packaging only; 3) enable code; 4) translate software menus and dialogs; 5) translate online help and tutorials; 6) add support for locale-specific hardware; 7) customize features for locale. The more levels are implemented, the more risk is taken and the more return is [Ca98]. Other authors point out three levels of software localization: 1) ability to process locale's texts; 2) adaptation of main locale-specific elements (e.g. date, time formats); translation and localization of all the dialog components in full extent [Gri98].

Problems, arising during the first part of localization (adaptation to the target locale), are usually solved by the means of formal locale definitions. Therefore, the main locale models are analyzed and compared in this work with an aim to find out the elements that are important in localization, but not included in existing formal locale definitions. The second part (translation and adaptation of user interface) is problematic due to large amount of text and lack of context of user interface strings. Therefore, the ways of separation of such texts and their representation formats are analyzed.

2.2. Comparison of Locale Models

According to international standard ISO/IEC 15897 [ISO99], locale is "the definition of the subset of a user's information technology environment that depends on language, territory, or other cultural customs". Locale is usually identified by the language, using two-letter language code [ISO02], and by territory, using two-letter territory code [ISO97]. Results of comparison of the most widely used locale models (POSIX [ISO03], FDCC [ISO04], Java [DC01], ISO/IEC 15897 [ISO99], and CLDR [Uni07]) are presented in Table 1.

Table 1. Comparison of locale models, according to formalized locale elements

Locale element	**POSIX**	**FDCC**	**Java**	**ISO/IEC 15897: 1999**	**CLDR**
Character classification and case conversion	+	+	+	+	+
Collation, ordering, searching	+	+	+	+	+
Monetary formatting	+	+	+	+	+
Numeric (non-monetary) formatting	+	+	+	+	+
Date and time formats, names of weekdays, months, eras	+	+	+	+	+
Time zones and their names	–	–	+	+	+
Affirmative and negative responses	+	+	+	+	+
Format of postal addresses	–	+	+	+	+
Information on measurement system	–	+	+	+	+
Format of writing personal names	–	–	–	+	+

Paper formats	–	+	–	+	+
Format for telephone numbers and other telephone information	–	+	–	+	+
Layout of graphical elements	–	–	+	+	+
Terminology of information technologies	–	–	–	+	–
National standards	–	–	–	+	–
Usage of special characters and separators	–	–	–	+	+
Character visualization	–	–	–	+	–
Character input, keybords, keybord layouts	–	–	–	+	–
Hyphenation, spelling, etc. (references to rules)	–	–	–	+	–
Information about the country	–	–	–	+	–
Email addresses	–	–	–	+	–
Bank account numbers	–	–	–	+	–
Text formatting rules	–	–	–	+	–
Tools to separate locale's resources	–	–	+	–	–
Total (out of 24)	6	10	10	23	14

As we can see, there is no unified way to represent locale data in a formal way. Each locale model adds additional elements to existing POSIX locale categories and extends POSIX categories. In addition to locale models mentioned here, various programming languages and environments have their own locale models. Locale models are not compatible with each other, usually only compatibility with POSIX locale is maintained, but POSIX has too limited set of locale elements. The standard ISO/IEC 15897 extends POSIX locale elements, adds many necessary locale elements. Unfortunately, not enough locale data are collected in the repository; therefore, developers cannot benefit from using it in their products. The largest locale data repository today is CLDR. We could also see that some elements, especially semantic language elements are not covered by existing locale models.

2.3. Localizable Resources and their Formats

One of the main steps during software preparation for localization is separation of the localizable resources from the source code. This means externalization of all texts, graphics, video, sounds, and locale-specific parameters, used in the program. Most of the large software producers create their own methods of separation of such resources from the program source code. Open source software development initiatives have their own adopted methods. But all the existing methods share the same shortcoming: they present localizable resources without context of their usage in the graphical interface of software, or with slight and not extensive context information. The process of development, separation and delivery of localizable resources is shown in Fig. 1.

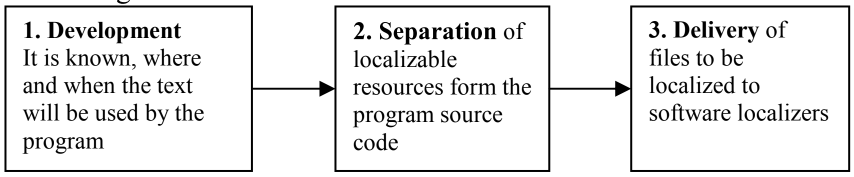

1. Development
It is known, where and when the text will be used by the program
2. Separation of localizable resources form the program source code
3. Delivery of files to be localized to software localizers

Fig. 1. Software preparation for localization: a scheme of the main stages

During the step one (Fig. 1) the context of all the texts and other resources is known. After resources separation from the source code, the context (or a major part of context) is lost. During the step three separated resources files are delivered to localizers. They do not get information on the context of the resources. A very important part of contextual information is lost between step 2 and step 3. Separated localizable resources are delivered in the default format of the separation method, or transformed into other (e.g. simple textual or intermediate) format.

