UDC 004.51

V. Matuzko
Zaporizhzhia National University, 66 Zhukovskoho Str., Zaporizhzhia, Ukraine, 69600; e-mail: matuzkovd@ukr.net

SOFTWARE IMPLEMENTATION OF AUTOMATED USER INTERFACE TRANSLATION TOOL

В. Матузко. **Програмна реалізація утиліти для автоматизованого перекладу інтерфейсу програм.** Велика кількість повсякденних дій вже давно виконується за допомогою мобільних додатків та ресурсів у мережі Інтернет. Більшість з них мають інтерфейс на обмеженій кількості мов світу. Через це у користувачів постає проблема знання мови для можливості користування цими ресурсами. Не всі розробники в світі мають доступ до професійних послуг перекладу або можливість створити належний переклад власноруч. Зважившись ці фактори можна зазначити, що існує потреба в зручній та доступній програмі, що спеціалізована для створення та забезпечення якості перекладу саме інтерфейсів програмного забезпечення та веб-сторінок. В даній роботі проведено аналіз переваг та недоліків можливих методів реалізації та існуючі алгоритми, такі як GNU gettext. На базі цього аналізу визначені вимоги до запропонованої програми, а саме зручний інтерфейс та набір функцій, що спеціалізовані для роботи з програмними інтерфейсами та файлами вихідного коду. Перелік головного функціоналу включає в себе взаємодію з інтернет-сервісами машинного перекладу та можливість переглядати вихідний код програми або веб-сайту в контексті текстових елементів інтерфейсу користувача. Розроблено та описано комп'ютерну утиліту у версії для ОС Microsoft Windows, яка реалізує всі вимоги до базового функціоналу. Проектна версія утиліти реалізована за допомогою мови програмування C# та середовища розробки Microsoft Visual Studio. Програма спеціалізована для перекладу інтерфейсів користувача та дозволяє автоматизувати цей процес за допомогою використання сервісу Google Translate. Для максимальної зручності використання утиліта зберігає результати роботи в простому текстовому форматі, призначений для зчитання під час роботи цільовою програмою або веб-сайтом. Також наведена детальна демонстрація робочого процесу на прикладі перекладу інтерфейсу самої утиліти, та визначено напрямки потенційного розвитку в наступних версіях.

Ключові слова: машинний переклад, розробка програмного забезпечення, інтерфейс користувача, Google Translate

V. Matuzko. **Software implementation of automated user interface translation tool**. Numerous daily activities are long since accomplished using mobile applications and international resources available through the Internet. The majority of such resources support only a limited amount of world languages for the interface. This raises the issue of the need for end users to know the languages required to operate and use these programs. Not every developer in the world has access to professional translation services, or the ability to create such translations on their own. Evaluating these factors shows an existing need of a convenient and accessible program specialized in creating and ensuring quality translation of software user interfaces specifically. Analysis was conducted to determine advantages and disadvantages of possible implementation methods and existing algorithms, i.e. GNU gettext. Based on this analysis requirements for the resulting program are formed, those being a comfortable user interface and a set of functions specific to working with user interfaces and source code files. The list of primary functionality includes interaction with internet services for machine translation and the ability to view the source code of a program or website in context of text elements of the user interface. A program tool for the Microsoft Windows environment that implements all requirements to base functionality is conceptualized and developed. The project version of the program was implemented using C# programming language and Microsoft Visual Studio development environment. The program is specialized for the translation of user interfaces, and enables automation of this process via usage of the Google Translate service. To maximize ease of use the program stores the resulting translation in a simple text format, designed to be loaded and read from during runtime of the target program or website. Also included is a detailed demonstration on the workflow using the interface of the translation program as an example, and potential directions of further developments are determined.

Keywords: machine translation; software development; user interface; Google Translate

Introduction

Throughout decades computerization in the modern world allows to perform a large number of daily and civic activities digitally. A large percentage of institutions and businesses have websites and apps available for personal devices. The biggest of these companies are able to conduct business on the international market, which poses the task of enabling cooperation with clients and users that speak various languages of the world.

