

Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Marcin Świnoga

Nr albumu: 181099

**Narzędzia wspomagające
tłumaczenie tekstów
informatycznych**

Praca magisterska
na kierunku INFORMATYKA

Praca wykonana pod kierunkiem
dr. hab. Janusza S. Bienia, prof. UW
Katedra Lingwistyki Formalnej UW

Sierpień 2006

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data

Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data

Podpis autora (autorów) pracy

Streszczenie

W pracy przedstawiono techniki i narzędzia wspomagające tłumaczenie tekstów. Szczególną uwagę poświęcono tłumaczeniu ogólnie pojętych tekstów informatycznych. Praca poświęcona została również stworzeniu narzędzia, umożliwiającego tworzenie słowników na podstawie pamięci tłumaczeniowych.

Słowa kluczowe

tłumaczenie tekstów, CAT, TM, pamięć tłumaczeniowa, słowniki, wspomaganie tłumaczenia

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.0 Matematyka, Informatyka:

11.3 Informatyka

Klasyfikacja tematyczna

J. Computer Applications

J.5 Arts and Humanities

Linguistics

Tytuł pracy w języku angielskim

Tools supporting translation of texts related to Information Technology

Spis treści

Wprowadzenie	5
1. Wstęp	7
1.1. Podstawowe pojęcia	7
1.2. Zarys historyczny	9
2. Internacjonalizacja aplikacji	13
2.1. Gettext i pliki <i>po</i>	13
2.2. <i>ResourceBundle</i> i pliki <i>properties</i>	15
3. Tłumaczenie programów i dokumentacji	17
3.1. Tłumaczenie tekstu przy użyciu programu CAT	17
3.2. Projekt tłumaczeniowy dokumentacji Debiana	19
3.2.1. Projekt Międzynarodowy	20
3.2.2. Polski projekt tłumaczenia dokumentacji Debiana	20
3.3. Tłumaczenie środowiska graficznego KDE	21
3.4. Fedora Translation Project	22
4. Narzędzia wspomagające tłumaczenie tekstów	23
4.1. Swobodne oprogramowanie	23
4.1.1. Podział tekstu na segmenty	23
4.1.2. Tłumaczenie tekstu	24
4.1.3. Scalenie dokumentu i ostateczna weryfikacja	25
4.2. Słowniki	25
4.2.1. Oprogramowanie i źródła dostępne w Internecie	25
4.2.2. Protokół DICT	26
5. Kreator Słowników	27
5.1. Podział na moduły	27
5.2. Stosowane wzorce i konwencje	29
5.2.1. Logowanie zdarzeń	29
5.2.2. ShutdownHook	30
5.2.3. Wzorzec <i>Singleton</i>	31
5.2.4. Internacjonalizacja	31
5.3. Interfejs użytkownika	32
5.3.1. SourceTuvViewer	33
5.3.2. TargetTuvViewer	33
5.3.3. SegmentViewer	33
5.3.4. TranslationViewer	33

5.3.5. StatusBar	33
5.4. Możliwości rozbudowy programu	34
5.4.1. Obsługa plików słownika	35
5.4.2. Obsługa plików pamięci tłumaczeniowej	37
5.4.3. Dzielenie segmentów na wyrazy	38
5.4.4. Analiza morfologiczna	39
Podsumowanie	41
A. Kreator Słowników - Podręcznik Użytkownika	43
A.1. Wstęp	43
A.2. Opis programu	43
A.3. Interfejs użytkownika	45
A.3.1. Podgląd pamięci tłumaczeniowej	45
A.3.2. Aktywny segment pamięci tłumaczeniowej	47
A.4. Edycja słowników	49
A.4.1. Usuwanie tłumaczeń ze słownika	49
A.4.2. Dodawanie tłumaczeń do słownika	49
A.5. Otwieranie pamięci tłumaczeniowej	49
A.6. Otwieranie słownika	51
A.7. Zapisywanie słowników	51
A.8. Menadżer słowników	52
A.9. Łączenie wyrazów	53
A.10. Tworzenie słownika krok po kroku	53
A.11. Pytania i odpowiedzi (FAQ)	55
B. Wybrane formaty plików	57
B.1. debiandoc-sgml	57
B.2. docbook	58
B.3. wml	59
Bibliografia	61

Wprowadzenie

W ramach niniejszej pracy napisany został program umożliwiający tworzenie słowników w oparciu o pamięci tłumaczeniowe, powstające podczas pracy z oprogramowaniem typu CAT. Powstała również nowa dystrybucja Linuxa oparta na systemie KNOPPIX¹ o nazwie LINGUATRIX. Choć istnieje już kilka dystrybucji Linuxa związanych z przetwarzaniem języka naturalnego², wydaje się, że ta jest pierwszą poświęconą zagadnieniu komputerowego wspomaganie tłumaczenia tekstów. Pierwsza wersja tej dystrybucji zawiera między innymi programy OmegaT oraz Kolokacje. Na płycie znajduje się również niniejsza praca magisterska oraz program „Kreator Słowników”. Ponadto, w pracy zostały przedstawione narzędzia wspomagające pracę tłumaczy. Wśród przedstawionych narzędzi znalazły się zarówno zwykłe edytory tekstu, jak i wyszukiwarki, słowniki, czy zaawansowane programy automatyzujące proces tłumaczenia.

W pierwszej części pracy staram się usystematyzować pojęcia związane z komputerowym wspomaganie tłumaczenia tekstów oraz przedstawić krótko historię użycia komputera do ich tłumaczenia.

Druga część pracy poświęcona jest praktyce tłumaczenia wspomaganego komputerowo. Szczególny nacisk położono na tłumaczenie tekstów informatycznych, do których zaliczyć można tak różne dokumenty, jak projekty techniczne, wydawnictwa książkowe czy interfejs użytkownika w systemach komputerowych. Przedstawiono tu oraz porównano kilka programów wspomagających tłumaczenie wspomnianych tekstów. W tej części pracy, na przykładzie między innymi projektu tłumaczeniowego Debiana, pokazano również, jak wygląda codzienna praca ludzi zaangażowanych w tłumaczenie otwartego oprogramowania oraz jego dokumentacji.

Trzecia część pracy poświęcona jest roli słowników i glosariuszy jako komponentów systemów wspomagających tłumaczenie. Omówione zostały standardy używane obecnie do przechowywania danych terminologicznych, programy oraz formaty plików przechowujących te dane. W tej części przedstawiono również program „Kreator Słowników”.

Jako dodatek, do pracy dołączona została instrukcja użytkownika stworzonego programu. Instrukcja, jako niezależny dokument, zawiera powtórzenie definicji pojęć używanych w treści. Program „Kreator Słowników” udostępniony jest na zasadach licencji GNU General Public License, natomiast tekst niniejszej pracy oraz instrukcji użytkownika programu na zasadach licencji GNU Free Documentation License. Treści licencji dostępne są pod adresem: <http://www.gnu.org/licenses/>.

¹KNOPPIX - dystrybucja linuxa na płycie typu Live-CD/Live-DVD umożliwiająca uruchomienie systemu operacyjnego wprost z płyty CD/DVD.

²wśród dystrybucji poświęconych przetwarzaniu tekstu wymienić można *Knorpora*[KNR01], *Morphix-NLP*[MOR01] czy *Collocatrix*[COL01]

Rozdział 1

Wstęp

Niniejszy rozdział poświęcony został usystematyzowaniu pojęć używanych w pracy oraz przedstawieniu historii tłumaczenia z użyciem komputera.

1.1. Podstawowe pojęcia

Tłumaczenie maszynowe (ang. machine translation, MT) – automatyczne tłumaczenie tekstu przez system komputerowy. Przetłumaczony w ten sposób tekst poddawany jest czasami korekcie.

CAT (ang. computer assisted translation) – proces, w którym człowiek tłumaczy tekst przy użyciu programów komputerowych wspomagających ten proces. Najczęściej programy CAT kojarzone są z systemami opartymi na pamięciach tłumaczeniowych.

Segment – spójny segment tekstu służący jako jednostka tłumaczeniowa w systemach opartych na pamięciach tłumaczeniowych. Segmenty najczęściej stanowią zdania i paragrafy.

Segmentacja - proces podziału tekstu na segmenty. Najczęściej system komputerowy przeprowadza automatycznie dopasowanie wstępne, po którym umożliwia użytkownikowi weryfikację i wprowadzenie poprawek.

TM - patrz Pamięć Tłumaczeniowa

Pamięć tłumaczeniowa (ang. translation memory) – baza danych zawierająca segmenty tekstu źródłowego i jego tłumaczenia, w formie powiązanych ze sobą krotek zawierających odpowiednie segmenty z obu tekstów. Pamięć tłumaczeniowa wykorzystywana jest do wyszukiwania tłumaczeń segmentów w tłumaczonym tekście. Popularnymi formatami plików przechowujących pamięci tłumaczeniowe są TMX oraz XLIFF.

TMX (ang. Translation Memory eXchange format) - jest formatem standaryzującym opisywanie pamięci tłumaczeniowych. Format umożliwia wymianę danych między różnymi narzędziami i/lub twórcami oprogramowania.

XLIFF (ang. XML Localisation Interchange File Format) - format umożliwiający przechowywanie danych lokalizacyjnych aplikacji niezależnie od narzędzi używanych do internacjonalizacji aplikacji. Możliwe są różne reprezentacje danych zawartych w plikach XLIFF, jak na przykład pliki *properties* używane w Javie czy pliki *po* używane przez pakiet *gettext*.

Dopasowanie (ang. alignment) – proces segmentacji tekstu źródłowego i jego tłumaczenia oraz łączenie segmentów w odpowiadające sobie pary. Dopasowanie tekstów umożliwia automatyczne generowanie pamięci tłumaczeniowych na podstawie już istniejących tłumaczeń. Powstałą w ten sposób pamięć tłumaczeniową można wykorzystać do tłumaczenia kolejnych tekstów.

Jednostka tłumaczeniowa (ang. translation unit, TU) – zbiór segmentów będących swoimi odpowiednikami w różnych językach. Najczęściej spotykane są jednostki tłumaczeniowe składające się z dwóch segmentów w różnych językach. Przy dużych wielojęzycznych projektach spotyka się również pamięci tłumaczeniowe z wielojęzycznymi jednostkami tłumaczeniowymi.

Trafienie pełne – zdarzenie polegające na odnalezieniu w pamięci tłumaczeniowej segmentu identycznego z aktualnie tłumaczonym. Często jako trafienie pełne traktuje się trafienie, w którym segment z pamięci tłumaczeniowej oraz segment z tłumaczonego tekstu różnią się jedynie liczbami i/lub datami.

Trafienie częściowe (ang. fuzzy match) – zdarzenie polegające na odnalezieniu w pamięci tłumaczeniowej segmentu podobnego do aktualnie tłumaczonego. Podobieństwo segmentów wyrażone jest zazwyczaj w procentach, a obliczane jest przy pomocy różnych algorytmów. Najprostszy sposób obliczania podobieństwa segmentów oparty jest na miarze odległości Levenshtein'a. Miara Levenshtein'a określana jest przez liczbę operacji (dodanie, zamiana, usunięcie słowa) potrzebnych do zamiany jednego tekstu na drugi.

Słownik - zbiór wyrazów ułożonych i opracowanych według pewnej zasady, zwykle alfabetycznie, najczęściej objaśnianych pod względem znaczeniowym i ilustrowanych przykładami użycia. Istnieją również słowniki dwujęzyczne, w których podane są wyrazy jednego języka i odpowiadające im wyrazy innego (zwykle wraz z ich znaczeniami i użyciami).

Glosariusz - specjalistyczny słownik, zbiór glos, tj. tłumaczeń oraz wyjaśnień zwrotu lub wyrazu niejasnego bądź obcego. Najczęściej dotyczy specjalistycznej dziedziny wiedzy.

Internacjonalizacja (i18n) – proces dostosowania oprogramowania do obsługi wielu języków.

Lokalizacja (i10n) – proces tworzenia wersji językowej oprogramowania.

Gettext – zestaw narzędzi dla programistów i tłumaczy ułatwiający tworzenie programów obsługujących wiele języków oraz lokalizację tych programów. Komunikaty w programach wykorzystujących pakiet *gettext* przechowywane są w plikach *po* zawierających tłumaczenia komunikatów w różnych językach. Pod pojęciem *gettext* przyjęło się również rozumieć format wspomnianych plików.

LISA (ang. Localisation Industry Standard Association) – wiodące forum międzynarodowe dla firm działających na rynku globalnym, określające otwarte standardy dotyczące tłumaczeń i lokalizacji oraz obsługi i wsparcia klientów, produktów i usług świadczonych na całym świecie.

Przeciągnij i Upuść (ang. Drag and Drop) - mechanizm używany w graficznych interfejsach użytkownika do osadzania obiektów w komponentach programu, np. przeciągnięcie

ikony reprezentujące plik tekstowy do okna edytora tekstu skutkuje otwarciem tego pliku do edycji.

1.2. Zarys historyczny

Pierwsze pomysły dotyczące automatyzacji tłumaczenia tekstów pojawiły się już w XVII wieku, kiedy to niemiecki mnich, Johannes Becher, opisał matematyczny język opisu zdań w dowolnym języku. Zakodowany tekst składał się z równań matematycznych, których znaczenie było takie samo w każdym języku.

Realne możliwości użycia maszyny do tłumaczenia tekstów pojawiły się dopiero z nadejściem XX wieku, kiedy to w połowie lat 30, Georges Artsrouni oraz Petr Troyanskii zgłosili patenty „maszyn tłumaczących”. Skonstruowali oni niezależnie mechaniczne maszyny odczytujące wyrażenia zapisane na taśmach dziurkowanych w jednym języku i zapisujące je w innym języku na drugiej taśmie.

