УДК 004.51
DOI https://doi.org/10.32689/maup.it.2024.1.9

Володимир МАТУЗКО
аспірант кафедри програмної інженерії,
Запорізький національний університет, matuzkovd@ukr.net
***ORCID:** 0000-0002-3005-6051*

АЛГОРИТМИ В ПРОГРАМНІЙ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОГО ПЕРЕКЛАДУ ІНТЕРФЕЙСУ ПРОГРАМ

Анотація. *Велика кількість повсякденних дій вже давно виконується за допомогою мобільних додатків та міжнародних ресурсів у мережі Інтернет. Через це у користувачів постає проблема знання мови для можливості користування цими програмами. Не всі розробники в світі мають доступ до професійних послуг перекладу або можливість створити належний переклад власноруч. Важливим та зазвичай вирішальним фактором є вартість програмного забезпечення для створення перекладу. Великі професійно-спрямовані програмні пакети мають вартості ліцензій, що вимірюються в сотнях доларів США. Також більшість з існуючих засобів спрямовані на переклад довільних текстів у загальній формі. Альтернатиним підходом є залучення команд перекладачів, але в цьому випадку треба враховувати додатковий час та обсяг перекладу, а також обмежений доступ до перекладачів на менш відомих мовах світу.*

Зважившиці фактори можна зазначити, що існує потреба в зручній та доступній програмі, що спеціалізована для створення та забезпечення якості перекладу саме інтерфейсів програмного забезпечення. Після проведеного аналізу визначені вимоги до запропонованої програми, а саме зручний інтерфейс та набір функцій, що спеціалізовані для роботи з програмними інтерфейсами та файлами вихідного коду.

***Мета роботи** – розробка програми з необхідним функціоналом для автоматизації перекладу інтерфейсів користувача.*

***Методологія.** Програмне забезпечення розроблене з використанням мови C# в середі Microsoft Visual Studio. Інтерфейс програми створено з урахуванням вимог розробників програмного забезпечення.*

***Наукова новизна.** Виявлено напрямки вдосконалення існуючих методів та засобів та розробка програмного забезпечення для надання нового засобу для перекладу. В даній роботі описано та розроблено комп'ютерну утиліту у версії для ОС Microsoft Windows, яка реалізує всі вимоги до базового функціоналу. Програма спеціалізована для перекладу інтерфейсів користувача та дозволяє автоматизувати цей процес. Оптимізовано та спрощено процес створення перекладу інтерфейсів для власних програмних розробок невеликих команд розробників шляхом повної прив'язки існуючого вихідного коду незалежно від використаної мови програмування.*

***Висновки.** Розроблену програму можна використовувати для створення та перекладу інтерфейсів користувача. Функція автоматизованого перекладу потребує володіння особистого ключа для користування Google Translate API.*

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

Volodymyr MATUZKO. ALGORITHMS IN SOFTWARE SOLUTION FOR AUTOMATED USER INTERFACE TRANSLATION

Abstract. *Numerous daily activities are long since accomplished using mobile applications and international resources available through the Internet. 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. An important and usually determining factor is the cost of translation software. Large professionally-targeted software packages have license fees measured in hundreds of US dollars. Also, the majority of existing solutions are meant for various kinds of general freeform text. An alternative approach would be hiring teams of translators, but this needs accounting for the extra time and size of translation, as well as limited availability of translators for the lesser known world languages.*

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 leads to confirmed requirements for the program, those being a comfortable user interface and a set of functions specific to working with user interfaces and source code files.

***The purpose of this work** is to develop software that meets the functionality requirements for automated user interface translation.*

***Methodology.** Software is developed using the C# language and Microsoft Visual Studio environment. The program's interface was designed according to needs of software developers.*

***Scientific novelty.** Determine ways and approaches to improve existing methods and tools, and the development of software to provide a new tool for translation. This work describes the development of such software tool for Microsoft Windows that implements every listed requirement for base functionality. The program is specialized for translation of user interfaces and enables automation of the process. The process of providing user interface translation for software created by small teams of developers were optimized and simplified via a full link to existing source code regardless of programming language used.*

***Conclusions.** The developed program can be used to create and translate user interfaces. Machine translation functionality requires possession of a personal key to utilize Google Translate API.*

***Key words:** machine translation, software development, user interface, Google Translate.*

Постановка проблеми. У сучасних умовах надзвичайно зростає роль синхронного перекладу як засобу, який обслуговує економічні, суспільно-політичні, наукові, культурно-естетичні та інші відносини народів світу. Тому проблема якісного перекладу є актуальною на даний час і вимагає розробки все нових методів та засобів синхронного перекладу.