However, a large number of these apps only have a version available with just one language, or uses machine translation without error checking the text or context within the user interface. Also, the quality of the resulting machine translation depends on the language pair. Usually this occurs due to a lack of resources or comfortable universal instruments to create user interface translation with. This is especially the case with small teams and individual software developers.

Analysis of literary data and problem definition

There exist numerous software solutions made to ease the work of translators, each with a unique approach to solving this problem. Several of them are premium-level paid software packages for professional translation businesses [1]. However, a closer look can determine a lack of programs specifi-

DOI: 10.15276/opu.1.69.2024.12

ISSN 2076-2429 (print)
ISSN 2223-3814 (online)

cally aimed at translation of computer user interfaces. This kind of translation differs from translating text articles and books. Many words and phrases are used several times in different places in applications, and their meaning can drastically change depending on the context and program functionality. Usually, as well, translation tools have an interface mostly designed for translating relatively large blocks of text, and may lack efficiency when translating numerous short strings and separate words.

Evaluating these facts, a list of required functionalities for a specialized tool, designed for use by independent or small developer teams to translate their programs can be formed. One of the main features must be the ability to view the original source code of a program to fully understand the context during the translation of the user interface. Another important feature is providing the ability to use machine translation to improve the speed and efficiency of translation, with the ability to do quality control afterwards. Such automation can be implemented using existing online services for machine translation through an application interface. An example system is Google Translate API, which allows developers to incorporate access to the translation service into their applications [2].

To use machine translation the developer may use their personal API access key. This allows to automatically localize some or all strings in a chosen translation file.

To be able to potentially use the translation program the developer must account for translation by preparing the user interface and source code of their program. This also enables the functionality of the source code viewing window when using the translation tool.

One of the possible ways to implement this functionality is using the PO format, which is used in the GNU gettext localization system [3]. When using this approach the developer generates a PO file, which contains a list of strings that corresponds to the text interface of their program. The format of PO file entries is shown in Fig. 1.

The resulting PO file can then be opened in a translation tool. But the main disadvantage of this approach is that the automation via gettext is only implemented for a limited selection of popular programming languages, and thus it is inadvisable to use it to translate webpage interfaces [4].

Unlike the previous method XLIFF is not limited to any programming language, which allows its usage in many more situations than gettext [5]. But this requires developers to create or use an existing compatibility library to extract information from XLIFF files. Much like with gettext the user interface must be prepared for localization, but unlike gettext the exact implementation is up to the developer. The XLIFF format is also noticeably more cumbersome as a result (Fig. 2).

```
white-space
#  translator-comments
#. extracted-comments
#: reference…
#, flag…
#| msgid previous-untranslated-string
msgid untranslated-string
msgstr translated-string
```

Fig. 1. Format of entries in a PO file (Source: https://www.gnu.org/software/gettext/manual/html_node/PO-Files.html)

```
<xliff xmlns="urn:oasis:names:tc:xliff:document:2.0" version="2.0"
 srcLang="en-US" trgLang="ja-JP">
 <file id="f1" original="Graphic Example.psd">
  <skeleton href="Graphic Example.psd.skl"/>
  <unit id="1">
   <segment>
    <source>Quetzal</source>
    <target>Quetzal</target>
   </segment>
  </unit>
  <unit id="2">
   <segment>
    <source>An application to manipulate and process XLIFF documents</source>
    <target>XLIFF 文書を編集、または処理 するアプリケーションです。</target>
   </segment>
  </unit>
 </file>
</xliff>
```

Fig. 2. Format of entries in an XLIFF file (Source: https://en.wikipedia.org/wiki/XLIFF)

Each of the described formats has its advantages, but with the aim to provide developers a neat and functioning solution a better approach is to generate and work with localization files in a simple universal format. The translation tool must provide complete functionality to work with such files. After their creation these files are loaded with the target program or webpage during runtime and provide text for all user interface elements.

The goal of this study is to develop a software implementation of the user interface translation program that allows automation of this process.