Niedługo po pojawieniu się pierwszych komputerów rozpoczęto badania nad możliwością wykorzystania ich do tłumaczenia naturalnego języka. Początek tych działań datuje się na lipiec 1949, kiedy to Warren Weaver z Rockefeller Foundation napisał memorandum, w którym przedstawił pomysły oparte na doświadczeniach związanych z łamaniem kodów podczas Druhej Wojny Światowej oraz dokonaniach Claude’a Shannon’a w dziedzinie teorii informacji. Weaver zakładał, że dokumenty pisane językiem naturalnym można traktować jak szyfry. Jego rozumowanie oparte było na fakcie, że skoro szyfry można odszyfrować, to teksty pisane w języku naturalnym również - tzn. przełożyć z jednego języka na drugi, stosując techniki znane przy rozszyfrowywaniu zakodowanych wiadomości. Oto fragment memorandum zawierający to stwierdzenie:

Mam przed sobą tekst, który jest napisany po rosyjsku. Będę jednak udawał, że on jest napisany w języku angielskim i został zakodowany przy pomocy dziwnych symboli. Wszystko, co muszę zrobić, aby dotrzeć do informacji zawartej w tekście, to odkodować szyfr.

tłum. z angielskiego Marcin Świnoga

W 1954 roku jeden z projektów badawczych, oparty na współpracy firmy IBM oraz Uniwersytetu Georgetown, dotyczących automatycznego tłumaczenia tekstów zakończył się prezentacją możliwości skonstruowanego systemu tłumaczenia maszynowego. Choć program działał na bardzo ograniczonym zasobie słownikowym (250 słów), zainteresował badaczy swoimi możliwościami i stał się inspiracją do powstania kolejnych projektów badawczych na całym świecie.

Pierwsze systemy tłumaczenia maszynowego składały się głównie z dużych dwujęzycznych słowników oraz pewnego zbioru reguł słownikowych, mówiących o poprawnej kolejności słów w zdaniu. Szybko zorientowano się, że reguły słownikowe były zbyt skomplikowane - powstała potrzeba stworzenia bardziej systematycznych metod analizy syntaktycznej. Wiele projektów oparło swoje badania na ówczesnych osiągnięciach w dziedzinie lingwistyki, a szczególnie tych, dotyczących modeli gramatyk formalnych. Właśnie ten kierunek rozwoju systemów tłumaczenia maszynowego wydawał się mieć przed sobą możliwości daleko idącej poprawy jakości tłumaczeń.

Przez pierwsze dziesięciolecie badaniom towarzyszył wysoki poziom optymizmu oraz zapowiedzi rychłych przełomów. Niestety, wraz z napotykanymi problemami związanymi ze zmieniającą się semantyką słów, z którymi nie można było sobie poradzić w prosty sposób, rosło rozczarowanie wynikami badań.

Jakość tłumaczeń nie była zadowalająca, mimo że wdrożono kilka systemów tłumaczenia maszynowego - Mark II zbudowany przez IBM oraz Uniwersytet Washington, system uniwersytetu Georgetown oraz Euraton. Jedynie wąska grupa odbiorców, zainteresowana szybkim otrzymywaniem tłumaczeń o niekoniecznie najwyższej jakości, była zadowolona z dostarczonych narzędzi.

W 1964 roku, po tym jak pogłębiły się złe nastroje wokół prowadzonych badań, rząd Stanów Zjednoczonych powołał komitet o nazwie ALPAC (ang. Automatic Language Processing Advisory Committee), który w 1966 roku wydał słynny raport podsumowujący wyniki prac nad rozwojem tłumaczenia maszynowego. Raport, nazywany "Czarną Księgą Tłumaczenia Maszynowego", zawierał informacje mówiące o tym, że tłumaczenie automatyczne jest wolniejsze, niedokładne oraz dwukrotnie droższe od tłumaczeń wykonywanych metodami tradycyjnymi. W raporcie pojawiła się również informacja o braku perspektyw na zadowalające wyniki tłumaczeń wykonywanych maszynowo. Komitet nie widział sensu dalszych inwestycji w technologii związane z automatycznym tłumaczeniem tekstów. Zamiast tego, rekomendował rozwój systemów wspomagających (machine aids) tłumacza, takich, jak na przykład słowniki elektroniczne, oraz dalsze wspieranie badań nad lingwistyką komputerową (ang. computational linguistics). Choć raport został uznany za krótkowzroczny, to miał on kluczowy wpływ na zakończenie finansowania projektów badawczych związanych z MT w Stanach Zjednoczonych przez kolejne dziesięciolecie. Wpłynął również na zmniejszenie zainteresowania tymi systemami w Rosji i Europie. Rekomendacja rozwoju systemów wspomagających tłumacza wpłynęła natomiast pozytywnie na rozwój narzędzi CAT (ang. Computer-assisted-translation lub Computer-aided-translation).

Choć koncepcje dotyczące tłumaczenia maszynowego i tłumaczenia wspomaganego komputerowo są podobne, różnią się rolą człowieka w obu procesach oraz zakresem prac przeprowadzanych przez system komputerowy. Tłumaczenie maszynowe polega na wykonaniu tłumaczenia przez system komputerowy na podstawie zawartych w nim reguł, bardzo często uniemożliwiając wręcz weryfikację tłumaczenia przez człowieka. Tłumaczenie wspomaganie komputerowo to proces, w którym człowiek tłumaczy tekst wykorzystując systemy komputerowe jako pomoce automatyzujące prace, które musiałby wykonywać samodzielnie. Idąc za taką definicją, określenie „tłumaczenia wspomaganego komputerowo” jest dość obszerne i pozwala na wyodrębnienie przynajmniej kilku sposobów jego realizacji:

- sprawdzanie pisowni - zarówno wbudowane do edytorów tekstów, jak i samodzielnych programów,
- sprawdzanie gramatyki – podobnie jak w przypadku sprawdzania pisowni może być realizowane przez funkcje wbudowane w edytory tekstów, jak i przez samodzielne programy,
- zarządzanie terminologią - umożliwiają tłumaczowi zarządzanie używaną przez niego terminologią w formie elektronicznej. Używane mogą być zarówno proste tabele stworzone przez tłumacza w edytorze tekstów lub arkuszu kalkulacyjnym, bazy danych stworzone na przykład w OpenOffice.org Base lub Microsoft Access, jak i specjalistyczne oprogramowanie takie jak TRADOS MultiTerm, Terminotix LogiTerm czy Mercury Termex,
- słowniki elektroniczne – zarówno jednojęzykowe, jak i wielojęzykowe, dostępne jako programy instalowane na komputerze oraz dostępne przez Internet, np.:

– <http://www.dict.pl>,

- Słownik Języka Polskiego PWN
(<http://sjp.pwn.pl>),
 - (<http://dictionary.reference.com/>),
 - Słownik Informatyczny
(http://www.idg.pl/slownik/index_alfabetyczny.asp?sownik=pa),
 - Słownik terminologii internetowej
(<http://zls.mimuw.edu.pl/~alx/slownik/slownik.html>),
 - Słownik wyrażzeń związanych z bezpieczeństwem sieci
(<http://zls.mimuw.edu.pl/~robmar/slownik.html>),
- bazy terminologiczne – dostępne na płytach CD oraz przez Internet, np.:
 - The Open Terminology Forum (www.terminologie.com),
 - Grand Dictionnaire Terminologique (www.granddictionnaire.com),
 - listy dyskusyjne - stanowią źródło informacji, a zarazem umożliwiają wymianę doświadczeń i wiedzy pomiędzy tłumaczami, np.:
 - Lanta-L - międzynarodowe forum dotyczące wszystkich aspektów tłumaczenia ustnego i pisemnego (<http://www.geocities.com/Athens/7110/lantra.htm>),
 - U-forum - niemieckojęzyczna lista dyskusyjna dotycząca głównych problemów związanych z zawodowym tłumaczeniem ustnym i pisemnym
(<http://www.techwriter.de/thema/u-forum.htm>),
 - U-cat - niemieckie forum dyskusyjne dotyczące w szczególności wszystkich pytań o narzędzia tłumaczeniowe (<http://www.techwriter.de/thema/u-cat.htm>),
 - TW_Users - międzynarodowe forum dyskusyjne dotyczące wszystkich aspektów użycia narzędzi tłumaczeniowych programu Trados (<http://groups.yahoo.com/>).
 - wyszukiwarki pełnotekstowe - umożliwiają tłumaczowi przeszukiwanie już przetłumaczonych tekstów oraz odnajdywanie dokumentów powiązanych
 - konkordancery – programy służące do tworzenia konkordancji na podstawie korpusu,
 - biteksty¹ – umożliwiają tłumaczowi przechowywanie i odwoływanie się do przetłumaczonych dokumentów,
 - systemy oparte na pamięciach tłumaczeniowych – umożliwiają automatyzację tłumaczenia tekstów przez ich podział na mniejsze fragmenty zwane segmentami (zazwyczaj zdania) i podpowiadanie tłumaczenia użytkownikowi, jeśli przetłumaczył on wcześniej (i zapamiętał w pamięci tłumaczeniowej) segment podobny do aktualnie tłumaczonego,

¹Bitekst (ang. bitext) – rezultat połączenia tekstu źródłowego z tłumaczeniem. Biteksty oparte są na stosunkowo nowym pomysle Briana Harrisa, który napisał pracę poświęconą temu zagadnieniu w 1988r. Choć bitekst jest podobny do pamięci tłumaczeniowej, jest między nimi znacząca różnica. Polega ona na tym, że pamięć tłumaczeniowa jest bazą, w której segmenty tekstu pozbawione są kontekstu, a ich kolejność jest utracona. Bitekst natomiast zachowuje oryginalną kolejność segmentów. Ponadto bitekst przeznaczony jest do przeszukiwania go przez człowieka i nie służy do automatycznego tłumaczenia tekstu, co ma miejsce w przypadku pamięci tłumaczeniowych. Można powiedzieć, że bitekst to jednostka tłumaczeniowa, której segmentami są całe dokumenty: źródłowy i przetłumaczony.

- rozpoznawanie tekstu - narzędzia OCR² rozpoznające tekst z map bitowych (np. zeskanowanych dokumentów). Narzędzia te często stosowane są w fazie przygotowawczej tłumaczenia w przypadku, gdy klient nie może dostarczyć dokumentów w formie edytowalnych plików tekstowych.

²OCR (ang. optic character recognition) – proces umożliwiający przekształcenie map bitowych (np. zeskanowanych dokumentów, plików zapisanych w formacie PostScript lub PDF) na dokumenty tekstowe

Rozdział 2

Internacjonalizacja aplikacji

Jeszcze kilkanaście lat temu bardzo utrudnione było tworzenie oprogramowania, którego lokalizacja byłaby łatwa i szybka. Większość programów najczęściej zawierała komunikaty porzucane po wielu plikach kodu źródłowego. Lokalizacja takiego oprogramowania była bardzo żmudna, ponieważ wymagała od tłumacza przejrzenia wszystkich plików źródłowych w poszukiwaniu komunikatów oraz ich przetłumaczenia. Dopiero po kompilacji kodu źródłowego z przetłumaczonymi komunikatami dostępna była zlokalizowana wersja programu. Co więcej, w przypadku programów kompilowanych do kodu maszynowego, zlokalizowany program należało przekompilować dla każdej wersji systemu operacyjnego, na którym miał on działać. Tak skomplikowany proces sprawiał, że niewiele aplikacji było dostępnych w więcej niż jednym języku.

Powszechna chęć i potrzeba udostępniania aplikacji użytkownikowi w zrozumiałym dla niego języku wymusiła powstanie narzędzi ułatwiających pisanie aplikacji, które można łatwo lokalizować. W lipcu 1995 roku powstała pierwsza oficjalna wersja (opatrzona numerem 0.7) programu *GNU gettext*. Od tego momentu zaczęto przerabiać aplikacje tak, aby używając pakietu *gettext*, mogły być łatwo lokalizowane [GET01]. Niezależnie, od początku lat dziewięćdziesiątych XX wieku rozwijane było środowisko Java. Wraz z wersją JDK 1.1 wydaną w grudniu 1996, pojawiła się możliwość łatwej internacjonalizacji przez wykorzystanie klasy *java.util.ResourceBundle* [JAV02].

2.1. Gettext i pliki *po*

Z punktu widzenia programisty C można powiedzieć w skrócie, że użycie pakietu *gettext* sprowadza się do użycia funkcji o takiej samej nazwie zamiast literału (np. `gettext(„Hello world!”)` zamiast `„Hello world!”`). Przykładowy program wykorzystujący pakiet *gettext* może wyglądać następująco:

Listing 2.1: obsługa *gettext*

```
#include <libintl.h>
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    setlocale( LC_ALL, "" );
    bindtextdomain( "test", "/usr/share/locale" );
    textdomain( "test" );
    printf( gettext( "Hello, world!\\n" ) );
}
```

```
    exit(0);  
}
```

Należy zwrócić uwagę na wywołanie funkcji `bindtextdomain`, ustawiającej katalog bazowy, w którym poszukiwane będą pliki z komunikatami dla danej domeny (zbioru komunikatów) oraz `textdomain`, ustawiającej aktywną domenę.

Aby umożliwić pracę tłumaczowi, programista musi przygotować plik `pot` (Portable Object Template), w którym zapisywane są wszystkie używane w kodzie źródłowym komunikaty. Służy do tego program `xgettext`, którego wywołanie wygląda następująco:

```
xgettext -d domena -s -o test.pot test.c
```

Powyższa komenda utworzy plik `test.pot` ze wszystkimi komunikatami zawartymi w pliku źródłowym `test.c`. Przełącznik `-s` powoduje, że komunikaty w pliku `pot` posortowane są leksykograficznie, natomiast przełącznik `-d` określa domenę, w jakiej będzie wygenerowany plik `pot`. Podana domena musi być taka sama, jak domena określona przez wywołanie funkcji `textdomain` w pliku źródłowym.