The analysis of main widely used localizable resources separation methods and formats (RC, RESX, GNU Gettext, Java Resource Bundles, Mozilla, XLIFF, and PHP) has shown the following.

Despite the variety of separation methods and formats, existing today, most of them present textual localizable resources in a similar way: a simple database of key-value pairs (key is an identifier of a text string; value is a text string to be shown on the screen or to specify preferences of the program). Additional information can be added through localization comments. Exceptions are RC, RESX dialog sections formats, and XLIFF format.

RC and RESX formats are rarely used in internet software (only in Microsoft's client internet software), and they do not present contextual information on the strings section.

XLIFF format has many advantages, comparing with any other format. However, it inherits all the shortcomings of other formats, because it is used as an intermediate format, to which data is converted from other existing formats. So, the main contextual information is not present in fact. XLIFF is a universal format not only for interface strings, but also for long texts whith enough context (e.g. documentation) to translate. Therefore, it has a lot of tools to handle formatting and other information that is not so important for software interface strings.

GNU Gettext is the only format of all had been analyzed, that brings tools to handle forms of words, written after a number.

3. Investigation of Localization Process

In this section, a structure of localization process and classification of internet software cultural elements are presented. The results are based on generalized more than seven year experience in software localization, scientific literature resources and standards.

3.1. Phases of Localization Process

The *preparational phase* of localization process is aimed to evaluate the number of potential users of localized product, understand software license, find information about the author (developer) of the program and his/her other products, evaluate the program, the potential of localization, contact the author (or register on the software localizers website or database in case of open source projects), and analyze an internationalization level of the program. The preparational phase helps to choose suitable software to localize, foresee expenditures of adaptation.

Software adaptation is the second phase of localization process, which can be run in parallel with the next, dialog translation and adaptation phase. During it, all cultural elements of software are adapted to the target locale. Formal locale definitions are used to prepare software for such adaptation. The difficulty of this phase depends on the level of internationalization of the program.

Translation and adaptation of dialogs is the next phase, composed of several components, discussed below.

A primary translation of user interface strings can be treated as a *draft translation* only, because of lack of the context of each string. Unless in many localization projects, overall localization progress is measured by the percent of translated strings, this measurement does not coincide with actual localization work done. For example, in [DG06], a model is proposed, according to which localization work completeness can be measured by the percent of strings translated, e.g. translated 90% of strings mean that 42% of overall localization work is done. Therefore, this stage is related a lot with continuous *testing of translation and its correction*. Testing is done by running the program and looking for interface strings. After correction of translation, testing and correction cycle is repeated, because correction of one string can cause errors in other interface parts (due to usage of composed, parameterized, repeated and separated, but multi-used strings).

Translation and adaptation of help documents is similar to normal text translation and usually do not cause many problems, but it is closely related to interface strings testing and translations correction.

Editing of all dialog texts, including interface strings and help documents, is the next stage, done by professional language editors if previous translation was done by informatics professionals or by subject professionals if previous translation was done by language specialists.

Overall localization testing includes internal and external testing when all the previous stages of localization are completed. According to O'Sullivan, this stage includes testing of consistency of user interface elements, consistency in token functionality, aesthetics, inter-product communication, scripting considerations, error messages, cross-platform, hardware platform and other [Sul01].

3.2. Cultural Elements in Internet Software

Classification of culture has been a topic of studies by Hofstede [Hof91], Trompenaar [Tro97], and Hall [HH90]. The cultural dimensions, identified by Hofstede, are most cited and offer possibility to structure culture according to the five concepts: Power Distance; Individualism vs. Collectivism; Masculinity vs. Femininity; Uncertainty Avoidance and Long-term vs. Short-term Orientation. Trompernaar and Hall suggest respectively seven- and four-dimensional culture model. These are categories that organize general cultural data. Speaking about software, we can look at software elements that are based on culture and cultural conventions.

Basing on the analysis of the main localization-related standards, locale models and experience in software localization, we suggest classifying the most important software elements for successful software portability to different cultural environments into language-driven (those that depend on language used by the culture) and community-driven (those that depend on other cultural traditions and conventions) (Fig. 2) [DJ09].