Software implementation

To implement the user interface translation program the programming language C# and development environment Microsoft Visual Studio were chosen due to its powerful toolset for creating windowed applications for Microsoft Windows. The main functionality of the developed program lies

in demonstrating the translation algorithm, thus the quickest available method to design and create a working interface for the translation tool was chosen (Fig. 3).

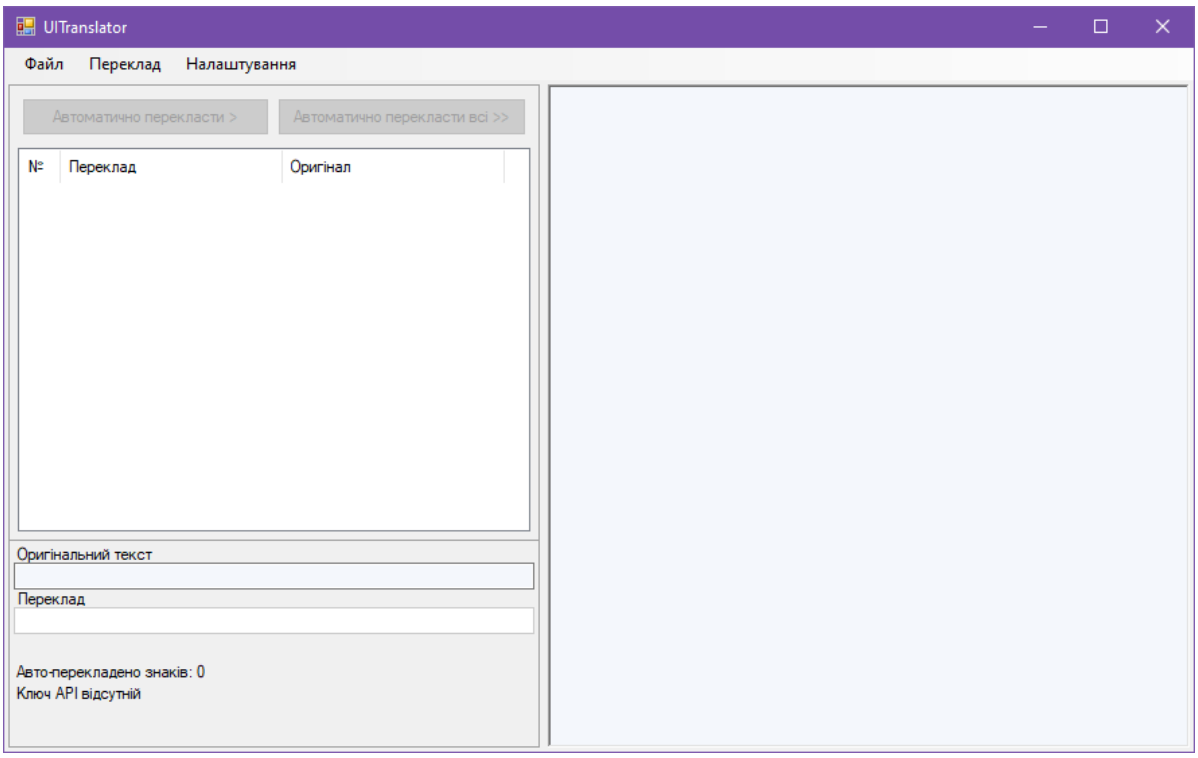


Fig. 3. The main window of the translation program (Source: compiled by the author)

To start with the developer must first create a new project using existing source code using the “File” menu. In the case of a properly prepared source code file the list will contain all detected strings that correspond with the text interface of the program and numbering used in the code (Fig. 4). For convenience the source code lines will be copied over into the “Source” column, and to prevent accidental misuse the automated translation feature is unavailable when creating new translation projects.

Text for a chosen string is inputted in the “Translation” field (Fig. 5). After confirming the change, it can be instantly seen in the list of strings, and the “Translation” field temporarily changes color.