Lokalizacja oprogramowania wykorzystującego omawiany pakiet zaczyna się od stworzenia pliku `po` (Portable Object). Można go utworzyć przez skopiowanie pliku `pot`, zaleca się jednak zastosowanie programu `msginit`, uaktualniającego wybrane wartości domyślne znajdujące się w szablonie. Wywołanie programu ma następującą postać:

```
msginit -l pl_PL -o polish.po -i test.pot
```

Powyższe polecenie utworzy odpowiedni plik (`polish.po`) na podstawie szablonu (`test.pot`). Przy wywołaniu programu konieczne jest podanie identyfikatora przygotowywanej wersji językowej (`ang. locale`).

Podczas pracy z plikami `po` należy przestrzegać ich formatu, który ma następującą postać:

Listing 2.2: plik `po`

```
BIAŁY-ZNAK  
# KOMENTARZE TŁUMACZA  
#. AUTOMATYCZNIE GENEROWANE KOMENTARZE  
#: ODNIESIENIA (GENEROWANE AUTOMATYCZNIE)  
#, FLAGI (GENEROWANE AUTOMATYCZNIE)  
msgid TEKST-ŹRÓDŁOWY  
msgstr TEKST-PRZETŁUMACZONY
```

Po przetłumaczeniu wszystkich komunikatów (`msgid`) i ich zapisaniu (`msgstr`) w pliku, należy przygotować jego binarną postać, umożliwiającą szybkie wczytanie i wyszukiwanie przez funkcję `gettext` użytą w programie. Plik ten, o rozszerzeniu `mo`, należy wygenerować przy pomocy programu `msgfmt`:

```
msgfmt -o test.mo polish.po
```

Należy zwrócić uwagę, aby skompilowany plik `mo` miał taką nazwę, jak nazwa domeny będąca parametrem wywołania funkcji `textdomain` w kodzie źródłowym aplikacji. Aby umożliwić użytkownikowi komunikację z programem w przygotowanym właśnie języku należy tak przygotowany plik umieścić w odpowiednim katalogu: `KATALOG/LL/LC_MESSAGES` lub `KATALOG/LL_CC/LC_MESSAGES`, gdzie `KATALOG` to drugi parametr wywołania funkcji `bindtextdomain`, `LL` to dwuliterowy skrót języka, a `CC` to skrót kraju; w naszym przypadku będzie to katalog `/usr/share/locale/pl_PL/LC_MESSAGES`.

2.2. *ResourceBundle* i pliki *properties*

Pliki *properties* składają się z linii, w których identyfikatorom przypisuje się literały:

Listing 2.3: plik *properties*

```
\# KOMENTARZ
hello = Hello World!
id1 = tekst
```

Przykładowa klasa wykorzystująca powyższy plik może wyglądać następująco:

Listing 2.4: obsługa plików *properties*

```
import java.util.ResourceBundle;

public class TestApp {

    private static ResourceBundle messages =
        ResourceBundle.getBundle("messages");

    public static void main(String[] args) throws Exception {
        System.out.println(messages.getString("hello"));
    }
}
```

Pracę z obiektem *ResourceBundle* należy zacząć od stworzenia obiektu tego typu przy użyciu metody statycznej *getBundle*. Wywołując wspomnianą metodę jako parametr, należy podać nazwę pliku *properties*, z którego będą odczytane komunikaty. Aby otrzymać literał zapisany w pliku pod tak zwanym kluczem, należy wywołać metodę *getString* na stworzonym wcześniej obiekcie.

Jak widać, praca z obiektami *ResourceBundle* znacznie różni się od pracy z pakietem *getText* – programista musi zaprojektować identyfikatory dla wszystkich komunikatów, tłumacz natomiast musi jednocześnie pracować z plikiem *properties*, w którym zapisane są oryginalne komunikaty oraz z nowym plikiem zawierającym tłumaczenia. Aby możliwe było uruchomienie programu w nowej wersji językowej, wystarczy dołączyć do programu nowy plik o nazwie *nazwabazowa_LL_CC.properties*, gdzie *nazwabazowa* to nazwa oryginalnego pliku, *LL* to identyfikator języka, a *CC* to identyfikator kraju, np: *messages_pl_PL.properties*

Z pakietem Java SDK (Software Development Kit) nie są dostarczane żadne narzędzia ułatwiające pracę z plikami *properties*. Co więcej, edycja tych plików jest utrudniona przez konieczność używania w nich kodowania unicode w notacji `'\uNNNN'`, gdzie *NNNN* to kod znaku w tym kodowaniu. Istnieją co prawda narzędzia umożliwiające edycję tych plików, ale należy je samodzielnie odnaleźć, pobrać z witryny i zainstalować w systemie. Przykładem swobodnego oprogramowania służącego do edycji plików *properties* jest *propedit*.

Rozdział 3

Tłumaczenie programów i dokumentacji

W rozdziale opisano praktyczne aspekty wykorzystania dostępnych narzędzi umożliwiających tłumaczenie tekstów informatycznych na przykładzie „Debian Documentation Project”, projektu tłumaczenia środowiska graficznego „KDE” oraz „Fedora Translation Project”.

3.1. Tłumaczenie tekstu przy użyciu programu CAT

Pod pojęciem „tekst informatyczny” kryje się cały szereg tekstów różnego rodzaju. Możemy mieć do czynienia z projektami technicznymi, specyfikacjami, dokumentacjami użytkownika czy administratora, opisami algorytmów, komentarzami kodów źródłowych, wreszcie z prasą specjalistyczną czy wydawnictwami technicznymi. Odrębną dziedzinę tekstów informatycznych stanowią komunikaty prezentowane użytkownikom przez programy komputerowe. W tym rozdziale zostaną zaprezentowane narzędzia ułatwiające i automatyzujące przekład wspomnianych tekstów, postaram się również zwrócić uwagę na specyficzne funkcje poszczególnych programów oraz ich przydatność w tłumaczeniu różnych typów tekstów.

Dla potrzeb niniejszego rozdziału, jeśli nie jest to określone inaczej, użycie sformułowania „program CAT” oznacza system wspomagający pracę tłumacza oparty na pamięciach tłumaczeniowych, tzn. system, w którym tłumacz przekłada segmenty tekstu źródłowego, umieszcza je w pamięci tłumaczeniowej i wykorzystuje do tłumaczenia kolejnych segmentów.

Schemat procesu tłumaczenia z użyciem narzędzi komputerowych jest niezależny od rodzaju tłumaczonego tekstu. Pełny proces tłumaczenia wymaga przygotowania dokumentu do tłumaczenia, wstępnego przetworzenia go przez program tłumaczeniowy oraz weryfikacji i tłumaczenia przez tłumacza. Poszczególne etapy prac są następujące:

1. Przygotowywanie wersji elektronicznej dokumentu w odpowiednim formacie.

Często zdarza się sytuacja, w której tłumaczony dokument nie jest dostępny w formie edytowalnego pliku tekstowego. Istnieje wtedy potrzeba konwersji dokumentu do formatu umożliwiającego późniejsze wykorzystanie narzędzi do komputerowego wspomaganie tłumaczeń. Czasami potrzebna jest jedynie konwersja między różnymi formatami plików do takiego, który obsługuje program CAT.

Zdarzają się jednak sytuacje, w których istnieje konieczność przekształcenia dokumentu na odpowiedni format z wykorzystaniem narzędzia OCR (np. ABBYY FineReader). Przed rozpoczęciem kolejnego etapu prac związanych z tłumaczeniem, konieczna jest

wtedy weryfikacja zgodności otrzymanego dokumentu z oryginałem, a w razie konieczności jego ręczna poprawa.

2. Przygotowywanie dokumentu do tłumaczenia.

Proces tłumaczenia tekstu nie sprowadza się jedynie do jego przełożenia z języka źródłowego na docelowy. Tłumaczenie dobrej jakości, obok treści, oddaje również wygląd dokumentu, poczynając od rodzaju i wielkości czcionek, przez pogrubienia i podkreślenia, po formatowanie tabeli i szereg innych właściwości. Idea systemów opartych na pamięciach tłumaczeniowych opiera się na odnajdywaniu segmentów tekstów przetłumaczonych wcześniej i wykorzystaniu ich w odpowiednich miejscach tłumaczonego tekstu. Aby sprostać wymaganiom z jednej strony jak najwyższego stopnia dokładności przy odnajdywaniu podobnych segmentów, z drugiej zaś zachowania wyglądu, przydatna jest ekstrakcja z dokumentu treści przeznaczonej do tłumaczenia. Przygotowany w ten sposób tekst tłumaczony jest przez specjalistę, a następnie przy pomocy zebranych wcześniej informacji o wyglądzie dokumentu jest on składany i poddawany edycji, mającej na celu dostosowanie wyglądu dokumentu do oryginału. Często systemy wspomagające tłumaczenie tekstów nie umożliwiają wykonania takiej ekstrakcji. Wtedy segmenty wyodrębnione z tekstu zawierają informację o formatowaniu.

Przed rozpoczęciem tłumaczenia warto również zweryfikować poprawność formatowania dostarczonych przez klienta plików i ujednoczyć formatowanie w całym dokumencie. Uwaga ta dotyczy szczególnie dokumentów otrzymanych w procesie rozpoznawania tekstu z dokumentów papierowych lub plików bitmapowych. Otrzymane w ten sposób dokumenty zawierają zazwyczaj niezależne formatowanie poszczególnych elementów, co utrudnia późniejsze dostosowanie wyglądu dokumentu. Dobrym pomysłem jest stosowanie stylów formatowania dostępnych w popularnych edytorach tekstów, takich jak „OpenOffice.org Writer” czy „Microsoft Office Word”. Należy również zauważyć, że tłumaczenie często przebiega iteracyjnie – kolejne wersje są sprawdzane i poprawiane. Poprawne przygotowanie dokumentu przed rozpoczęciem procesu tłumaczenia ogranicza do minimum późniejsze prace edycyjne i pozwala na automatyczne składanie tekstu po zmianach tłumaczeń odpowiednich segmentów w pamięci tłumaczeniowej.

Na etapie przygotowywania dokumentu do tłumaczenia, przeprowadza się czasami proces ekstrakcji terminologii. Operacja ta umożliwia tłumaczowi zapoznanie się terminologią używaną w dokumencie (lub całym projekcie składającym się z wielu dokumentów), przygotowanie specjalistycznych tłumaczeń i umieszczenie ich w słowniku. Wpływa to również na używanie jednorodnej terminologii przy tłumaczeniu, podnosząc tym samym jego jakość.

3. Podział tekstu na segmenty.

Tekst źródłowy poddawany jest podziałowi na mniejsze fragmenty zwane segmentami. Poprawnie przeprowadzona segmentacja tekstu umożliwia tłumaczenie jego fragmentów w oderwaniu od kontekstu. Dzięki temu, że segment stanowi spójny fragment tłumaczonego tekstu, możliwe jest wykorzystanie jego tłumaczenia we wszystkich miejscach, w których on (lub segment mu podobny) się znajduje. Bardzo często narzędzia umożliwiają automatyczną segmentację tekstu na podstawie zadanych parametrów - najczęściej stosuje się podział tekstu na zdania. Spotyka się również podział na paragrafy i linie. Należy zwrócić uwagę na fakt, że nie istnieje najlepszy sposób segmentacji tekstu; każdy tekst należy oddzielnie przeanalizować i wybrać odpowiednią w danym przypadku metodę. Przykładem może być tutaj instrukcja użytkownika, w której segmentami by-

łyby prawdopodobnie zdania, oraz log z programu wypisującego komunikaty wierszowo, gdzie najodpowiedniejszą metodą podziału na segmenty byłby podział na linie.

4. Tłumaczenie wstępne.

Po segmentacji możliwe jest przeprowadzane wstępne tłumaczenia tekstu na podstawie istniejących pamięci tłumaczeniowych. Podczas tego opcjonalnego kroku, system komputerowy wyszukuje kolejne segmenty tekstu źródłowego w TM, a odnalezione tłumaczenia podstawia w tłumaczonym dokumencie. Zazwyczaj po wykonaniu tłumaczenia wstępnego użytkownik jest informowany o statystykach związanych z trafnością poszczególnych tłumaczeń oraz ich ilości w stosunku do ogólnej liczby segmentów w tekście źródłowym. Taka informacja pozwala oszacować tłumaczowi czas potrzebny do przetłumaczenia dokumentu.

5. Weryfikacja tłumaczenia wstępnego i tłumaczenie pozostałych fragmentów tekstu.

Praca tłumacza zaczyna się po zakończeniu fazy wstępnego tłumaczenia wykonywanego przez system komputerowy. Tłumaczenie tekstu polega na tłumaczeniu kolejnych segmentów tekstu źródłowego. W przypadku, gdy program komputerowy zaproponował tłumaczenie na podstawie bazy pamięci tłumaczeniowej, użytkownik weryfikuje je i zatwierdza. Często zdarza się, że w bazie nie było identycznego tłumaczenia, a jedno lub więcej bardzo podobnych. W takim przypadku użytkownik ma możliwość wyboru najbardziej odpowiedniego tłumaczenia i jego zmianę. Może się również zdarzyć, szczególnie w przypadku, gdy pamięć tłumaczeniowa jest pusta, że segment nie został znaleziony w pamięci tłumaczeniowej. Wtedy należy wykonać pełne tłumaczenie fragmentu. Po zatwierdzeniu tłumaczenia segmentu przez użytkownika, jest ono dodawane do pamięci tłumaczeniowej tak, aby w przypadku wystąpienia podobnego segmentu w dalszej części tekstu, możliwe było odnalezienie go i podpowiedzenie tłumaczowi.

Wiele programów oferuje jeszcze jeden element wspomagający tłumacza w jego pracy - słownik zintegrowany z edytorem tłumaczenia. Podczas tłumaczenia segmentu, terminy w nim występujące wyszukiwane są w bazie słownika aplikacji, a w przypadku znalezienia odpowiedników w języku, na który tłumaczony jest tekst, zostają przedstawione użytkownikowi. Funkcja ta, w połączeniu z opisanym wcześniej procesem ekstrakcji terminologii z oryginalnego dokumentu, wpływa znacząco na jednorodne tłumaczenie terminów występujących w tekście.