The classification presented here, is aimed to help researchers to evaluate the level of internationalization of original software, check the user-friendliness of localized software, software developers to develop better-internationalized software, localizers to adapt more cultural elements for the target culture and detect internationalization bugs. We believe that it can help to pay more attention to various cultural elements in internet software and raise quality of its localizations.

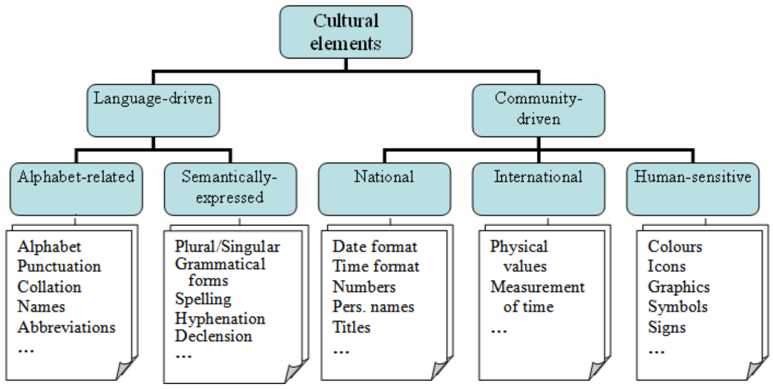


Fig. 2. A classification of internet software cultural elements

4. A Method of Formalization of Localizable Resources Metainformation

Generalizing results of the previous chapters, we can state that in general textual localizable resources are presented as a set $L = \{v, e\}$, $v \in V$, $e \in E$, V – a set of text strings names, E – a set of text strings. Text srings here are not only texts and messages, shown on the screen during program's runtime, but also values of some parameters, e.g. fonts, encodings, dialog box sizes, which adaptation can have functional effect on the program.

Localizers, working with the set of localizable strings L, face these main problems:

1. See separated words of phrases without context of use in program's graphical user interface.
2. A context of localizable text is obtained running the program and searching for particular text, or examining program's source code (if software license allows doing it).
3. A part of dialog texts can be noticed only in rare and exceptional situations (e.g. errors, interaction with other programs), that are very difficult or impossible to model in practice.
4. Fusional languages face a lot of problems choosing correct form, part of speech, etc., deciding from short English phrase without context of use. (Most of software today is localized from English.) For example, *File*, noun and verb are written in the same way, but in Lithuanian it is *failas*

(noun), *įtraukti* [į aplanką] (verb), *login* is *prisijungti* (verb), *prisijungimas*, *prisijungimo vardas* (noun). Another problem is terminology variations, slang and metaphors [Gri08].

Analysis of internet programs has shown that more than 50% of strings are strings of 1 to 4 words (this means that they will probably miss context). Localization would be accelerated if context (linguistic and user interface) was provided.

The method, presented in this section, is aimed to solve issues mentioned above.

The method is based on an attribute grammar $AG = <G, A, R>$, consisting of three components [Knu68]:

1. Context-free grammar G.
2. Finite set of attributes A.
3. Finite set of semantic rules R.

Here, $G = <N, T, P, S>$:

1. N is a finite set of nonterminal symbols;
2. T is a finite set of terminal symbols;
3. P is a finite set of grammar productions $X \to \alpha$, where $X \in N$, $\alpha \in (N \cup T)^*$. V^* is a set of all strings, made of alphabet V symbols, including empty string ε.
4. S ($S \in N$) is a starting non-terminal symbol.

Unless the main application scope of attribute grammars is semantics of programming languages, textual localizable resources can also be treated as a structured data which we would augment with semantic information. The main steps of the algorithm of the method as well as the main aspects of attribute grammar formalism modification are presented below.

Formalization of localizable resources metainformation is started by describing a context-free grammar, according to the main principles:

1. Context-free grammar G is written for a particular program, taking into account program's graphical user interface structure and relating its elements with corresponding localizable strings.
2. The grammar has two main parts, according to usage of nonterminal and terminal symbols: representation of structure of program's graphical user elements, and representation of localizable strings and their structure.
3. Usage of nonterminal symbols:
   - 3.1. A nonterminal symbol for every graphical user interface element (control), according to the specifics of the program.
   - 3.2. A nonterminal for a whole string from localizable resources.