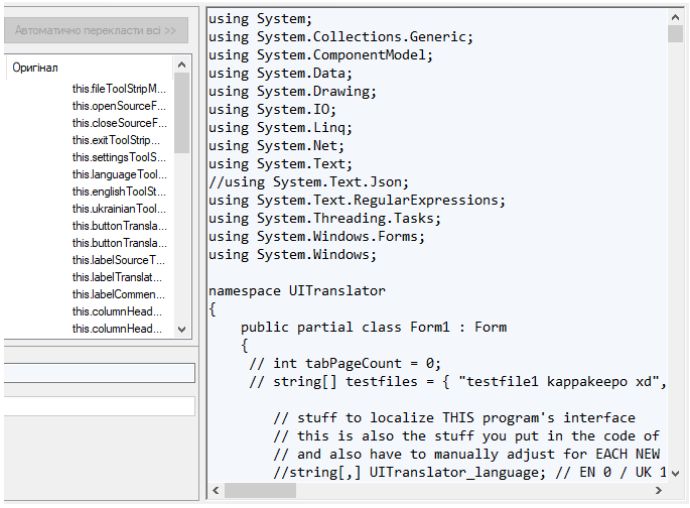


Fig. 4. New project from source code (Source: compiled by the author)

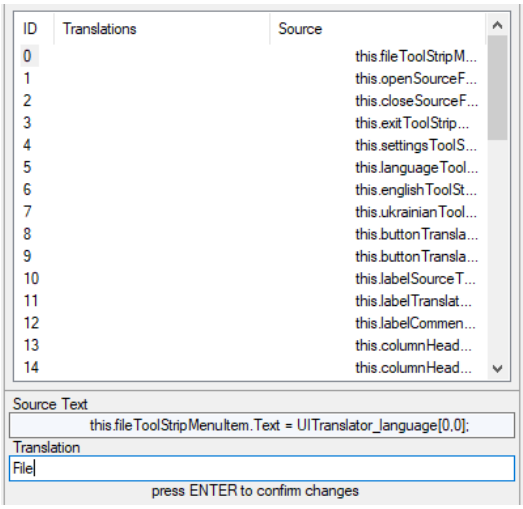


Fig. 5. Manual input of interface text (Source: compiled by the author)

ISSN 2076-2429 (print)
ISSN 2223-3814 (online)

To allow for the functionality of automated translation an existing localization file must be used as a source for strings for the “Source” column (Fig. 6). For this the “New project from existing language file...” in the “File” menu must be used. Afterwards the program source code file can be loaded according to the user’s needs.