6. Scalenie dokumentu i ostateczna weryfikacja

Po zakończeniu tłumaczenia, jeśli jest to konieczne, następuje scalenie dokumentu, konwersja do odpowiedniego formatu oraz jego ostateczna weryfikacja. W wyniku całego procesu, oprócz przetłumaczonego dokumentu, mamy uaktualnioną pamięć tłumaczeniową.

3.2. Projekt tłumaczeniowy dokumentacji Debiana

DDP (Debian Documentation Project) jest projektem, którego celem jest tworzenie i tłumaczenie dokumentacji systemu Debian. Najpopularniejsza jest obecnie linuksowa dystrybucja Debiana, trwają jednak prace nad innymi, niezależnymi adaptacjami, takimi jak Debian GNU/Hurd, który ma szansę stać się pierwszym systemem operacyjnym opartym całkowicie na oprogramowaniu GNU.

3.2.1. Projekt Międzynarodowy

Dokumentacja tworzona i utrzymywana obecnie w ramach projektu DDP to zgodnie z informacją dostępną na stronach projektu:

- podręczniki użytkownika
- podręczniki dewelopera
- różne podręczniki

Polityka dokumentacji DDP wskazuje jako format źródłowy dokumentacji SGML'owy podzbiór o nazwie debiandoc-sgml. Jako alternatywa brane były pod uwagę: DocBook, który został uznany za rozwiązanie zbyt duże, zbyt skomplikowane i trudne do skonfigurowania oraz nie rozwijany już linuxdoc-sgml. Przykładowy dokument w formacie debiandoc-sgml zaprezentowano w dodatku B.1.

Tłumaczenie podręczników odbywa się poprzez edycję plików źródłowych przy pomocy dostępnych edytorów tekstów – na liście dyskusyjnej debian-doc (<http://lists.debian.org/debian-doc/>) można znaleźć posty, mówiące o pracy z edytorami vim oraz emacs. Wspomniane edytory najczęściej uruchamiane są w trybie umożliwiającym podświetlanie składni SGML oraz sprawdzania pisowni, co ułatwia tłumaczowi odnalezienie fragmentów, które ma przetłumaczyć oraz ogranicza do minimum występowanie błędów orograficznych.

Trwają obecnie prace nad skryptem wydobywającym strukturę SGML z plików źródłowych podręczników. Skrypt ma pomóc utrzymać jednakową strukturę dokumentu niezależnie od wersji językowej [DDP1].

3.2.2. Polski projekt tłumaczenia dokumentacji Debiana

Początki Polskiego Projektu Dokumentacji Debiana (ang. Polish Debian Documentation Project) datować można na 17 marca 2003 roku, kiedy to Bartosz Feński przetłumaczył „Przewodnik po kompilacji jądra w Debianie”. Po przetłumaczeniu przewodnika i umieszczeniu informacji na listach dyskusyjnych, na jednej z nich (alt.pl.comp.os.linux.debian) wywiązała się ciekawa dyskusja, w wyniku której kilkanaście osób zaczęło współpracować w celu przetłumaczenia dokumentacji Debiana na język polski.

Oprócz podręczników w formacie debiandoc-sgml, PDDP zajmuje się tłumaczeniem stron WWW Debiana, szablonów DebConf, które wyświetlane są podczas instalacji i aktualizacji pakietów oraz DWN (ang. Debian Weekly News), czyli „Cotygodniowe Wiadomości ze świata Debiana”.

Strony WWW Debiana

Strony WWW Debiana generowane są z plików źródłowych w formacie WML. Treść prezentowana użytkownikowi końcowemu w postaci strony HTML generowana jest na podstawie zawartych w plikach źródłowych tekstów oraz skryptów sterujących.

Tłumaczenie pliku wml polega, podobnie jak w przypadku podręczników, na edycji pliku źródłowego i tłumaczeniu tekstu z pominięciem znaczników i instrukcji sterujących. Podobnie jak przy tłumaczeniu podręczników, tłumacze używają standardowych edytorów tekstów podświetlających składnię SGML.

Szablony DebConf

Szablony DebConf zapisane są w formacie gettext, który umożliwia bardzo łatwą lokalizację bez ingerencji w kod źródłowy tłumaczonej aplikacji. Szczegóły dotyczące formatu gettext oraz tłumaczenia takich plików zawarto w rozdziale dotyczącym dostosowywania oprogramowania do obsługi wielu języków (2.1).

DWN

Tłumaczenie „Cotygodniowych wiadomości ze świata Debiana” utrudnione jest z powodu zmieniającej się w czasie treści pojedynczego wydania - przed ostatecznym opublikowaniem jest ono przez cały tydzień aktualizowane. Choć samo tłumaczenie przebiega analogicznie jak w przypadku pozostałych typów dokumentów, wspomaganie tłumacza zostało w tym przypadku usprawnione przez udostępnienie skryptu umożliwiającego sprawdzenie, czy dokument źródłowy został zmieniony. W przypadku, gdy oryginalny dokument został uaktualniony, tłumacz otrzymuje informację o zmianach w formie pliku diff pokazującym wprowadzone zmiany. Narzędzie w postaci wspomnianego skryptu umożliwia sukcesywne tłumaczenie dokumentu, dzięki czemu polska wersja DWN jest gotowa do publikacji w niedługim czasie po opublikowaniu wersji oryginalnej.

3.3. Tłumaczenie środowiska graficznego KDE

Wiele pakietów oprogramowania organizuje lokalizację swojego oprogramowania niezależnie od dystrybucji Linuxa. Dobrym przykładem jest tutaj środowisko graficzne KDE, którego lokalizacja polega na tłumaczeniu plików *po*. Jest to możliwe, ponieważ pakiet został napisany z wykorzystaniem narzędzi gettext. Programem rekomendowanym do tłumaczenia KDE jest „KBabel”. Najważniejsze z punktu widzenia tłumacza cechy programu to:

- wsparcie dla plików *po* pakietu GNU gettext (łącznie z formami mnogimi)
- eksperymentalne wsparcie dla formatu XLIFF 1.0
- pełna funkcjonalność edycji plików dostępna zarówno przez dostosowywalny interfejs graficzny, jak i przez skróty klawiaturowe definiowane przez użytkownika
- zintegrowany z edytorem weryfikator poprawności ortograficznej
- możliwość generowania i prezentacji różnic między aktualnymi a wcześniejszymi wersjami komunikatów
- podświetlanie składni
- automatyczne uaktualnianie nagłówek
- system wtyczek dla słowników takich, jak pliki kompendium¹ (dostępnych np. pod adresem http://l10n.kde.org/po_overview/)
- funkcja wstępnego tłumaczenia sugerująca tłumaczenie komunikatów na podstawie pamięci tłumaczeniowej

¹w KBabel kompendium to plik zawierający wszystkie tłumaczone komunikaty. Zazwyczaj pliki te tworzone są przez konkatencję wszystkich plików *po* zawartych w projekcie.

- możliwość sprawdzenia spójności przetłumaczonych komunikatów, na przykład przez porównanie argumentów funkcji `printf` w `msgid` oraz `msgstr`
- wsparcie dla słowników opartych na TMX 1.4

KBabel to rozbudowane narzędzie umożliwiające kompleksowe zarządzanie plikami *po*. Interfejs użytkownika aplikacji umożliwia szybką i intuicyjną edycję oraz automatyzację tłumaczenia plików. Główne okno aplikacji umożliwia edycję, wyszukiwanie podobnych tłumaczeń oraz pokazywanie różnic w stosunku do poprzednich wersji komunikatów. „Menadżer Katalogów” umożliwia sprawdzanie poprawności plików, wyświetlanie informacji o statusie oraz statystykach mówiących o tym, które pliki zostały przetłumaczone w jakim stopniu. Do pakietu KBabel dołączony jest również program KBabelDict, umożliwiający dostęp do słowników przez rozbudowany system wtyczek.

3.4. Fedora Translation Project

Oprogramowanie zawarte w dystrybucji *Fedora Core* wykorzystuje ten sam pakiet (*gettext*), który używany jest w przypadku dystrybucji Debiana. Do tłumaczenia plików *po*, poleca się wykorzystanie opisywanego wcześniej programu *KBabel* oraz *gtranslator*. Główne funkcje programu *gtranslator*, to:

- łatwy w przystosowaniu mechanizm podświetlania składni definiowany w plikach XML,
- możliwość tworzenia plików *mo* bezpośrednio z interfejsu graficznego aplikacji,
- sprawdzanie pisowni oparte na *ispell*,
- wyszukiwanie we wcześniej przetłumaczonych komunikatach,
- obsługa sygnałów systemu operacyjnego (ang. signal) umożliwia utrzymanie spójności danych w plikach obsługiwanych przez program w przypadku awarii systemu,
- zaawansowany mechanizm wyszukiwania trafień częściowych.

Dokumentacja dystrybucji *Fedora Core* zgodna jest z DocBook. Dostęp do wersji źródłowych dokumentacji realizowany jest przez CVS. Nie wystarczy tutaj jednak konto anonimowe - konieczna jest rejestracja w serwisie *Fedora Account System*.

Tłumaczenie dokumentacji polega na wygenerowaniu plików *pot* na podstawie plików źródłowych dokumentacji, oraz przetłumaczeniu tych plików przy pomocy ulubionego narzędzia (rekomendowane są *KBabel* oraz *gtranslator*). Generowanie plików *pot* realizowane jest przy pomocy komendy:

```
make pot
```

W ten sposób przygotowane, a następnie przetłumaczone pliki *po* umieszcza się w repozytorium CVS projektu.

Rozdział 4

Narzędzia wspomagające tłumaczenie tekstów

W niniejszym rozdziale porównane zostały podstawowe właściwości wybranych programów wspomagających tłumaczenie różnego rodzaju tekstów. Na końcu rozdziału opisano rodzaje dostępnych słowników elektronicznych.

4.1. Swobodne oprogramowanie

Na rynku dostępnych jest szereg darmowych programów wspomagających tłumaczy w ich codziennej pracy. Jak pisałem w pierwszym rozdziale, programy pełnią różne funkcje i są użyteczne na różnych etapach procesu tłumaczenia.

Okazuje się, że cały proces tłumaczenia przedstawiony w rozdziale 3.1 "Tłumaczenie tekstu przy użyciu programu CAT" można zrealizować przy pomocy darmowych narzędzi:

4.1.1. Podział tekstu na segmenty

Jednym z programów, umożliwiających podział tekstu na segmenty, tłumaczenie wstępne, weryfikację, ręczne tłumaczenie pozostałych segmentów i scalenie ich w całość, jest program OmegaT. Program ten obsługuje następujące formaty dokumentów:

- HTML, XHTML
- OpenOffice.org / StarOffice

Bardziej szczegółowy opis programu znajduje się w pracy [GNI01].

Tłumaczone teksty, zwłaszcza informatyczne, przekształcane są na pliki *po*, które obsługiwane są przez kilka popularnych programów CAT. W przypadku dokumentacji Fedora Core, pliki te generowane są przy pomocy dostarczonego skryptu, uruchamianego z katalogu, w którym znajduje się dokumentacja.

Istnieje ogólnodostępny pakiet narzędzi umożliwiających generowanie plików *po* na podstawie dokumentów w różnych formatach, a następnie scalanie przetłumaczonego tekstu na podstawie oryginalnego dokumentu i tłumaczenia w pliku *po*. Pakiet nazywa się *po4a* i jest dostępny na stronie: <http://po4a.alioth.debian.org/>.

Podział tekstu na segmenty w przypadku pakietu *po4a* polega na stworzeniu pliku *po*. W przypadku tłumaczenia pierwszej wersji dokumentu, należy użyć następującego polecenia:

```
po4a-gettextize -f <format> -m <plik_do_tłumaczenia> -p <plik_po>
```

	poEdit	KBabel	gtranslator
Obsługa formatu PO	Tak	Tak	Tak
Obsługa formatu XLIFF	Nie	Tak	Nie
Funckje programów			
Obsługa TMX	Nie	Tak	Nie
Podświetlanie składni	Nie	Tak	Tak
Obsługa UTF-8	Tak	Tak	Tak
Sprawdzanie pisowni	Nie	Tak	Tak
Integracja ze słownikiem	Nie	Tak	Nie
Automatyczne tłumaczenie	Tak	Tak	Tak
Ocena interfejsu użytkownika	4	10	6

Tabela 4.1: Porównanie programów

gdzie:

- format to jeden z poniższych formatów plików, obsługiwanych przez pakiet *po4a*
 - nroff - format stron podręcznika ekranowego
 - pod - format dokumentacji perla
 - sgml - bardzo popularny format tworzenia dokumentacji;obecnie wspierane jest DTD *debian-doc-sgml*
 - xml - również popularny format tworzenia dokumentacji; obecnie wspierane jest DTD *docbook*
 - TeX/LaTeX - główny format dokumentacji używany w publikacjach pisanych przez ludzi związanych z wolnym oprogramowaniem
 - texinfo - format dokumentacji GNU
- plik_do_tłumaczenia to nazwa pliku, który będzie analizowany
- plik_po nazwa pliku *po*, który ma zostać wygenerowany

Jeśli tłumaczona ma być kolejna, zmieniona wersja dokumentu, można użyć polecenia:

```
po4a-updatepo -f <format> -m <plik_do_tlumaczenia> -p <plik_po>
```

gdzie: *plik_po*, to plik *po*, który ma zostać zaktualizowany, a pozostałe parametry, tak jak w przypadku *po4a-gettextize*

Tak przygotowane pliki należy przetłumaczyć, najlepiej wykorzystując dostępne programy CAT.

4.1.2. Tłumaczenie tekstu

Istnieje kilka programów wspomagających tłumaczenie plików *po*. Wcześniej opisane zostały programy KBabel, gtranslator. Dość popularny jest jeszcze program poedit.