3.3. If a resource string has a parameter(s) inside, a nonterminal is used to address each parameter. The decision to do so is based on the result of investigation of localizable resources of internet software that about 10% of localizable strings has one or more parameters inside. Usage of a parameter (variable) is a potential source of errors for most of fusional languages. The production, having on the left hand side a parameter symbol, on the right hand side has one or more nonterminals, representing resource strings, planned to be used instead of parameter, or ε, if value of parameter is obtained dynamically at program's runtime.

3.4. If menu element or other control uses text, composed of several resource strings, then in grammar's derivation tree these strings should appear alongside, introducing separate nonterminals for the control and each string.

4. Terminal symbols are resource strings or resource strings segments. If a string do not has parameters then all the string is a terminal symbol, if there are parameters, then segments between parameters are terminal symbols.

The next steps are extension of a context-free grammar to attribute grammar:

5. Grammar symbols are augmented with attributes, which are used to present information, important from localization point of view. Selection of attributes is based on the analysis of localization errors and features of the language.
6. The rules to calculate values of attributes (semantic rules) are defined.

In order to formalize metainformation of localizable resources, attribute grammars have been augmented with new tools:

- A notion of primary attribute has been introduced. A set of attributes of a grammar can be expressed as $A(X) = S(A) \cup I(A) \cup E(A)$, here $S(A)$ is a set of synthesized attributes, $I(A)$ is a set of inherited attributes, and $E(A)$ is a set of primary attributes. Primary attributes are assigned not only to terminal symbols, but to non-terminal symbols as well.
- The values of primary attributes are provided externally, e.g. using attribute questionnaire.
- The scope of attribute passing through the attributed derivation tree has been conrolled, limiting it to the surrounding of the particular node.

Fig. 3 shows a fragment of grammar's derivation tree for a hypothetical internet software dialog box. Marked area corresponds to graphical user interface part of a grammar. Another part is a strings part. Grammar symbols, written in bold, correspond to terminal symbols (localizable strings). One of such strings has a parameter.

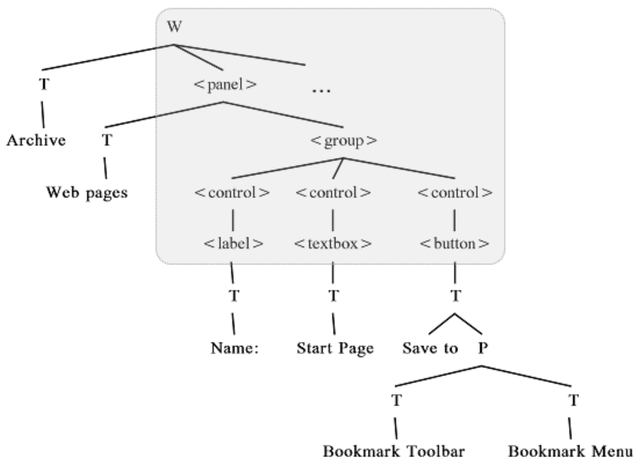


Fig. 3. Localizable resources grammar's structure: GUI and strings parts

A list of some attributes, used to extend context-free grammar, is presented below. All the attributes are expressed independently of the language, by means, available in English language, so that the author of software is able to assign the values. So, the most attributes and values of the attributes are common for fusional alphabetic languages.

Noun/verb phrase. This attribute is used to avoid an error arising from the same spelling of English nouns and verbs. The attribute is especially important for short strings. For example, the same phrase *Open file* in menu context would have verb form, but in a dialog box title would have noun form in Lithuanian. When assigned automatically, the attribute is recomendational.

Controlling word of a phrase. This attribute marks the word that controls forms of other words of a phrase. In fusional languages it can help to select correct case of surrounding words.

String description. The attribute is provided for strings, having special characters, abbreviations, unclear semantics or rare error messages.

First letter (capital or small). In English, capital letters are usually used in the middle of the phrases and sentences, e.g. months, days names, titles. In other languages, e.g. in Lithuanian, usually only proper nouns, abbreviations in the middle of the phrase are written from the capital letter. In other cases a small letter is used.

Form. A word form, expressed by the question it answers to, e.g. *who*, *what*, *whom*, *where*, *when* or preposition: *to*, *from*, *in*. *Who* and *what* questions are separated into two groups: *subject* and *object*.

Term. The attribute is designed to specify meaning of homonymous terms, marking the number of meaning according to agreed dictionary. E.g. term *key* has at least four meanings in Lithuanian: (1) *klavišas*, (2) *raktas*, (3) *kodas*, (4) *šifras*, etc. The dictionary can be a special dictionary, designed for a particular program's localization needs, or a common dictionary of computer terms.