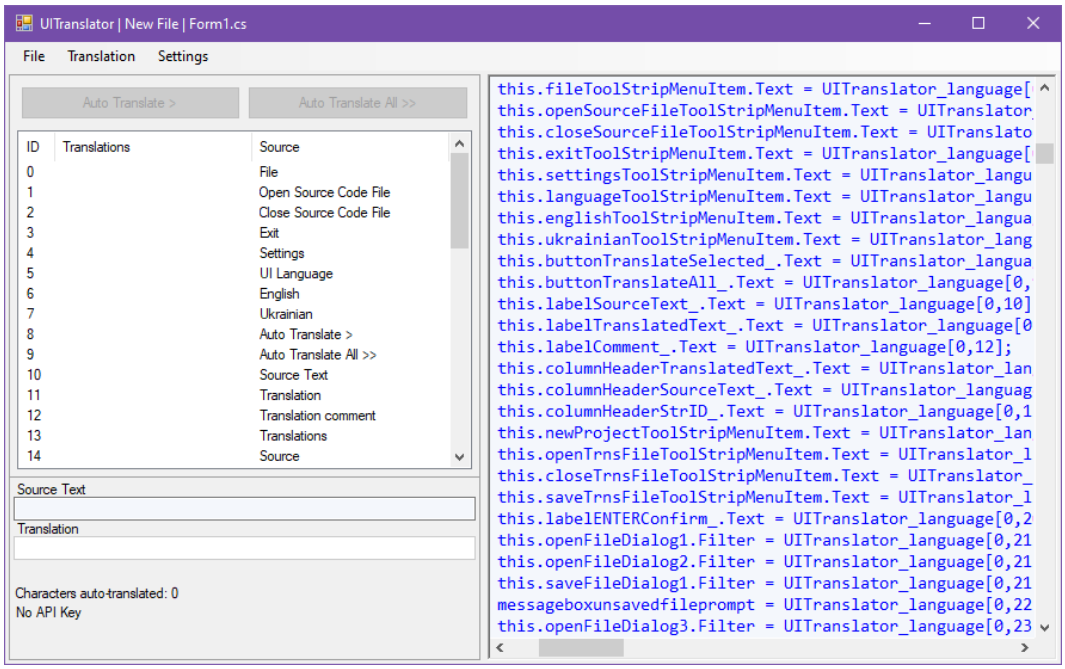


Fig. 6. New project based on an existing user interface text file (Source: compiled by the authors)

The presence of existing text allows the translator to automate the translation. For this the “Translation” menu must be used to set the language pair and to enter the personal access key to the online translation API. At the moment the tool supports access to the Google Translate API (Fig. 7). The list of available languages is represented with their two-letter code and corresponds to a big portion of the most used languages that are available in Google Translate [6]. The selection of the source language in the pair is done manually to avoid errors caused by incorrect server-side automatic language detection.

The automated translation process resembles the manual one – selecting the required row and pressing the “Auto Translate >” button. After pressing the button, the tool will ask to confirm the operation (Fig. 8). This is due to the fact that online services set limits on programmatically translated symbols for each user. Normally a certain amount is available for free – at the time of writing Google Translate allows developers to translate 500 thousand symbols every month without requiring payment. Going over this limit will cost a developer a certain amount of money [7]. This warning can be partially turned off in the “Translation” menu. Also the number of symbols translated during the session is displayed in the program interface to allow better tracking of resource usage.

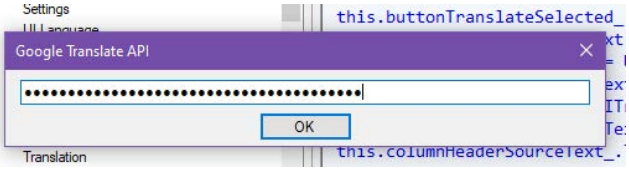


Fig. 7. Entering the key to use the Google Translate API (Source: compiled by the author)

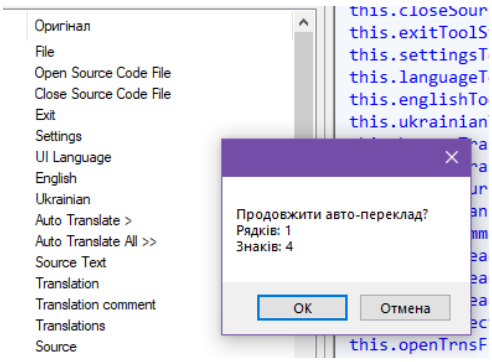


Fig. 8. Confirmation of automated translation of a text string (Source: compiled by the author)

If the operation is successful the “Translation” column will contain the machine translation results, received after a web request. If this web request encounters any sort of error, then a message with the HTTP error code will be displayed and the operation will be cancelled without using up the allotted symbols. Server access denial is displayed in the same manner, and is usually caused by using an invalid or inactive API key.

Using automated translation allows translation of several rows of text in one operation.

This is done by selecting the required rows and pressing the “Auto Translate >” button. Translation of 10 or more rows always requires confirmation of the translation operation to avoid accidental requests, even if the warning is turned off in the menu (Fig. 9).