Pliki *po* nie są obsługiwane przez programy pisane w języku Java. Ten język programowania ma swój mechanizm przechowywania komunikatów, który został opisany w rozdziale 2 - "Internacjonalizacja". Jak opisano w tym rozdziale, edycja tych plików w edytorach tekstu jest uciążliwa, a narzędziem ułatwiającym to zadanie jest program *propedit* dostępny na stronie http://propedit.sourceforge.jp/index_en.html.

4.1.3. Scalenie dokumentu i ostateczna weryfikacja

Na początku pracy z dokumentem, jego treść została oddzielona od formatowania przy pomocy programu `po4a-gettextize`. Teraz, po zakończonym procesie tłumaczenia, należy sformatować przetłumaczony tekst tak, aby wyglądał jak oryginał. Do tego celu służy program `po4a-translate`, który w miejsce oryginalnego tekstu w dokumencie źródłowym wstawia tłumaczenia z pliku `po`:

```
po4a-translate -f <format> -m <plik_do_tlumaczenia> \  
-p <plik_po> -l <plik_wyjściowy>
```

gdzie: *plik_po*, to plik *po* z przetłumaczonym tekstem, *plik_wyjściowy* to nazwa dokumentu generowanego przez program, a pozostałe parametry, tak jak w przypadku *po4a-gettextize*.

Tak przygotowany dokument, po weryfikacji, gotowy jest do opublikowania.

4.2. Słowniki

W poprzednim rozdziale omówione zostały sposoby przygotowywania oprogramowania do obsługi wielu języków oraz metody i narzędzia wspomagające automatyzację tłumaczenia tekstów.

Ten rozdział poświęcony został części komputerowego wspomagania tłumaczenia, umożliwiającej tłumaczowi szybkie i w miarę możliwości automatyczne dotarcie do tłumaczeń poszczególnych terminów. Z pomocą przychodzą tutaj słowniki, glosariusze, bazy terminologiczne oraz takie usługi dostępne w Internecie, jak grupy dyskusyjne i portale informacyjne.

Coraz częściej tłumacz sięga po elektroniczne wersje słowników, skracając czas potrzebny do odnalezienia tłumaczenia szukanego terminu. Internet stał się potężną bazą, umożliwiającą jednocześnie przeszukiwanie wielu słowników. Możliwa stała się również integracja dostępnych w Internecie zasobów słownikowych z oprogramowaniem CAT, co zwalnia użytkownika z potrzeby wpisywania szukanых słów do formularza wyszukiwarki.

4.2.1. Oprogramowanie i źródła dostępne w Internecie

Jeszcze kilka lat temu najpopularniejsze elektroniczne słowniki dystrybuowane były w postaci programów na płytach CD-ROM. Takie programy były elektronicznymi wersjami wydań papierowych i dzięki interfejsowi użytkownika umożliwiały łatwe wyszukiwanie terminów. Popularnym słownikiem był swego czasu "Słownik języka polskiego PWN". Przykłady innych słowników wydanych na płycie CD-ROM, to "Multimedialny słownik polsko-niemiecki" firmy LexLand s.c., czy "słowniki angielsko-polskie polsko-angielskie" firmy Leksykon.

Dziś bardzo dużo różnego rodzaju słowników dostępnych jest przez Internet. To medium ma jedną, bardzo ważną przewagę nad oprogramowaniem instalowanym na komputerze. Jest nią natychmiastowy dostęp użytkowników do najbardziej aktualnej wersji bazy terminologicznej. Umożliwia to zarówno wprowadzanie do słownika nowych terminów, jak i umożliwia natychmiastową rekację w przypadku wykrycia błędu w istniejących danych. Kilka słowników dostępnych przez Internet:

- Słownik wyrazów obcych i zwrotów obcojęzycznych Władysława Kopalińskiego (<http://www.slownik-online.pl/index.php>)
- Słownik języka polskiego PWN (<http://sjp.pwn.pl/>)

- Darmowy Słownik LING (www.ling.pl), zawierający słowniki angielski, niemiecki, francuski, włoski, hiszpański, rosyjski i polski słownik ortograficzny
- Słownik angielski (<http://www.dict.pl/>)
- Słownik niemiecki (<http://www.dep.pl/>)
- Słownik języka angielskiego (<http://dictionary.reference.com/>)
- Encyklopedia Matematyki (<http://mathworld.wolfram.com/>)

Omawiając słowniki dostępne w Internecie, nie sposób nie wspomnieć o rewolucyjnym pomysle stworzenia „wolnej encyklopedii”. Wikipedia (<http://www.wikipedia.org/>), bo tak nazywa się to przedsięwzięcie, tworzona jest przez wszystkich internautów, chcących dzielić się swoją wiedzą z innymi. Powstała ona w roku 2000 jako Nupedia, natomiast jej odsłona oparta o koncepcję wiki została uruchomiona 10 stycznia 2001. Dziś Wikipedia ma prawie 1 300 000 haseł. Polska wersja encyklopedii powstała ledwie dziewięć miesięcy później - 2 września 2001 roku i ma już ponad 270 000 haseł.

Sukces Wikipedii spowodował powstanie Wikisłownika (<http://www.wiktionary.org/>) - słownika wielojęzycznego z definicjami, etymologią, ortografią, formami fleksyjnymi, itp. dostępnego (podobnie, jak wikipedia) na zasadach Licencji Wolnej Dokumentacji GNU. Od czasu powstania polskiego wydania Wikisłownika 22 marca 2004, znajduje się tam już ponad 45 000 haseł.

4.2.2. Protokół DICT

Rosnąca liczba słowników dostępnych w Internecie umożliwia odnalezienie tłumaczeń coraz większej liczby terminów. Wraz z rosnącą liczbą słowników pogłębia się jednak problem odnalezienia tego, który będzie zawierał definicję lub tłumaczenie poszukiwanego słowa. Istnieje coraz większa potrzeba udostępniania pojedynczego interfejsu użytkownika udostępniającego wiele słowników równocześnie. Istnienie takiego interfejsu umożliwia również łatwiejsze dotarcie do odbiorcy, wystarczy bowiem przygotować odpowiednią bazę, nie martwiąc się o interfejs dla użytkownika końcowego.

Z pomocą przyszedł w 1997 roku DICT[RFC01] (The Dictionary Server Protocol), definiujący protokół pytań i odpowiedzi opartych na transmisji TCP, który umożliwia aplikacjom klienckim dostęp do definicji haseł z różnych słowników.

Protokół DICT oparty jest na stałym połączeniu, takim jak TCP, oraz zestawie pytań wysyłanych do serwera i odpowiedzi. Szczegóły dotyczące protokołu dostępne są w dokumencie [RFC01].

Rozdział 5

Kreator Słowników

W ramach niniejszej pracy stworzony został program umożliwiający tworzenie i edycję słownika dwujęzycznego. Słownik tworzony jest na podstawie pamięci tłumaczeniowej będącej wynikiem tłumaczenia dowolnego tekstu. Tworzenie słownika polega na kojarzeniu w pary odpowiadających sobie słów z segmentu źródłowego i docelowego.

Niniejszy rozdział zawiera opis techniczny programu. Przedstawiono jego podział na poszczególne moduły, zaprezentowano główne interfejsy i struktury danych oraz opisano sposoby rozszerzania programu o obsługę nowych formatów danych i inne dostępne funkcje.

Program „Kreator słowników” został napisany w języku Java, przy wykorzystaniu J2SE¹. Jediną biblioteką spoza J2SE używaną w programie jest *jdom*². Do napisania programu użyto środowiska programistycznego Eclipse w wersji 3.0³. Sam program dostępny jest na zasadach licencji GNU GPL, natomiast jego dokumentacja na zasadach licencji GNU FDL, których treści znaleźć można pod adresem: <http://www.gnu.org/licenses/>.

5.1. Podział na moduły

Naturalnym mechanizmem do wydzielenia w programie modułów funkcjonalnych było wykorzystanie pakietów dostępnych w języku Java. Główna klasa programu, klasa `Main`, znajduje się w pakiecie `edu.mimuw.nmpt.msw.creator`, który jednocześnie jest pakietem nadrzędnym dla wszystkich innych pakietów programu. Oprócz pakietu głównego (`edu.mimuw.nmpt.msw.creator`), w programie należy wyróżnić następujące pakiety:

`edu.mimuw.nmpt.msw.creator.dict` - zawiera klasy związane z obsługą słowników. Pakiet podzielony został na następujące moduły funkcjonalne:

- **`fileHandler`** - w tym pakiecie znajdują się klasy odpowiedzialne za obsługę (odczyt/zapis) słowników w różnych formatach. Obecnie obsługiwane formaty to:

¹Java 2 Platform, Standard Edition (J2SE) jest kolekcją API języka Java przydatnych w aplikacjach pisanych na stacje robocze, w odróżnieniu od Java 2 Platform, Enterprise Edition (J2EE) zawierającej dodatkowo interfejsy przydatne programom uruchamianym na serwerach

²JDOM jest biblioteką umożliwiającą łatwą manipulację na XML w języku Java. Biblioteka dostępna jest pod adresem <http://www.jdom.org/> na zasadach opartych na licencji Apache z wyłączeniem klauzuli mówiącej o zachowaniu informacji o autorze - jest to jedna z najmniej restrykcyjnych licencji

³Eclipse jest rozbudowanym środowiskiem IDE (Integrated Development Environment). Oprogramowanie dostępne jest na zasadach licencji EPL (Eclipse Public License) dostępnej pod adresem: <http://www.eclipse.org/org/documents/epl-v10.php>

- **tei** - format FreeDict⁴, którego podstawowa obsługa została zaimplementowana w klasie `FreeDictHandler`,
- **properties** - format plików używany w języku Java do przechowywania struktur odwzorowujących klucz na wartość. Format został wykorzystany do przechowywania słowników powstających w trakcie pracy z programem „Kreator Słowników” i jest obsługiwany przez klasę `PropertiesHandler`
- **ma** - obsługa analizatora morfologicznego. W pakiecie znajduje się interfejs `IMorphologicalAnalyzer` odpowiedzialny za określanie dostępnych form bazowych danego słowa. Obecnie zaimplementowany jest analizator języka polskiego zaimplementowany w klasie `PolishMA`. Analizator oparty jest o program *Morfeusz*.
- **impl** - implementacja interfejsów zdefiniowanych w pakiecie nadrzędnym:
 - `IDictionary`
 - `IForm`
 - `ITranslation`
 - `IWord`

edu.mimuw.nmpt.msw.creator.tm - zawiera klasy związane z obsługą pamięci tłumaczeniowych. Podobnie jak pakiet `dict` podzielony jest na następujące moduły funkcjonalne:

- **fileHandler** - moduł odpowiedzialny za obsługę plików pamięci tłumaczeniowej. Zaimplementowane obecnie w „Kreatorze Słowników” formaty plików TM to:
 - **TMX** - obsługa formatu TMX zaimplementowana została w klasie `TMXHandler`,
 - **po** - obsługa popularnego formatu przechowującego tłumaczenia komunikatów zaimplementowana w klasie `POHandler`
- **tokenizer** - pakiet odpowiedzialny za podział segmentów znajdujących się w pamięci tłumaczeniowej na listę słów, które służą następnie do budowania słownika. Oprócz domyślnego podziału segmentu na słowa (gdzie słowo jest ciągiem znaków niezawierającym znaku specjalnego) zaimplementowanego w klasie `DefaultSegmentTokenizer`, opracowano podział, w którym na podstawie konfiguracji filtrowane są wybrane słowa. Podział taki zastosowano dla języka angielskiego, gdzie filtrowane są słowa „a” oraz „the”, a implementacja znajduje się w klasie `FilteredSegmentTokenizer`.

edu.mimuw.nmpt.msw.creator.ui klasy związane z obsługą graficznego interfejsu użytkownika, który został zaimplementowany przy pomocy pakietów JFC⁵ Swing. Pakiet zawiera następujące moduły podzielone zgodnie ze wzorcem MVC⁶ na trzy moduły funkcjonalne:

- **model** - zawierający model danych używany przez system interfejsu graficznego. Model obejmuje struktury danych pozwalające na operowanie na:

⁴FreeDict jest XML’owym formatem danych umożliwiającym przechowywanie słowników dwujęzycznych

⁵Java Foundation Classes (JFC) są zbiorem bibliotek klas języka Java, będących częścią (J2SE). Biblioteki JFC wspierają budowę graficznych interfejsów użytkownika (GUI) w aplikacjach uruchamianych na popularnych platformach, takich jak Microsoft Windows, Linux, czy Mac OS

⁶Model-view-controller (MVC) (ang. model-widok-kontroler) - wzorec projektowy obejmujący architekturę aplikacji, rozdzielający model danych, interfejs użytkownika oraz część aplikacji odpowiedzialną za jej logikę w trzy niezależne komponenty tak, aby zmiany wprowadzane w jednym z nich miały minimalny wpływ na pozostałe

- *pamięci tłumaczeniowej*,
 - *segmentach* pamięci tłumaczeniowej,
 - *wariantach segmentów* pamięci tłumaczeniowej, tj. segmentach w poszczególnych wersjach językowych, oraz
 - *wyrazach*,
 - i ich *tłumaczeniach*
- **control** - zapewniający prezentację treści użytkownikowi. W tym pakiecie znajdują się implementacje kontrolek zintegrowanych między sobą tak, aby możliwe było stworzenie spójnego i intuicyjnego interfejsu użytkownika,
 - **action** - zawierający implementację akcji wykonywanych przez użytkownika w aplikacji.

Ponadto dostępne są pakiety:

edu.mimuw.nmpt.msw.creator.exception zawierający klasy wyjątków używanych w programie, oraz

edu.mimuw.nmpt.msw.creator.util zawierający narzędzia, których nie można przyporządkować do żadnego z powyższych pakietów.