Internal function. A link to internal function, implementing the particular command. In some cases is needed to understand better the meaning of the command.

String identifier. An identifier of a string in a resource system. Is used as an auxiliary attribute to keep a set of strings to be substituted in the place of parameter, composed strings or alternative strings.

Width or height. An attribute, used to define a width or a height of a dialog box or its control. Used to calculate if there is enough place to display the translated string.

Indicator of composed string. Several strings, used to compose a single user interface string, are included in the grammar tree alongside. As composable strings and alternative strings have the same formal description, the attribute is used to indicate if the strings are composed. The attribute is also used to calculate the length of the composed string.

Data type. If a string has a parameter, it is useful to know what type of data will be written instead of it during program runtime. The attribute will help to choose forms of parameter's surrounding words, e.g. if a number of object is inserted instead of parameter, then it is necessary to adjust forms of a noun, written after parameter; if the parameter is a person's name, then it is necessary to foresee change of the forms and gender of surrounding words. This is also an indicator for software developers to include a component to adjust such change of forms.

Tracking of access keys and command keys. This group of attributes is designed to keep control of access and command keys, changed during localization: check of duplicated keys in the scope.

The next step is definition of semantic rules (functions, depending on the values of other attributes or values, submitted externally). A part of attributes is calculated; another part of attributes is submitted externally, e.g. via questionnaire. Semantic information on localizable resources is spread over all attributed derivation tree, but not collected in the attributes of a root symbol of a tree.

Two generalized cases of attribute grammars are presented: menu and typical dialog box.

5. EXPERIMENTAL EVALUATION OF LOCALIZABLE RESOURCES METAINFORMATION FORMALIZATION METHOD

The method, proposed in the previous section, is aimed to ease localization process and raise quality of localizations. Due to attributes, primary translation of localizable resources is expected to be of higher quality, therefore testing and translation correction component of localization process will need less expenditure of efforts. However, additional expenditures are included due to preparation of attribute grammar description (formalization of metainformation). Localizable resources metainformation is formalized once per program, therefore if program is localized to more than one language, overall expenditures of efforts decrease.

In aim to evaluate the ratio between the main expenditure of works involved, and find minimal number of languages of localizations, an expert opinion poll was conducted. Three experts were selected, according to formulated competence requirements (not less than 7 year experience in the field of software localization and (or) internationalization; practically localized not less than 5 programs; published at least 3 scientific papers in the field of software localization/internationalization).

A questionnaire, consisting of eight questions, was prepared to address the main aspects of localization with formalized metainformation, using the method, and localization without it. As a unit of measurement, expenditures of work, needed to conduct primary translation of textual localizable resources without contextual information, were selected.

Table 2 presents the main aspects of each question and corresponding coefficient, received from the expert's answer to that question. Here, W1–W8 mark localization works (the components of localization process using the method and localization process without using it).

In case of localization using the method, two cases are evaluated: 1) an author of software prepares contextual information of localizable resources; 2) a localizer or an experienced user of software prepares contextual information of localizable resources. Descriptions of works, related to localization (W1–

W8), are explained and corresponding formula to evaluate expenditures of that work, according to experts' answers, are presented in Table 3.

Table 2. Questions and the corresponding aspects being analyzed

Question	**Aspect**	**Coefficient**
Question 1	Ratio of expenditures of W2 and W1	a_1
Question 2	Ratio of expenditures of W3 and W1	b_1
Question 3	Ratio of expenditures of W4 and W2	b_2
Question 4	Ratio of expenditures of W5 and W1	b_3
Question 5	Ratio of expenditures of W6 and W3	b_4
Question 6	Ratio of expenditures of W7 and W3	c_1
Question 7	Ratio of expenditures of W8 and W7	c_2
Question 8	General comments in free form	–

Table 3. Description of works and their evaluation formulas

Work No.	**Description of work**	**Formula**
W1	Translation of textual resources (usual localization practice, without metainformation)	100
W2	Testing and correction of translation (usual localization practice, without metainformation)	$a_1 \times 100$
W3	Metainformation formalization (by author)	$b_1 \times 100$
W4	Testing and correction of translation (with metainformation)	$a_1 b_2 \times 100$
W5	Translation of textual resources (with metainformation)	$b_3 \times 100$
W6	Metainformation formalization for another fusional alphabetic language (by author)	$b_1 b_4 \times 100$
W7	Metainformation formalization (by localizer or experienced user)	$b_1 c_1 \times 100$
W8	Metainformation formalization for another fusional alphabetic language (by localizer or experienced user)	$b_1 b_4 c_2 \times 100$