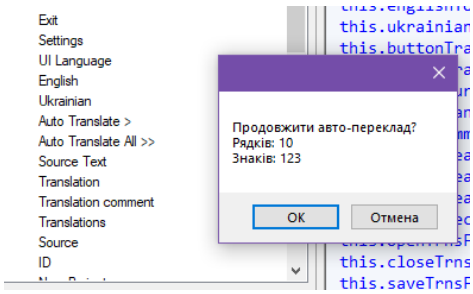
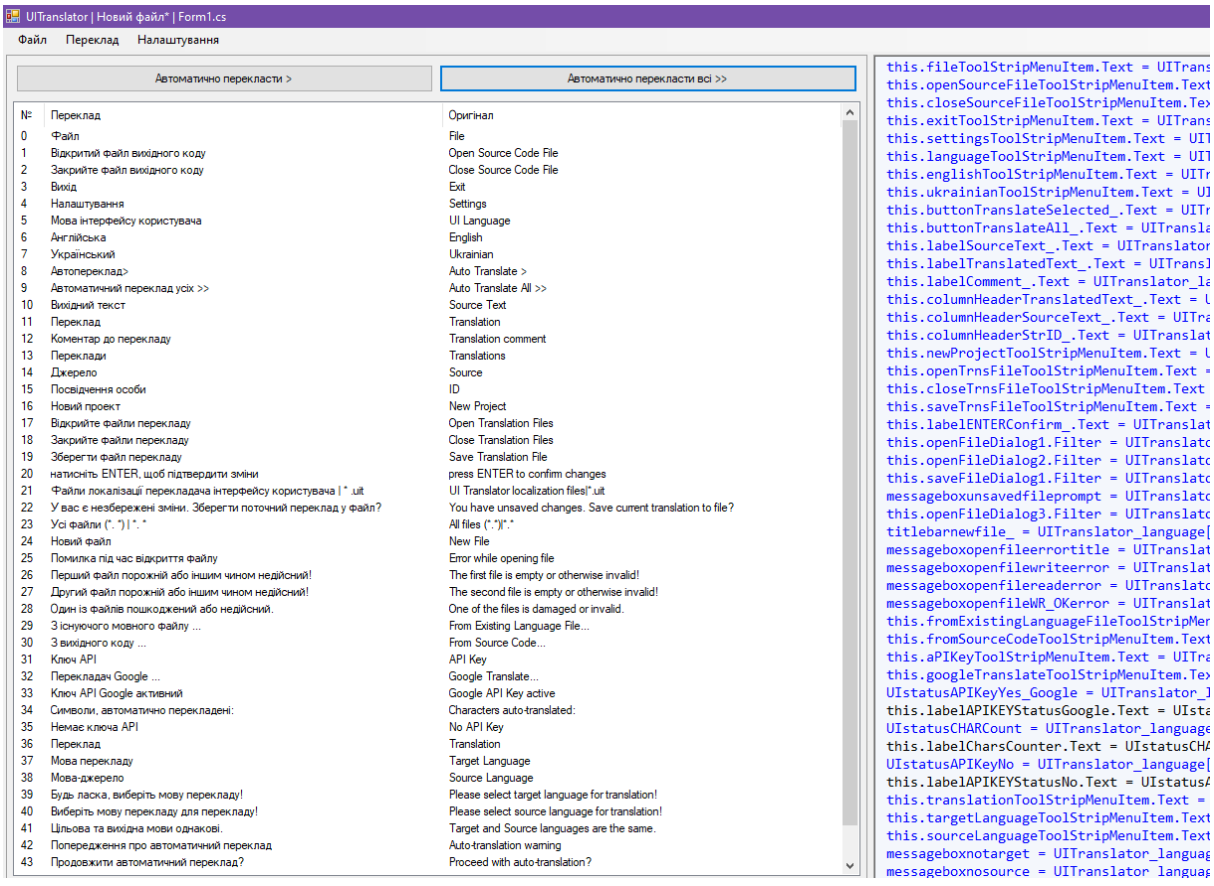


Fig. 9. Translation confirmation window, which also displays the number of selected rows and symbols that will be sent in the web request (Source: compiled by the author)

In case of a successful machine translation operation the result obtained from the online service will irreversibly remove any information contained in the “Translations” column for the selected rows. This is also important to keep in mind when using the “Auto Translate All >>” function, which will translate absolutely all rows of text in the “Source” column. Result of such translation is shown in Fig. 10.