Науково-технічний прогрес, який охоплює все нові сфери життя і пов'язані з нею міжнародне співробітництво наук, очікуваний демографічний вибух і інші найважливіші явища розвитку цивілізації призводять до небувалого розвитку різного роду контактів між державами та іншими різномовними товариствами людей. Глобалізація політики, економіки, виробництва й досліджень, а також об'єднання зусиль для подолання наслідків кризових явищ та катастроф планетарного масштабу жорстко поставили на порядок денний проблему забезпечення комунікації в умовах сучасної полілінгвокультурної світової спільноти. У таких умовах наріжним каменем взаємодії є оптимізація процесу комунікації, основним засобом якої завжди була й залишається природна мова. Ці та інші виклики, які з часів промислової революції постали перед людством, стимулювали створення новітніх напрямків лінгвістики, спрямованих на вивчення функціональних аспектів природної мови на кшталт лінгвістики фахових мов, лінгвістики тексту тощо, а також виникнення суміжних дисциплін, що сформувались в результаті спеціалізації й інтеграції наукових досліджень з метою оптимізації процесу комунікації: термінознавство, психолінгвістика, когнітологія, комп'ютерна лінгвістика, корпусна лінгвістика, штучний інтелект тощо. Поступова імплементація доробків цих наук та накопичення серйозного масиву лінгвістичних ресурсів результували виокремленням комплексного наукового напрямку — автоматизованого опрацювання природної мови. До пошуку нових можливостей вирішення проблеми глобальної комунікації стимулювала усіх зацікавлених і еволюція інформаційних та мережевих технологій [1].

Синхронний переклад також присутній у світі комп'ютерних інтерфейсів. Особливо гостро питання стоїть серед користувачей мобільних додатків, тому що кількість смартфонів та інших подібних пристроїв вимірюється мільярдами [5]. Велика кількість цих додатків має версії лише на одній мові, або використовує машинний переклад без перевірки коректності тексту та урахування контексту в інтерфейсі. Також якість машинного перекладу також залежить від конкретної мовної пари. Як приклад, Китай є одним з найбільших постачальників сучасних мобільних додатків. До того ж, машинний переклад з китайської історично відрізнявся своєю складністю у порівнянні з перекладом між європейськими мовними групами. Дослідження та розробки у цьому напрямку відбуваються постійно [2]. Щодо інтерфейсів програм для комп'ютерів ситуація дуже схожа, але зазвичай розробники користуються допомогою команд перекладачів, тим самим уникаючи ситуацій з низькою якістю перекладу. Однак ці перекладачі мають виконувати переклад та знаходити методи і засоби для цього власноруч.

Одним з можливих варіантів реалізації такого функціоналу є використання існуючого формату .PO, який використовується в системі локалізації GNU gettext [3]. При використанні цього алгоритму розробник генерує PO-файл, який буде містити набір рядків, відповідних тексту інтерфейсу його програми. Для цього розробник має форматувати рядки з текстом згідно до вимог алгоритму gettext – кожний рядок має бути у вигляді функції, для якої вхідними даними є сам текст інтерфейсу на деякій базовій мові, а як вихіні дані повертає відповідний текст з доступного набору даних для перекладу. Використання цього функціоналу також потребує від розробника залучення та налаштування бібліотеки GNU. Gettext, як у випадку з мовою програмування C# [4].

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

Рис. 1. Формат запису одного рядку в файлі PO

Отриманий таким чином файл PO потім можна відкрити в утиліті для перекладу. Прикладом вже існуючої такої програми-утиліти є Poedit [6]. Poedit дозволяє користувачу редагувати ці файли

в інтерфейсі, схожому на інші існуючі програми для перекладу. Також Poedit надає вбудовану можливість використання онлайн-сервісів для машинного перекладу, таких, як Microsoft Translator та Google Translate, але цей функціонал наявний лише в платних версіях програми [7]. Poedit націлений на перекладачів, які працюють виключно з файлами перекладу, тому в ньому відсутня можливість переглянути вихідний код для отримання повного розуміння, де і як використовується текст.

Після завершення роботи над перекладом розробнику потрібно конвертувати отриманий PO-файл в коректний формат для обраної мови програмування за допомогою інструментів GNU gettext. Також потрібно зазначити, що автоматизація за допомогою gettext реалізована тільки для конкретного переліку популярних мов програмування, тому цей алгоритм важко назвати універсальним рішенням для всіх розробників.

Аналіз останніх досліджень і публікацій. В [1, 2, 5] розглянуто досягнутий прогрес в сфері машинного перекладу та визначенно актуальні проблеми. Доцільність розробки альтернативних алгоритмів доведена на основі існуючих рішень, що описано в ресурсах [3, 4, 6, 7].