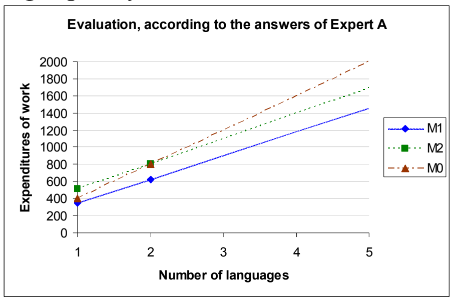
According to the coefficients, derived from the answers of experts to the Questions 1–7 (Table 4), we evaluate expenditures of work, needed to localize software for one and two fusional alphabetic languages. Answers to Question 8 help to define future works and general opinion about the method.

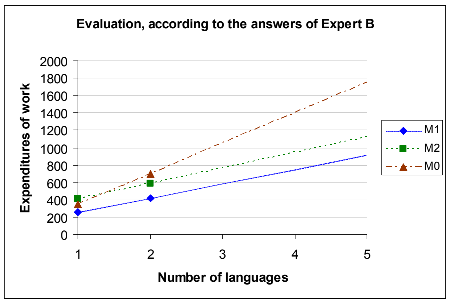
Table 4. Coefficients, derived from the answers of experts

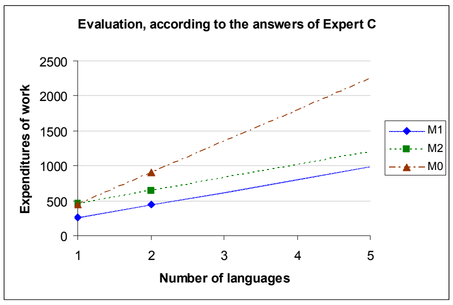
Expert A	**Expert B**	**Expert C**
$a_1 = 3$	$a_1 = 2,5$	$a_1 = 3,5$
$b_1 = 0,8$	$b_1 = 1$	$b_1 = 1$
$b_2 = 0,4$	$b_2 = 0,3$	$b_2 = 0,3$

$b_3 = 1,5$	$b_3 = 0,8$	$b_3 = 0,5$
$b_4 = 0,1$	$b_4 = 0,1$	$b_4 = 0,3$
$c_1 = 3$	$c_1 = 2,5$	$c_1 = 3$
$c_2 = 3$	$c_2 = 2,5$	$c_2 = 1$

Applying the method, localizable resources are augmented with contextual information, helping to find a correct translation at once; therefore work of context search is considerably reduced. As a result, in case when software is localized to different languages by different localizers (a usual localization practice), the expenditures of localization work of additional fusional alphabetic language are almost the same. The simplified dependencies of number of languages to be localized for and expenditures of general localization works are presented graphically in Fig. 4. In all cases only high quality localization work is considered.







M0 – localization without the method

M1 – localization using the method, when the context is prepared by the author of software

M2 – localization using the method, when the context is prepared by the localizer or experienced user of software

Fig. 4. Expenditures of localization works, depending on the number of languages and various types of localization (M0–M2), according to the evaluations, derived from the answers of experts

According to evaluations, derived from the answers of Expert A, the method is worth applying, if software is localized for 2,03 languages (x_A = 2,03). According to Expert B answers, the method is worth applying, if software is localized for 1,32 languages (x_B = 1,32). According to Expert C evaluations, the method is worth applying, if software is localized for 1,02 languages ($x_C = 1{,}02$).

Aggregating the evaluation results, we apply not the mean value calculation, but ceiling rounding of the maximum (the most pessimistic) result, derived from experts' evaluations: $\lceil \max(x_A, x_B, x_C) = 3$, here $\lceil$ marks an operation of ceiling rounding to the whole number.

According to evaluation, derived from the answers of experts, the method, proposed in this work, is worth applying if software is going to be localized for not less than 3 languages. In this case, expenditures of localization works are considerably lower, comparing to expenditures of localization works when the method is not used.

Other conclusions, derived from the answers of experts: 1) expenditures of testing and translation correction work decreases 3 times, if localizable resources are presented with metainformation on their context; 2) preparation of metainformation, according to the method, would require expenditures of work, equal to the same of translation of resources; 3) if software is going to be localized for one language only, then the method is worth applying only if metainformation is prepared by the author.