5.2. Stosowane wzorce i konwencje

Program „Kreator Słowników” został napisany zgodnie z najlepszą wiedzą autora o wzorcach projektowych oraz konwencjach stosowanych przy programowaniu obiektowym, a w szczególności związanych z pisaniem oprogramowania w języku Java. W tej części dokumentu zostały przedstawione konstrukcje i mechanizmy najbardziej istotne dla implementacji projektu.

5.2.1. Logowanie zdarzeń

Od wersji 1.4 środowiska programistycznego Java, dostępny jest pakiet umożliwiający logowanie zdarzeń zachodzących w programie. Odpowiedzialny za to zadanie jest pakiet `java.util.logging` udostępniający interfejs klasy zapisującej logi oraz podstawowe implementacje, z których najważniejsze to:

- `java.util.logging.FileHandler` umożliwiający logowanie do pliku
- `java.util.logging.ConsoleHandler` logujący na standardowe wyjście błędów (`System.err`)

Pakiet umożliwia takie przygotowanie oprogramowania, aby możliwe było wypisywanie zarówno błędów związanych z jego działaniem, jak również wszystkich komunikatów służących szczegółowej diagnostyce programu. Wspomniany mechanizm oparty jest o tzw. poziom ważności komunikatu. Podczas pracy z programem możliwe jest zdefiniowanie w pliku konfiguracyjnym poziomu granicznego, powyżej którego komunikaty będą przetwarzane.

Każda klasa wypisująca komunikat posiada deklarację obiektu `Logger` oraz linie wypisujące komunikaty przy pomocy zdefiniowanego obiektu. Przykład zastosowania omawianego mechanizmu zilustrowany został na listingu 5.1

Listing 5.1: Użycie mechanizmu logowania

```
import java.util.logging.Logger;
import java.util.logging.Level;

public class NazwaKlasy implements NazwaInterfejsu
{
    private static final Logger logger = Logger.getLogger(NazwaKlasy.class.getName());
    ...

    public void metoda() {
        ...
        logger.log(Level.FINE, "komunikat zapisywany do logu");
        ...
    }

    ...
}
```

5.2.2. ShutdownHook

Środowisko JRE⁷ zawiera mechanizm umożliwiający wykonanie pewnych czynności w momencie zamknięcia programu. Bardziej szczegółowo, w momencie sygnału mówiącego o konieczności zamknięcia programu (wywołanego przez samego użytkownika lub z przyczyn od niego niezależnych związanych np. z wystąpieniem błędu krytycznego) uruchamiane są wątki zarejestrowane wcześniej na taką okoliczność. Program „Kreator Słowników” wykorzystuje tę możliwość do zapisania słowników oraz stanu aplikacji w momencie jej zamknięcia. Poniżej znajduje się listing (5.2) pokazujący fragment klasy Main rejestrującej odpowiedni komponent:

Listing 5.2: Main.java

```
package edu.mimuw.nmpt.msw.creator;

import java.util.logging.Level;
import java.util.logging.Logger;

...

public class Main {

    private static final Logger logger = Logger.getLogger(Main.class.getName());
    ...

    public static void main(String[] args) {
        ...
        // wątek uruchamiany w momencie zamykania wirtualnej maszyny Java.
        Runtime.getRuntime().addShutdownHook(new ShutdownHook());
        ...
    }

    ...

    static class ShutdownHook extends Thread {
        ...
        ShutdownHook() {
```

⁷Java Runtime Environment (JRE) - środowisko uruchomieniowe dla programów napisanych w języku Java. Zawiera JVM (wirtualną maszynę Java) oraz zestaw standardowych bibliotek

```

    ... // inicjalizacja obiektu
}

public void run() {
    try {
        ... //wykonanie czynności w momencie zamknięcia programu
    } catch (Exception ex) {
        logger.log(Level.SEVERE, "error shutting down", ex);
    }
}
}
...
}

```

5.2.3. Wzorzec *Singleton*

Często stosowanym w programowaniu obiektowym wzorcem projektowym jest wzorzec o nazwie *Singleton*. Jest on stosowany tam, gdzie programista chce mieć pewność, że w całym programie będzie tylko jeden obiekt danej klasy/interfejsu.

W programie „Kreator Słowników” wzorzec stosowany jest w kilku miejscach. Przykładem może być klasa `ProgramRuntime` służąca jako kontroler aktualnie wczytanej pamięci tłumaczeniowej. Charakterystyczne cechy klasy napisanej zgodnie z omawianym wzorcem to prywatne stała przechowująca obiekt, prywatny konstruktor oraz statyczna metoda zwracająca obiekt. Wszystkie trzy elementy zaprezentowano na listingu 5.3.

Listing 5.3: `ProgramRuntime.java`

```

...
public class ProgramRuntime {

    private static final ProgramRuntime RUNTIME = new ProgramRuntime();
    ...

    private ProgramRuntime() {
        super();
    }

    public static ProgramRuntime getRuntime() {
        return ProgramRuntime.RUNTIME;
    }

    ...
}

```

5.2.4. Internacjonalizacja

Temat pracy: „Narzędzia wspomagające tłumaczenie tekstów informatycznych” nie tylko stał się inspiracją do napisania programu, ale wpłynął również na założenia dotyczące jego konstrukcji. Podczas pisania programu zwrócono szczególną uwagę na możliwość jego późniejszej lokalizacji. Do przygotowania programu do obsługi interfejsu użytkownika w wielu językach użyto omawianego w rozdziale 2 mechanizmu plików *properties*. Zaprojektowano plik *captions.properties* zawierający komunikaty związane z interfejsem użytkownika, w którym zapisano ich domyślną, angielską wersję. Wraz z programem dystrybuowana jest polska wersja komunikatów zawarta w pliku *captions-pl.properties*.

Program uruchamia się w wersji językowej zgodnej z ustawieniami w systemie operacyjnym. W przypadku, kiedy dana wersja językowa nie jest przygotowana, używana jest domyślna, angielska wersja. Bardzo łatwe jest przygotowanie kolejnych wersji językowych programu. Wystarczy przygotować plik *captions_LN.properties*, gdzie *LN* jest skrótem języka, i umieścić go w podkatalogu *locale* dystrybucji.

Wyizolowanie literałów

W programie wszystkie literały z wyjątkiem opisów błędów zostały wyizolowane jako stałe. Wszystkie klucze identyfikujące komunikaty zostały umieszczone w pliku *Captions.java*, dzięki czemu łatwiejsze staje się zarządzanie komunikatami używanymi w programie. Klasa ta stanowi również interfejs do pobierania komunikatów w odpowiedniej wersji językowej:

Listing 5.4: Captions.java

```
package edu.mimuw.nmpt.msw.creator.ui;

import java.util.MissingResourceException;
import java.util.ResourceBundle;

public class Captions {

    private static ResourceBundle captions = ResourceBundle.getBundle( "captions" );

    public static final String CAP_NotAWordFormMessage = "NotAWordFormMessage";
    ...
    public static final String CAP_TranslationsViewerTranslationDeleted = "
        TranslationsViewerTranslationDeleted";

    public static String get( String key ) {
        try {
            return captions.getString( key );
        } catch ( MissingResourceException ex ) {
            return key;
        }
    }
}
```

Listing 5.5: captions.properties

```
NotAWordFormMessage = {0} is not a valid form of '{1}'
...
TranslationsViewerTranslationDeleted = Translation deleted
```

Listing 5.6: użycie Captions - TranslationsViewer.java

```
...
String statusMsg = Captions.get( Captions.CAP_TranslationsViewerTranslationDeleted );
MainWindow.getWindow().getStatusBar().setStatus( statusMsg, 2000 );
...
```

5.3. Interfejs użytkownika

Klasa główna programu to klasa *Main*. Jest ona odpowiedzialna za inicjalizację programu oraz przekazanie sterowania do interfejsu użytkownika. Podczas inicjalizacji wczytywane są

wszystkie słowniki, które były otwarte przy ostatniej sesji użytkownika z programem oraz wszystkie słowniki znajdujące się w podkatalogu *dictionaries*. W tym miejscu tworzone jest również główne okno programu: `MainWindow`.

Główne okno programu składa się z szeregu powiązanych ze sobą komponentów prezentujących użytkownikowi poszczególne elementy interfejsu użytkownika oraz reagujących na jego akcje. Klasa dziedziczy po klasie `JFrame` należącej do pakietu *Swing*, a jego główna metoda, to metoda budująca zawartość okna: *createContents*. Poniżej opisane zostały krótko poszczególne komponenty interfejsu użytkownika. Szczegółów dotyczących implementacji komponentów należy szukać w kodzie źródłowym aplikacji dołączonym do niniejszej pracy na płycie CD-ROM.

5.3.1. `SourceTuvViewer`

Komponent prezentujący listę wyrazów segmentu źródłowego zaimplementowany jest w klasie `SourceTuvViewer`. Komponent reaguje na wybranie z listy słowa. W przypadku wybrania nowego słowa, wysyłane są komunikaty do komponentu prezentującego listę słów segmentu docelowego oraz do komponentu prezentującego tłumaczenia wybranego słowa.

5.3.2. `TargetTuvViewer`

Jest to komponent prezentujący listę słów segmentu docelowego. Po otrzymaniu komunikatu o wyborze nowego słowa na liście słów segmentu źródłowego, komponent odznacza wszystkie wybrane słowa na swojej liście, oraz zaznacza słowa znajdujące się w słownikach, dzięki czemu użytkownik jest informowany o aktualnych powiązaniach między wybranym słowem źródłowym a tłumaczeniami znajdującymi się na liście segmentu docelowego.

Wybór słowa na liście prezentowanej przez omawiany komponent skutkuje dodaniem tłumaczenia do słownika oraz powiadomienie komponentu `TranslationViewer` o konieczności odświeżenia listy tłumaczeń dla słowa źródłowego. Odznaczenie słowa przez użytkownika wywołuje akcję usunięcia tego tłumaczenia ze słownika.

5.3.3. `SegmentViewer`

Komponent umieszczony w lewym dolnym rogu aplikacji jest `SegmentViewer` odpowiedzialny za prezentację i nawigację między segmentami pamięci tłumaczeniowej. Składa się on z dwóch pól tekstowych prezentujących aktualne warianty segmentów (w języku źródłowym i docelowym) wraz z ich kontekstami. Komponent zawiera również przyciski *Następny* i *Poprzedni* służące do nawigacji po pamięci tłumaczeniowej.

5.3.4. `TranslationViewer`

Po prawej stronie okna aplikacji znajduje się komponent obsługujący edycję słowników. `TranslationViewer` składa się z dwóch zakładek odpowiedzialnych za usuwanie i dodawanie słów do słownika. Na każdej z zakładek znajduje się lista słowników, na których można wykonać operacje, oraz przycisk służący do ich potwierdzenia.

5.3.5. `StatusBar`

Na dole okna programu znajduje się pasek stanu, na którym wyświetlane są komunikaty związane z bieżącą obsługą programu: komunikaty potwierdzające wykonanie operacji, błędy programu oraz błędy użytkownika. Komponent obsługiwany jest przez metody *setStatus* umoż-

liwiający wyświetlenie komunikatu przez określony czas. Do usuwania komunikatów z opóźnieniem wykorzystano mechanizm dostarczany z J2SE oparty na klasie `java.util.Timer`.

Listing 5.7: StatusBar.java

```
...
import java.util.Timer;

public class StatusBar extends JPanel {

    private final Timer timer = new Timer();
    ...

    public void setStatus( String msg, Color color, long duration ) {
        StringBuffer buffer = new StringBuffer( "<font color='#" );

        buffer.append( toHex( color.getRed() ) );
        buffer.append( toHex( color.getGreen() ) );
        buffer.append( toHex( color.getBlue() ) );
        buffer.append( ">" ).append( msg ).append( "</font>" );
        String formattedMessage = buffer.toString();
        addMessage( formattedMessage );
        timer.schedule( new RemoveMessageTask( formattedMessage ), duration );
    }

    ...

    private class RemoveMessageTask extends TimerTask {
        ...
    }
}
```

5.4. Możliwości rozbudowy programu

Program „Kreator Słowników” umożliwia zaprezentowanie idei tworzenia słowników w oparciu o pamięci tłumaczeniowe w ograniczonym zakresie. Spowodowane jest to implementacją tylko podstawowych formatów plików oraz wybranych, przykładowych implementacji interfejsów programistycznych dostępnych w programie. Poniżej znajduje się opis możliwości rozbudowy programu o wymagane przez użytkownika funkcje.

Każda nowa implementacja opisanych funkcji wymaga konfiguracji. Plik konfiguracyjny ma nazwę `runtime.properties` i znajduje się w podkatalogu `etc` dystrybucji programu. Poniżej przedstawiono jego zawartość (wraz z numerami linii, których nie ma w oryginalnym pliku), do której, w kolejnych sekcjach, znaleźć można odwołania. Należy zaznaczyć, że w przypadku traktowania wartości poszczególnych jako list, program traktuje jako separator każdy z następujących znaków: „ ” - pusty znak (spacja), „:” - dwukropek, „;” - średnik oraz „,” - przecinek.