№	Переклад	Оригінал
0	Файл	File
1	Відкритий файл вихідного коду	Open Source Code File
2	Закрийте файл вихідного коду	Close Source Code File
3	Вихід	Exit
4	Налаштування	Settings
5	Мова інтерфейсу користувача	UI Language
6	Англійська	English
7	Український	Ukrainian
8	Автопереклад>	Auto Translate >
9	Автоматичний переклад усіх >>	Auto Translate All >>
10	Вихідний текст	Source Text
11	Переклад	Translation
12	Коментар до перекладу	Translation comment
13	Переклади	Translations
14	Джерело	Source
15	Посвідчення особи	ID
16	Новий проект	New Project
17	Відкрийте файли перекладу	Open Translation Files
18	Закрийте файли перекладу	Close Translation Files
19	Зберегти файл перекладу	Save Translation File
20	натисніть ENTER, щоб підтвердити зміни	press ENTER to confirm changes
21	Файли локалізації перекладача інтерфейсу користувача | *.uit	UI Translator localization files|*.uit
22	У вас є незбережені зміни. Зберегти поточний переклад у файл?	You have unsaved changes. Save current translation to file?
23	Усі файли (*. *) | *. *	All files (*.*)|*.*
24	Новий файл	New File
25	Помилка під час відкриття файлу	Error while opening file
26	Перший файл порожній або іншим чином недійсний!	The first file is empty or otherwise invalid!
27	Другий файл порожній або іншим чином недійсний!	The second file is empty or otherwise invalid!
28	Один із файлів пошкоджений або недійсний.	One of the files is damaged or invalid.
29	З існуючого мовного файлу ...	From Existing Language File...
30	З вихідного коду ...	From Source Code...
31	Ключ API	API Key
32	Перекладач Google ...	Google Translate...
33	Ключ API Google активний	Google API Key active
34	Символи, автоматично перекладені:	Characters auto-translated:
35	Немає ключа API	No API Key
36	Переклад	Translation
37	Мова перекладу	Target Language
38	Мова-джерело	Source Language
39	Будь ласка, виберіть мову перекладу!	Please select target language for translation!
40	Виберіть мову перекладу для перекладу!	Please select source language for translation!
41	Цільова та вихідна мови однакові.	Target and Source languages are the same.
42	Попередження про автоматичний переклад	Auto-translation warning
43	Продовжити автоматичний переклад?	Proceed with auto-translation?

Fig. 10. Expanded window with the results of complete machine translation (Source: compiled by the author)

The language pair can be set and freely changed at any time when working with localization files. Fig. 11 shows examples of English strings translated to Ukrainian, German and French.

Description of text storage formats

Localization files that are created by the tool are stored with the special file extension .uit, that filters the file open and save windows in the menu (Fig. 12). In the event of errors or damage to the file the operation will be canceled with the appropriate error message shown (Fig. 13).

ISSN 2076-2429 (print)
ISSN 2223-3814 (online)

To provide the automated translation functionality the program sends and receives web requests in the JSON (JavaScript Open Notation) format. Text information from the text strings is sent to the server using the HTTP protocol and POST request method, containing the selected language pair and personal API key. In case of success the response is received as a JSON formatted array of strings, which is then decoded and entered as the machine translation result to the list of user interface text in the program.

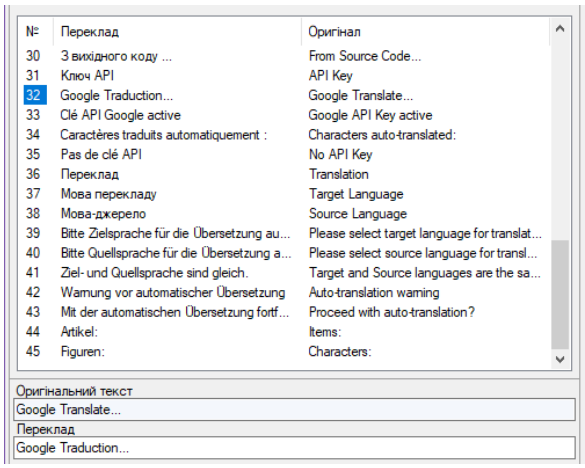


Fig. 11. Demonstrating multi-language translation (Source: compiled by the author)