General Conclusions and Results

1. The method of formalization of metainformation of localizable resources has been developed and presented, basing on collected and systematized knowledge through the analysis of software localization, investigation of software localization process, practical internet software localization. The proposed method:
    - 1.1. Extends attribute grammars with three main novelties: a) attribute grammars are augmented with a new attribute type; b) computed attributes are complemented with attributes, entered from outside; c) attribute context scope is controlled.
    - 1.2. Presents localizable resources in a systematized way and helps to notice internationalization errors during early, software development, stage.
    - 1.3. Includes into localizable resources information, describing their context and useful to raise quality of localization: place where resource string will appear on the screen, relations between related strings, and semantic information of each text string.

2. Improved attribute grammars method is important from the theoretical informatics point of view, because it presents a new attitude towards attribute grammars, and practical point of view, because it can be used not only in internet software localization, but to model other linguistic scopes as well.
3. The analysis and comparison of existing locale models, localizable resources separation methods and representation formats, the decisions, tended to contribute towards quality of localized software have been foreseen:
   3.1. Extending existing formats of textual localizable resources: a) adding contextual information of resource strings, b) marking the place, where static, as well as dynamic strings appear on the screen at program's runtime, c) management of parameterized, composed strings, d) inclusion of language semantic information into localizable resources.
   3.2. Considering additional attention to cultural elements, which are not included in the formal locale definitions: forms of user interface texts, usage of capital and small letters, phrase composition, homonymous computer terms, choosing of parts of speech, etc.
   3.3. Decision of the problems, related to formats of textual localizable resources and cultural elements, should be moved to the early stage of software development instead of fixing the problems at localization time, or even after product's release.
4. As a result of investigation of software localization process and the main cultural elements of internet software:
   4.1. Classification of internet software cultural elements is prepared, that is aimed to help a) software developers to develop better-internationalized software, b) localizers to adapt more cultural elements for the target culture and detect internationalization bugs, c) researchers to evaluate the level of internationalization of original software, check the user-friendliness of localized software.
   4.2. Determined that the way of software localization, when the translation of localizable resources makes just a small part of overall localization due to not sufficient information on context of such resources, is dominant. The errors, made during primary translation are corrected during software testing cycle (which requires 3 times more work than translation itself).
5. Evaluation of the proposed method has shown that:

5.1. Applying the method to formalize contextual information of the localizable resources, expenditures of testing and correction of localized resources decrease 3 times.
5.2. The method of formalization of localizable resources metainformation is worth applying if software is planned to be localized to 3 or more fusional alphabetic languages.

LIST OF LITERATURE, REFERENCED IN THIS SUMMARY

[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.

[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: 3–12, 2006.

[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.

[Ess00] Esselink, B. A practical guide to localization. John Benjamins, 2000.

[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. In: Lituanistika pasaulyje šiandien: darbai ir problemos. Baltos lankos, t. 3, p. 65–72, Vilnius, 1998.

[Hal90] Hall E., Hall M. Understanding Culture Differences. Maine, Intercultural Press, 1990.

[Hof91] Hofstede, G. Cultures and Organization. McGraw Hill, New York, 1991.

[ISO02] ISO 639-1:2002. Codes for the representation of names of languages – Part 1: Alpha-2 code.

[ISO03] ISO/IEC 9945-2:2003. Information technology – Portable Operating System Interface (POSIX). Part 2: System Interfaces.

[ISO04] ISO/IEC 14652:2004. Information technology – Specification method for cultural conventions.

[ISO97] ISO 3166-1:1997. Codes for the representation of names of countries and their subdivisions – Part 1: Country codes.

[ISO99] ISO/IEC 15897:1999. Information technology – Procedures for registration of cultural elements, 1999.

[Knu68] Knuth, D. E. Semantics of context-free languages. Theory of Computing Systems, vol. 2, no. 2, 1968, p. 127–145.

[Sul01] O‘Sullivan, P. A Paradigm for Creating Multilingual Interfaces. Doctoral Dissertation. University of Limerick, 2001.

[Tro97] Trompenaars, F. Riding the waves of culture – Understanding cultural diversity in business. Nicholas Brealey Publishing, London, 1997.