Постановка завдання. Розробити алгоритм та відповідну програмну реалізацію автоматизації перекладу інтерфейсів комп'ютерного програмного забезпечення, спрямовані на використання невеликими командами розробників та незалежно від обраної ними мови програмування. Для досягнення цієї мети поставлено завдання визначити сильні і слабкі сторони існуючих сучасних методів та засобів перекладу, а також їх варіанти застосування; сформувати перелік технічних вимог.

Виклад основного матеріалу дослідження. Однією з головних вимог до розробленої програми є можливість завантаження та взаємодії з вихідним кодом для забезпечення коректного перекладу елементів інтерфейсу. Для цього розробнику потрібно наперед підготувати код згідно вимог утиліти, а саме зберігати всі рядки зі змістом тексту інтерфейсу в належній формі. Головною перевагою такого підходу є майже повна незалежність від обраної мови програмування.

Вимоги поточної версії перекладача до форматування:

- двовимірний масив рядків розміром [кількість_мов, кількість_рядків];
- локалізація рядків інтерфейсу у вигляді = UITranslator_language[0,##], де 0 – порядковий номер першої мови локалізації, а ## – порядковий номер рядку в файлі локалізації.

Після завантаження вихідний код можна переглянути в правому вікні утиліти. Коректно оформлені рядки локалізації виділені синім кольором в тексті. При створенні нового проекту на основі файлу вихідного коду додатково відбувається створення порожніх рядків для заповнення в вікні для перекладу в кількості, що відповідає кількості коректно оформлених рядків в завантаженному коді.

Цей алгоритм створений для зручності пошуку та надання контексту в вихідному коді при перекладі тексту інтерфейсу програми. Для цього лише потрібно двічі натиснути на порядковий номер в списку – курсор в вікні коду буде переміщено до відповідного за номером рядку локалізації.

Інтерфейс програми забезпечує створення перекладу за допомогою таблиці з двома стовпчиками, які відповідають обраній мовній парі. Перший стовпчик містить текст для редагування, другий стовпчик містить оригінал тексту на іншій мові. Кожен рядок таблиці відповідає окремому рядку в масиві для локалізації, що знаходиться в завантаженному файлі вихідного коду програми.

Для початку роботи та створення перекладу потрібно ініціювати файл перекладу одним з трьох шляхів:

- створити новий проект на базі файлу вихідного коду;
- створити новий проект на базі існуючого файлу локалізації;
- відкрити два існуючі файли локалізації для редагування.

Після завантаження коректно оформлених файлів розробник може створювати та редагувати текст інтерфейсу для своєї програми. В перших двох випадках надаються порожні рядки для створення нового тексту, який потім зберігається у вигляді нового файлу локалізації.

Іншою важливою функцією утиліти є надання автоматизованого перекладу. Для цього потрібен завантажений існуючий файл локалізації в правому стовпчику таблиці. Цей файл використовується як джерело рядків на мові оригіналу в обраній мовній парі. Для початку автоматизованого перекладу розробник надає свій особистий ключ API для доступу до обраного сервісу перекладу (на даний момент реалізовано доступ до Google Translate), та обирає мовну пару в переліку доступних. Після цього треба лише обрати потрібні рядки для перекладу та натиснути відповідну кнопку.

Машинний переклад виконується за допомогою HTTP-запитів до онлайн-сервісу. Обрані рядки форматуються в необхідній формі та відсилаються у запиті. Для створення запиту до Google Translate необхідно використати нотацію JSON. Приклад інформації в запиті на переклад для N обраних рядків:

{“q”: [
"текст-рядку1", "текст-рядку2", "текст-рядку3", ... , “текст-рядкуN”
] }

Також в HTTP-запиті вказано індивідуальний API-ключ розробника, початкову та цільову мову перекладу, а також вказівка, що текст надається в базовому текстовому форматі. Таким чином можна перекласти до 128 рядків одразу (Google, 2024). При успішності запиту від онлайн-сервісу надходить HTTP-відповідь також у форматі JSON:

{ “data”: {
“translations”: [
{ “translatedText”: “переклад-тексту-рядка1” },
{ “translatedText”: “переклад-тексту-рядка2” },
{ “translatedText”: “переклад-тексту-рядка3” },
...
{ “translatedText”: “переклад-тексту-рядкаN” }]
} }

На рис. 2 показано як отриманий результат декодується і вставляється у відповідні запиту рядки лівого стовпчика таблиці. Також програма відслідковує кількість перекладених символів за поточну сесію.