Listing 5.8: etc/runtime.properties

```
1 dict.extensions      = properties, tei
2 dict.handler.properties = edu.mimuw.nmpt.msw.creator.dict.fileHandler.PropertiesHandler
3 dict.handler.tei     = edu.mimuw.nmpt.msw.creator.dict.fileHandler.FreeDictHandler
4
5 tm.extensions       = dummy, xml, tmx, po
6 tm.handler.dummy   = edu.mimuw.nmpt.msw.creator.tm.fileHandler.DummyHandler
```

```

7 tm.handler.xml = edu.mimuw.nmpt.msw.creator.tm.fileHandler.impl.TMXHandler
8 tm.handler.tmx = edu.mimuw.nmpt.msw.creator.tm.fileHandler.impl.TMXHandler
9 tm.handler.po = edu.mimuw.nmpt.msw.creator.tm.fileHandler.impl.POHandler
10
11 #morphological analyser.pl = edu.mimuw.nmpt.msw.creator.dict.ma.PolishMA
12 #morphological analyser.pl.programName = Sam34.exe
13 #morphological analyser.pl.programOptions = -b
14 #morphological analyser.pl.programInput = dane.sam
15 #morphological analyser.pl.programOutput = wyniki.sam
16 #morphological analyser.pl.programWorkingDir = sam
17 #morphological analyser.pl.streamEncoding = CP852
18 morphological.analyser.pl = edu.mimuw.nmpt.msw.creator.dict.ma.MorfeuszMA
19
20 segment.tokenizer.en = edu.mimuw.nmpt.msw.creator.tm.tokenizer.FilteredSegmentTokenizer
21 segment.tokenizer.en.filter = a, the

```

5.4.1. Obsługa plików słownika

Słowniki w prezentowanym programie reprezentowane są przez interfejs `IDictionary` zawarty w pakiecie `edu.mimuw.nmpt.msw.creator.dict`. Dostępna jest również przykładowa implementacja słownika - `impl.Dictionary`.

Obsługa różnych formatów plików przechowujących słowniki oparta jest na rozszerzeniach tych plików. W momencie żądania użytkownika o otworzenie nowego słownika wywoływana jest metoda `getDictionary` obiektu `fileHandler.FileHandlerManager`. Metoda ta pobiera klasy, które potencjalnie mogą obsłużyć plik z danym rozszerzeniem (metoda `getHandlers`, wywołując następnie metodę `canHandle` każdej z nich. Dzięki temu możliwa jest obsługa różnych formatów plików, które posiadają takie samo rozszerzenie np. XML). W momencie trafienia na obiekt potrafiący wczytać zadany plik ze słownikiem, wywoływana jest na nim metoda `load` zwracająca obiekt rozszerzający znany już nam interfejs `IDictionary`. Cały proces wczytywania pliku zaprezentowany jest na listingu 5.9

Listing 5.9: `FileHandlerManager.java`

```

...
public class FileHandlerManager {

    ...
    public IDictionary getDictionary( String pathToFile ) {
        try {
            File file = new File( pathToFile );
            Collection handlers;
            try {
                handlers = getHandlers( file );
            } catch ( IOException ex ) {
                throw new CanNotReadFromFile( pathToFile );
            }
            for ( Iterator iter = handlers.iterator(); iter.hasNext(); ) {
                IFileHandler handler = (IFileHandler) iter.next();
                if ( handler.canHandle( file ) )
                    return handler.load( file );
            }

            throw new NoHandlerFound( pathToFile );
        } catch ( Exception ex ) {
            logger.info( "no handler found for " + pathToFile );
            return null;
        }
    }
}

```

```

}
...
}

```

Klasy potrafiące wczytywać i zapisywać słowniki w poszczególnych formatach implementują interfejs `fileHandler.IFileHandler` przedstawiony na listunku 5.10.

Listing 5.10: `IFileHandler.java`

```

package edu.mimuw.nmpt.msw.creator.dict.fileHandler;

import java.io.File;
import java.io.IOException;

import edu.mimuw.nmpt.msw.creator.dict.IDictionary;
import edu.mimuw.nmpt.msw.creator.exception.DictionaryParseException;

/**
 * Interfejs klasy odczytującej pliki ze słownikami.
 * Poszczególne implementacje będą odczytywać konkretne formaty słowników.
 */
public interface IFileHandler {

    /**
     * Metoda sprawdza czy ta klasa umie odczytać słownik zapisany
     * w pliku. Test realizowany jest przez sprawdzenie
     * rozszerzenia pliku i/lub jego zawartości.
     *
     * @param file plik ze słownikiem do odczytania
     * @return informacja czy ta klasa umie przeczytać słownik z pliku
     */
    boolean canHandle( File file );

    /**
     * Odczytuje słownik z pliku. Przed wołaniem tej metody należy
     * najpierw sprawdzić czy ta klasa umie odczytać słownik
     * zapisany w pliku (metoda canHandle()).
     *
     * @param file plik ze słownikiem do odczytania
     * @return odczytany słownik
     */
    IDictionary load( File file ) throws DictionaryParseException, IOException ;

    /**
     * Zapisuje słownik do pliku, z którego został on odczytany.
     *
     * @param dictionary słownik do zapisania.
     * @return plik z zapisanym słownikiem.
     */
    File save( IDictionary dictionary ) throws IOException;

    /**
     * Zapisuje słownik do podanego pliku.
     *
     * @param dictionary słownik do zapisania.
     * @param file plik, do którego ma być zapisany słownik
     * @return plik z zapisanym słownikiem.
     */
    File saveAs( IDictionary dictionary, File file ) throws IOException;

```

```

/**
 * Zwraca opis formatu pliku obsługiwanego przez tą klasę.
 *
 * @return opis formatu pliku ze słownikiem
 */
String getDescription();
}

```

Konfiguracja

Aby dodać do programu obsługę kolejnego formatu pliku przechowującego słownik należy zaimplementować interfejs `IFileHandler` oraz skonfigurować go tak, aby mógł być użyty przez program.

Założmy, że chcemy dodać obsługę nowej wersji formatu `FreeDict`, którego rozszerzenie jest zgodne z wersją poprzednią i ma postać „*tei*”.

Do pliku konfiguracyjnego z listingu 5.8 należy dodać klasę obsługującą plik z danym rozszerzeniem.

1. Najpierw należy sprawdzić, czy rozszerzenie jest obsługiwane przez program. Obsługiwane rozszerzenia wymienione są w linii pierwszej jako wartość klucza `dict.extensions`. Jeśli rozszerzenie nie znajduje się na liście należy je dodać. W naszym przypadku na liście znajduje się rozszerzenie *tei*, więc nie modyfikujemy tej linii.
2. Następnie należy zlokalizować linię z listą klas obsługujących dane rozszerzenie. Klucz w pliku `properties` ma postać `dict.handler.EXT`, gdzie `EXT` jest naszym rozszerzeniem. W przypadku rozszerzenia *tei* szukaną linią jest linia nr. 3. Jeśli w pliku konfiguracyjnym nie ma odpowiedniej linii, należy ją dodać.
3. Ostatnim krokiem związanym z konfiguracją klasy obsługującej nowy format słownika jest dopisanie klasy na zlokalizowanej wcześniej liście. Wpisana klasa musi mieć pełną nazwę wraz z pakietem, w którym została umieszczona. Jeśli różne formaty plików mają takie samo rozszerzenie, klasy obsługujące te formaty należy wymienić z użyciem separatorów opisanych w sekcji 5.4

5.4.2. Obsługa plików pamięci tłumaczeniowej

Obsługa plików pamięci tłumaczeniowych jest analogiczna do obsługi plików zawierających słowniki. Jediną różnicą jest użycie klas z pakietu `edu.mimuw.nmpt.msw.creator.tm`: strukturą danych przechowującą pamięć tłumaczeniową jest klasa `TM`, natomiast interfejs `IFileHandler` zawiera tylko metody umożliwiające odczytanie `TM` z pliku:

Listing 5.11: `IFileHandler.java`

```

package edu.mimuw.nmpt.msw.creator.tm.fileHandler;

import java.io.File;

import edu.mimuw.nmpt.msw.creator.tm.TM;

/**
 * Interfejs klasy odczytującej pliki ze słownikami.
 * Poszczególne implementacje będą odczytywać konkretne formaty słowników.
 *
 */

```

```

* @author marcin
*/
public interface IFileHandler {

    /**
     * Metoda sprawdza czy ta klasa umie odczytać słownik zapisany
     * w podanym pliku. Test realizowany jest przez sprawdzenie
     * rozszerzenia pliku i/lub jego zawartości.
     *
     * @param file plik z pamięci tłumaczeniowa do odczytania
     * @return informacja czy ta klasa umie odczytac podany plik
     */
    boolean canHandle( File file );

    /**
     * Odczytuje TM z pliku. Przed wołaniem tej metody należy najpierw
     * sprawdzić czy ta klasa umie odczytać TM zapisany w pliku
     * (metoda canHandle()).
     *
     * @param file plik ze słownikiem do odczytania
     * @return odczytany słownik
     */
    TM load( File file );
}

```

Konfiguracja

Dodawanie obsługi nowych formatów plików pamięci tłumaczeniowych jest analogiczna jak w przypadku formatów plików ze słownikami. Aby dodać obsługę nowego formatu należy zaimplementować interfejs `IFileHandler` oraz rozszerzyć konfigurację programu.

Jedyną różnicą w konfiguracji klas obsługujących pamięci tłumaczeniowe w stosunku do klas obsługujących słowniki są identyfikatory kluczy w pliku konfiguracyjnym: lista obsługiwanych formatów zapisana jest pod kluczem `tm.extensions`, natomiast lista klas obsługujących dane rozszerzenie ma postać `tm.handler.EXT`, gdzie `EXT` jest obsługiwany rozszerzeniem pliku.

Na przykładzie pamięci tłumaczeniowych można pokazać, że ta sama klasa może służyć do obsługi formatu pliku z różnymi rozszerzeniami tak, jak ma to miejsce w przypadku rozszerzeń `xml` oraz `tei` skonfigurowanymi w pliku konfiguracyjnym w liniach odpowiednio 7 i 8.

5.4.3. Dzielenie segmentów na wyrazy

Podczas testowania wstępnej wersji programu przydatne okazało się stworzenie osobnego interfejsu umożliwiającego opisującego proces dzielenia segmentu na wyrazy. Obok standardowego podziału (na kolejne wyrazy w segmencie) przydatne było zaimplementowanie filtra, odrzucającego wybrane słowa. Filtr ten zastosowano w przypadku języka angielskiego, a odrzucane są wyrazy „a” oraz „the”. W efekcie przeprowadzonych prac wyodrębniony został interfejs `ISegmentTokenizer` znajdujący się w pakiecie `edu.mimuw.nmpt.msw.creator.tm.tokenizer`:

Listing 5.12: `ISegmentTokenizer.java`

```

package edu.mimuw.nmpt.msw.creator.tm.tokenizer;

```

```

import java.util.List;

public interface ISegmentTokenizer {

    /**
     * Metoda dzieli tekst na listę wyrazów.
     *
     * @param text tekst do podziału
     * @return kolekcja obiektów IWord
     */
    public List tokenize(String text);

}

```

Konfiguracja

Program umożliwia konfigurację sposobu podziału segmentu na wyrazy w zależności od języka, w jakim napisany jest dany segment pamięci tłumaczeniowej. Aby użyć innej niż domyślna implementacji, należy w pliku konfiguracyjnym dodać linię przyporządkującą daną klasę do obsługi konkretnego języka tak, jak jest to zrobione na listingu 5.8 w linii 19 dla języka angielskiego. Dodawany wpis powinien mieć klucz postaci *segment.tokenizer.LN*, gdzie *LN* oznacza dwuliterowy identyfikator języka. Wartością powinna być pełna nazwa pliku.

Program udostępnia implementację podziału segmentów na wyrazy filtrującą wybrane słowa. Aby jej użyć należy dodać odpowiedni wpis do pliku konfiguracyjnego określając klasę jako *edu.mimuw.nmpt.msw.creator.tm.tokenizer.FilteredSegmentTokenizer* oraz konfigurując listę filtrowanych słów analogicznie jak w linii 20 listingu 5.8.

W przypadku własnych klas obsługujących podział segmentów należy zaimplementować interfejs *ISegmentTokenizer* oraz zadbać o to, aby klasa posiadała konstruktor, którego jedyny argument będzie typu *java.util.Locale*, jak w przypadku implementacji domyślnej:

Listing 5.13: DefaultSegmentTokenizer.java

```

...
public class DefaultSegmentTokenizer implements ISegmentTokenizer {

    Locale locale;

    public DefaultSegmentTokenizer( Locale locale ) {
        this.locale = locale;
    }

    ...
}

```

5.4.4. Analiza morfologiczna

Słowniki zawierają tłumaczenia podstawowych form wyrazów, natomiast segmenty pamięci tłumaczeniowych zawierają dowolne formy. Stworzenie narzędzia umożliwiającego tworzenie słowników na podstawie pamięci tłumaczeniowych wymagało implementację mechanizmu umożliwiającego przeprowadzanie analizy morfologicznej wyrazów tak, aby móc określić ich podstawowe formy. Do tego celu wydzielono interfejs a pakiecie *edu.mimuw.nmpt.msw.creator.dict.ma* o nazwie *IMorphologicAnalyzer*. Domyślna implementacja interfejsu znajduje się w tym samym pakiecie w klasie *DefaultMA* i dla danego słowa zwraca listę

jednoelementową zawierającą formę o identycznym brzmieniu jak podane słowo. Przykładem implementacji analizy morfologicznej jest klasa `PolishMA` przeprowadzająca analizę morfologiczną dla języka polskiego przy pomocy programu `Morfeusz`.

Listing 5.14: `IMorphologicAnalyzer.java`

```
package edu.mimuw.nmpt.msw.creator.dict.ma;

import java.util.Collection;
import java.util.List;

/**
 * Interfejs umożliwiający przeprowadzenie analizy
 * morfologicznej kolekcji słów.
 *
 * @author marcin
 */
public interface IMorphologicAnalyzer {

    /**
     * Metoda wykonywana po wczytaniu pamięci tłumaczeniowej
     * umożliwia inicjalizację i przeanalizowanie wszystkich
     * słów z kolekcji.
     *
     * @param words kolekcja obiektów typu String
     */
    void initTranslationMemory( Collection words );

    /**
     * Zwraca listę form dla danego wyrazu
     *
     * @param word słowo, którego form szukamy
     * @return lista obiektów typu IForm
     */
    List getForms(String word);
}
```

Konfiguracja

Podobnie jak w przypadku klas implementujących podział segmentu na wyrazy możliwa jest konfiguracja różnych klas implementujących analizę morfologiczną dla poszczególnych języków. Konfiguracja polega na dodaniu w pliku konfiguracyjnym wpisu określającego pełną nazwę klasy obsługującej dany język. Klucz w pliku konfiguracyjnym powinien mieć postać *morphological.analyzer.LN*, gdzie *LN* jest dwuliterowym skrótem obsługiwanego języka. Podobnie jak w przypadku klas implementujących podział segmentu na wyrazy należy również zadbać o to, aby implementacje interfejsu `IMorphologicAnalyzer` posiadały konstruktor, którego jedyny argument będzie typu `java.util.Locale`, określającego język, który będzie obsługiwany przez klasę.