[Uni07] Unicode, Inc. Unicode CLDR Project: Common Locale Data Repository, 2007. http://unicode.org/cldr/

LIST OF PUBLICATIONS BY THE AUTHOR ON THE SUBJECT OF DISSERTATION

Articles in peer-reviewed periodical journals:

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. A Method of Formalization of Localizable Resources Metainformation. Information sciences. 50 (2009), p. 205–211. ISSN 1392-0561. (In Lithuanian)
3. Dagienė, V., Grigas, G., Jevsikova, T. Experience of Software Localization into Lithuanian. Information sciences. 31 (2004), p. 171–184. ISSN 1392-0561. (In Lithuanian)
4. Dagienė, V., Grigas, G., Jevsikova, T. Open Source Software Policy in Education. Information sciences. 34 (2005), p. 25–29. ISSN 1392-0561. (In Lithuanian)
5. Dagienė, V., Jevsikova, T. Virtual Learning Environments from the Localization Point of View. Lithuanian Mathematical Journal. 45 (spec. No). (2005), p. 197–202. ISSN 0132-2818. (In Lithuanian)
6. Grigas, G., Jevsikova, T. Internet Application Suite Mozilla Localization and Usage in Education. Lithuanian Mathematical Journal. 42 (spec. no). Vilnius, 2002, p. 241–248. ISSN 0132-2818. (In Lithuanian)

Articles, included in the list of Institute of Science Information (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]

Articles, published in other proceedings of scientific conferences:

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' [Electronic Resource]: 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. Software Adaptation to Lithuanian Locale. Information Technologies '2003, Conference Proceedings, Kaunas, 2003, p. I-(8–14). ISBN 9955-09-335-8. (In Lithuanian)

SHORT DESCRIPTION ABOUT THE AUTHOR

Tatjana Jevsikova received her B.Sc. degree in mathematics and informatics from Vilnius Pedagogical University, Faculty of Mathematics and Informatics, in 2000. In 2002 she received the M.Sc. degree in informatics from Vilnius Pedagogical Universtity, Faculty of Mathematics and Informatics. During the period of 2005–2009 she was a PhD student at the Institute of Mathematics and Informatics (physical sciences, informatics). Since 2001 Tatjana Jevsikova has been working as an engineer and as a junior researcher for the Institute of Mathematics and Informatics. Since 2008 she has been teaching at the faculty of Mathematics and Informatics, Vilnius University.

Email: tatjanaj@ktl.mii.lt

SANTRAUKA

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 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.

Darbo objektas

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

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.

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.

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.

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.

Praktinė darbo reikšmė

Pagrindinė pasiūlyto metodo praktinė vertė 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.

Aprobavimas ir publikavimas

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 leidiniuose. Disertacijos rezultatai pateikti 14 pranešimų 12 tarptautinių ir nacionalinių konferencijų bei seminarų.

Darbo struktūra

Darbą sudaro: terminų 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 darbo praktinė reikšmė, pateikiamas autorės mokslinių publikacijų disertacijos tema ir mokslinėse konferencijose skaitytų pranešimų disertacijos tema sąrašas.

Antrojoje dalyje „Programinės įrangos lokalizavimo analizė“ 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 būdai bei apžvelgiami lokalizavimo dėsniai.

Trečiojoje dalyje „Lokalizavimo proceso tyrimas“ 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 „Lokalizuojamųjų išteklių metainformacija ir jos formalizavimo metodas“ nagrinėjama tekstinių lokalizuojamųjų išteklių struktūra bei pasiūlytas 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 „Eksperimentinis lokalizuojamųjų išteklių metainformacijos formalizavimo metodo vertinimas“ pateiktas trečiojoje dalyje pasiūlyto metodo taikymo pavyzdys, atliktas metodo ekspertinis vertinimas ir apibendrinti jo rezultatai.

Bendra disertacijos apimtis yra 149 puslapiai B5 formatu.

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.

Trumpos žinios apie autorę

Tatjana Jevsikova matematikos ir informatikos bakalauro laipsnį įgijo 2000 m. Vilniaus pedagoginiame universitete, Matematikos ir informatikos fakultete. 2002 m. įgijo informatikos magistro laipsnį Vilniaus pedagoginiame universitete, Matematikos ir informatikos fakultete. 2005–2009 m. studijavo Matematikos ir informatikos instituto ir Vytauto Didžiojo universiteto doktorantūroje (fiziniai mokslai, informatika). Nuo 2001 m. dirba Matematikos ir informatikos institute inžiniere, jaunesniąja mokslo darbuotoja. Nuo 2008 m. dėsto Vilniaus universitete, Matematikos ir informatikos fakultete.

El. pašto adresas: tatjanaj@ktl.mii.lt

Tatjana Jevsikova

INTERNET SOFTWARE LOCALIZATION

Summary of Doctoral Dissertation
Physical Sciences (P 000)
Informatics (09 P)
Informatics, Systems Theory (P 175)

Išleido ir spausdino AB MINTIS,
Z. Sierakausko g. 15, LT-03105 Vilnius
Interneto svetainė http://www.mintis.eu/
Tiražas 60 egz.