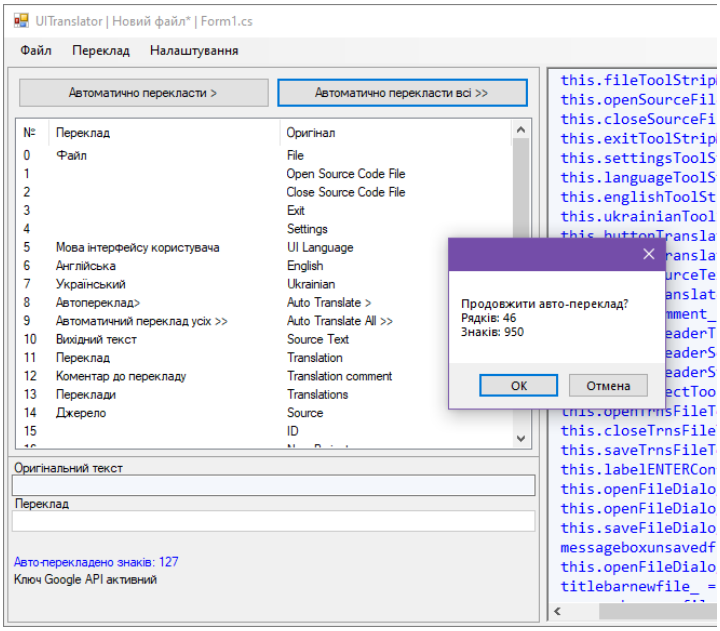

Рис. 2. Вікно програми під час використання машинного перекладу

Результатом роботи утиліти для перекладу є файл локалізації. Цей файл має розширення .uit та містить в собі рядки тексту та їх порядковий номер (починаючи з 0) в звичайному текстовому вигляді для легкої взаємодії з будь-якою програмою:

str0 “текст-рядку1”
str1 “текст-рядку2”
str2 “текст-рядку3”
...
strN “текст-рядкуN+1”

Один файл локалізації відповідає одній мові інтерфейсу користувача. Розробник має лише на свій розсуд налаштувати свою програму для коректного завантаження файлів локалізації в відповідні масиви рядків, які потім можна використовувати для зміни мови інтерфейсу в тому числі і під час роботи програми. Зразок оформлення коду в раніше наведеному форматі:

```
UIElement1.Text = UITranslator_language[0,0];
UIElement2.Text = UITranslator_language[0,1];
UIElement3.Text = UITranslator_language[0,2];
...
UIElementN+1.Text = UITranslator_language[0,N];
```

В цьому прикладі UITranslator_language є двовимірним масивом, що містить в собі всі доступні файли локалізації. Перший номер відповідає номеру мови, другий номер є порядковим номером рядку в файлі локалізації, N – загальна кількість рядків.

Як зазначено вище, результатом роботи розробленої утиліти є файли локалізації, що повністю готові до використання розробником в програмному проекті. До того ж, у порівнянні з алгоритмом gettext відсутня необхідність так би мовити початкової мови інтерфейсу, що незмінно наявна в вихідному коді програми. При використанні розробленої програми зменшується кількість необхідних етапів для отримання якісного інтерфейсу користувача на багатьох мовах світу, та незалежно від обраної мови програмування.

Висновки з даного дослідження та перспективи подальшого розвитку в даному напрямі. В даній статті описано алгоритми роботи першої початкової версію програми для відносно швидкого створення тексту інтерфейсів користувача перекладу. Головним завданням було спроектувати та створити програму, яка продемонструє весь необхідний для цього функціонал на належному рівні зручності.

Описана версія програми має простір для додавання нового фукнціоналу в подальшому. Запропонований формат масивів є лише одним з можливих варіантів оформлення. В подальших версіях утиліти розробник зможе сам обирати зручний синтаксис коду, за умови використання нумерованого переліку рядків. Також є потенціал для підтримки роботи з іншими сервісами машинного перекладу.

Список використаних джерел:

1. Міщенко А. Л. Лінгвістика фахових мов та сучасна модель науково-технічного перекладу : монографія. Вінниця : Нова Книга, 2013. 448 с

2. Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, et al. Achieving Human Parity on Automatic Chinese to English News Translation. arXiv:1803.05567 [cs.CL] URL: https://arxiv.org/abs/1803.05567 (дата звернення: 30.03.2024).

3. gettext GNU Project Free Software Foundation (FSF). URL: https://www.gnu.org/software/gettext/ (дата звернення: 03.04.2024).

4. C# (GNU gettext utilities). URL: https://www.gnu.org/software/gettext/manual/html_node/C_0023.html (дата звернення: 03.04.2024).

5. How Many People Have Smartphones Worldwide. URL: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world (дата звернення: 30.03.2024).

6. Poedit Translation Editor. URL: https://poedit.net/ (дата звернення: 03.04.2024).

7. Purchase Poedit Pro – Poedit. URL: https://poedit.net/purchase (дата звернення: 03.04.2024).