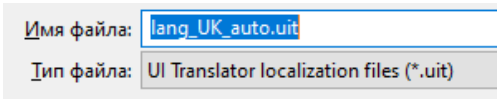


Fig. 12. .uit file type filter (Source: compiled by the author)

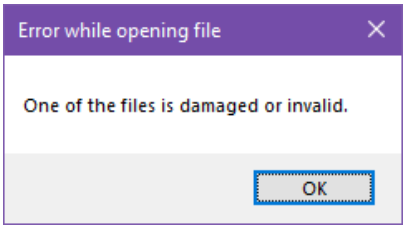


Fig. 13. Error message when opening localization file (Source: compiled by the author)

Conclusions

Based on the analysis of existing software products potential possibilities were found for the development of a program that implements functionality required by a particular set of users – independent software developers in need of a convenient method to translate user interfaces. With the understanding of these requirements a list of functions, methods and approaches to their use in the target program was described.

Based on this a functioning version of a user interface translation tool was conceptualized and developed. The created version of the program provides the required feature set and is ready to be used for the creation of localization for other software products. Translation automation is implemented using the online machine translation service Google Translate.

Potential directions for further development may include the ability to use other online machine translation services, as well as the ability to manually set the format (keyphrase) to detect strings of text in the source code.

Література

1. Top Translation Software Tools in 2020 – Some Even Free!. URL: https://www.smartcat.ai/blog/top-translation-software-tools-in-2019-some-even-free/. *Title from the screen*. (дата звернення: 25.12.2023).
2. Try the Free Website Translator & Translate API Google Translate. URL: https://translate.google.com/intl/en/about/forbusiness/. (дата звернення:25.12.2023).
3. gettext GNU Project Free Software Foundation (FSF). URL: https://www.gnu.org/software/gettext/. *Title from the screen*. (дата звернення:25.12.2023).
4. GNU gettext utilities: List of Programming Languages. URL: https://www.gnu.org/software/gettext/manual/html_node/List-of-Programming-Languages.html. Title from the screen. (дата звернення: 05.01.2024).
5. XLIFF Version 2.1. URL: https://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html. *Title from the screen*. (дата звернення: 05.01.2024).
6. Language support | Cloud Translation | Google Cloud. URL: https://cloud.google.com/translate/docs/languages. *Title from the screen*. (дата звернення:25.12.2023).
7. Pricing | Cloud Translation | Google Cloud. URL: https://cloud.google.com/translate/pricing. *Title from the screen*. (дата звернення:25.12.2023).

References

1. Top Translation Software Tools in 2020 – Some Even Free!. Retrieved from: https://www.smartcat.ai/blog/top-translation-software-tools-in-2019-some-even-free/. *Title from the screen.* (Last accessed: 25.12.2023).
2. Try the Free Website Translator & Translate API Google Translate. Retrieved from: https://translate.google.com/intl/en/about/forbusiness/. (Last accessed: 25.12.2023).
3. gettext GNU Project Free Software Foundation (FSF). Retrieved from: https://www.gnu.org/software/gettext/. *Title from the screen.* (Last accessed: 25.12.2023).
4. GNU gettext utilities: List of Programming Languages. Retrieved from: https://www.gnu.org/software/gettext/manual/html_node/List-of-Programming-Languages.html. *Title from the screen*. (Last accessed: 05.01.2024).
5. XLIFF Version 2.1. Retrieved from: https://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html. *Title from the screen*. (Last accessed: 05.01.2024).
6. Language support | Cloud Translation | Google Cloud. Retrieved from: https://cloud.google.com/translate/docs/languages. *Title from the screen*. (Last accessed: 25.12.2023).
7. Pricing | Cloud Translation | Google Cloud. Retrieved from: https://cloud.google.com/translate/pricing. *Title from the screen*. (Last accessed: 25.12.2023).

Матузко Володимир Дмитрович; Volodymyr Matuzko, ORCID: https://orcid.org/0000-0002-3005-6051

Received April 19, 2024

Accepted May 24, 2024