Podsumowanie

W dobie powszechnej komputeryzacji i ogólnodostępnego Internetu, szybki dostęp do informacji i usług jest niezbędny do realizacji zadań stawianych przed nami każdego dnia. Współczesny tłumacz, zamiast kartki i długopisu, korzysta w swojej pracy z komputera, a drukowany słownik często jest tylko uzupełnieniem zasobów internetowych. Z jednej strony, globalizacja wymaga prezentowania usługi w ojczystym języku klienta oraz koordynacji i szybkiego wdrażania zmian w wielu krajach równocześnie. Z drugiej strony, mamy rozwijający się rynek otwartych projektów tworzących darmowe oprogramowanie i powiększające się grupy ludzi zaangażowanych w ich lokalizację oraz tłumaczenie dokumentacji. Korporacjom zależy na szybkim wprowadzeniu usług w różnych krajach, tłumaczom hobbystom zależy na przyjemnej pracy, do której wyników możliwy jest jak najszybszy dostęp szerokiemu gronu odbiorców.

Analiza dostępnego oprogramowania wykazała lukę w postaci braku narzędzi do łatwego tworzenia słowników na podstawie wyników pracy tłumaczy, którymi są pamięci tłumaczeniowe. Lukę tę stara się wypełnić program „Kreator Słowników”, który powstał w ramach niniejszej pracy magisterskiej. Mam nadzieję, że program okaże się przydatny i będzie używany przez tłumaczy korzystających z oprogramowania typu CAT, oraz że będzie inspiracją dla kolejnych narzędzi wspierających tworzenie słowników. Program został umieszczony wśród innych programów wspomagających tłumaczenie tekstów w nowej dystrybucji Linuxa opartej na dystrybucji KNOPPIX. Mam nadzieję, że dystrybucja ta będzie rozwijana i uzupełniana o kolejne programy, oraz że przyczyni się do rozpowszechnienia wolnego oprogramowania wspierającego pracę tłumaczy.

Dodatek A

Kreator Słowników - Podręcznik Użytkownika

Program „Kreator Słowników” dostępny jest na zasadach licencji GNU GPL, natomiast jego dokumentacja na zasadach licencji GNU FDL, których treści znaleźć można pod adresem: <http://www.gnu.org/licenses/>.

A.1. Wstęp

W kolejnych rozdziałach opisane zostały:

Rozdział A.3: zawiera dokładny opis interfejsu użytkownika, przybliżając użytkownikowi możliwości aplikacji.

Rozdział A.10: opisana tu została przykładowa sesja użytkownika z programem. Rozdział ten jest krótkim samouczkiem, pokazującym krok po kroku wykonywane czynności.

Rozdział A.11: W tym rozdziale czytelnik znajdzie odpowiedzi na najczęściej zadawane pytania.

A.2. Opis programu

Jedną z grup programów CAT są słowniki elektroniczne i systemy zarządzania terminologią. Brak powszechnie dostępnych darmowych narzędzi wspomagających tworzenie słowników stał się inspiracją do napisania programu umożliwiającego tłumaczowi stworzenie bazy terminologicznej. Głównym celem było napisanie aplikacji, która umożliwi szybkie i proste tworzenie słownika na podstawie wcześniej przetłumaczonych przez użytkownika tekstów. Jako dane wejściowe wybrano pamięci tłumaczeniowe, które zawierają autonomiczne segmenty w języku źródłowym i ich tłumaczenia. Użytkownik aplikacji tworzy słownik, kojarząc ze sobą pary słów, co odbywa się poprzez ich wskazanie z lisy słów znajdujących się w segmencie tekstu źródłowego i jego tłumaczeniu. Dużo pracy poświęcono temu, aby interfejs użytkownika był przejrzysty, a jednocześnie wygodny w użyciu. Aby osiągnąć wspomniany cel, zaprojektowano odpowiednie komponenty interfejsu użytkownika, umożliwiając komunikację między nimi, co w konsekwencji umożliwiło zaimplementowanie w pełni interaktywnego pulpitu sterującego aplikacją.

Obok zaimplementowanej funkcjonalności i interfejsu użytkownika, przed aplikacją postawiono następujące cele:

- łatwość rozbudowy o kolejne formaty danych przechowujące pamięć tłumaczeniową,

Termin	Opis
pamięć tłumaczeniowa	Plik zawierający tłumaczenia segmentów tekstu. Program „Kreator Słowników” obsługuje dwa formaty takich plików: TMX oraz PO. Pliki pamięci tłumaczeniowych tworzone są przez programy typu CAT.
TMX	(ang. Translation Memory eXchange format) - jest formatem standardyzującym opisywanie pamięci tłumaczeniowych. Format umożliwia wymianę danych między różnymi narzędziami i/lub twórcami oprogramowania.
TEI	(Text Encoding Initiative) międzynarodowy i interdyscyplinarny standard umożliwiający reprezentację różnych tekstów do celów naukowych.
segment	Jednostka tekstu tłumaczona przez programy CAT. Najczęściej segmentami są części zdania oddzielone przecinkami, zdania oraz paragrafy.
aktywny segment	Segment aktualnie załadowany do aplikacji „Kreator słowników”. Na podstawie aktywnego (aktualnego) segmentu tworzone są listy źródłowa i tłumaczeń (patrz. „Aktywny segment pamięci tłumaczeń”)
słownik	Słownik obsługiwany przez program „Kreator słowników”
słownik domyślny	Otwarty słownik oznaczony jako „domyślny” (patrz „zarządzanie słownikami”). Wszystkie tłumaczenia tworzone za pomocą listy tłumaczeń zapisywane są do słownika domyślnego.
lista źródłowa	Lista słów aktywnego segmentu (patrz „Aktywny segment pamięci tłumaczeniowej”).
lista tłumaczeń	Lista słów tłumaczenia aktywnego segmentu (patrz „Aktywny segment pamięci tłumaczeniowej”).
słowo źródłowe	Słowo zaznaczone na liście źródłowej

Tabela A.1: Słownik pojęć używanych w podręczniku

- łatwość rozbudowy o kolejne formaty danych przechowujące bazę terminologiczną,
- internacjonalizacja prowadząca do łatwej lokalizacji,
- łatwa rozbudowa o wtyczki filtrujące słowa w segmentach pamięci tłumaczeniowej (np. angielskie przedimki) i konfiguracja filtrów w zależności od języka,
- interfejs umożliwiający wykorzystanie dowolnego narzędzia przeprowadzającego analizę morfologiczną,
- możliwość tworzenia słownika zawierającego różne formy hasel.

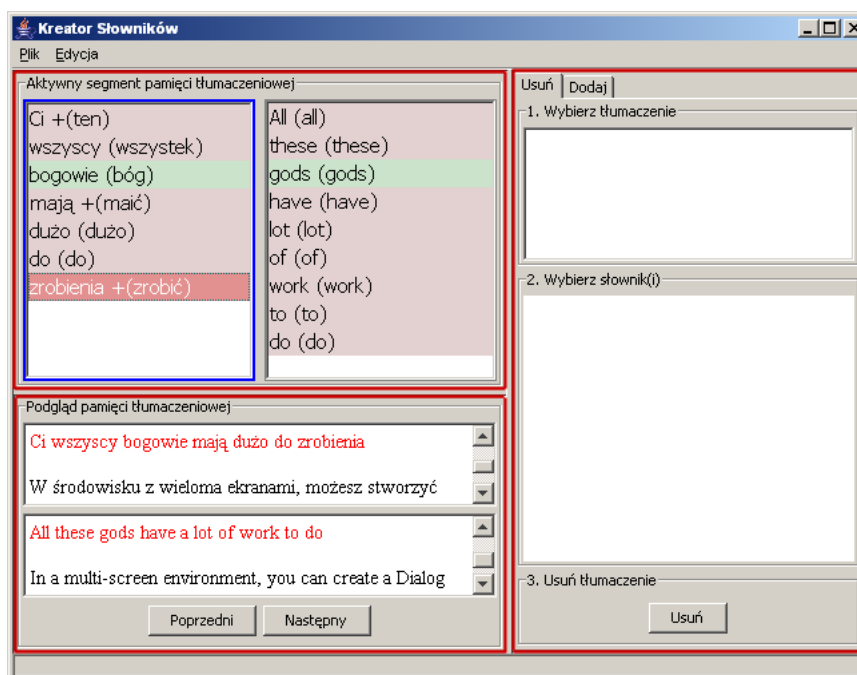
Wszystkie te cechy sprawiają, że „Kreator Słowników” może mieć różne zastosowanie, a możliwości jego integracji z innymi aplikacjami są na tyle szerokie, że pozwalają na dostosowanie programu do indywidualnych potrzeb użytkownika.

Na potrzeby prezentacji programu „Kreator Słowników” zdefiniowano szereg pojęć, wykorzystywanych w dalszej części dokumentu (Tabela A.1).

A.3. Interfejs użytkownika

Interfejs użytkownika programu „Kreator Słowników” składa się z okna głównego podzielonego na funkcjonalne obszary, umożliwiające użytkownikowi intuicyjną nawigację po różnych obiektach w aplikacji. Główne obszary okna aplikacji zostały wyróżnione na rysunku A.1:

- obszar podglądu pamięci tłumaczeniowej – umożliwia nawigację po pamięci tłumaczeniowej, to znaczy wybór aktywnego segmentu z TM,
- obszar aktywnego segmentu pamięci tłumaczeniowej – wyświetla wyrazy z aktywnego segmentu pamięci tłumaczeniowej umożliwiając kojarzenie wyrazów źródłowych i ich tłumaczeń. Skojarzone w tym obszarze wyrazy zostaną dodane do domyślnego słownika,
- obszar edycji słownika – umożliwia użytkownikowi ręczną edycję słownika: zawiera zakładkę „usuń”, umożliwiającą usuwanie tłumaczeń z wybranych słowników, oraz zakładkę „dodaj”, umożliwiającą dodanie tłumaczenia przez wpisanie go przez użytkownika.

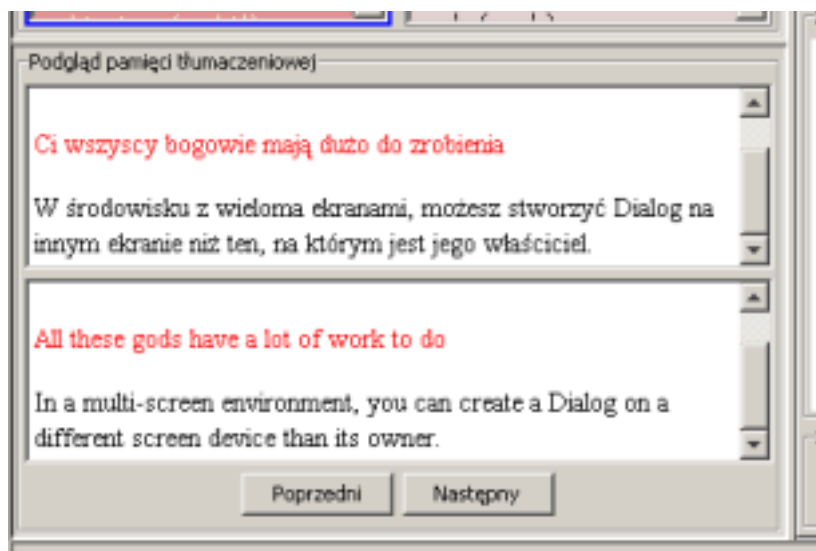


Rysunek A.1: Główne okno aplikacji

A.3.1. Podgląd pamięci tłumaczeniowej

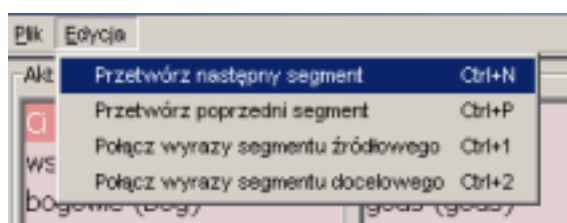
Nawigację po pamięci tłumaczeniowej umożliwia komponent nazwany „Podglądem pamięci tłumaczeniowej” (Rysunek A.2). Jest on podzielony na dwa obszary: pola pokazujące aktualny kontekst pamięci tłumaczeniowej oraz przyciski do nawigacji między jej kolejnymi segmentami. Pole znajdujące się na górze pokazuje tekst źródłowy, dolne pokazuje tłumaczenie tekstu źródłowego. W obu polach znajdują się maksymalnie trzy segmenty pamięci tłumaczeniowej: wyróżniony kolorem czerwonym, aktualnie przetwarzany (aktywny) segment znajdujący się

po środku oraz segmenty poprzedzający i następujący po aktywnym. W przypadku, gdy aktywny jest pierwszy segment pamięci tłumaczeniowej, nie jest wyświetlany poprzedni, a w przypadku, gdy aktywny jest ostatni, nie jest wyświetlany segment następny.



Rysunek A.2: Część okna związana z prezentacją i nawigacją między segmentami

Pod obszarem pokazującym aktualny kontekst pamięci tłumaczeniowej znajdują się przyciski służące do nawigacji: „Poprzedni” - przechodzi do poprzedniego segmentu, „Następny” - do następnego. Nawigacja po segmentach pamięci tłumaczeniowej dostępna jest również z menu aplikacji (rysunek A.3) oraz za pomocą skrótów klawiszowych: „Ctrl+N” – następny, oraz „Ctrl+P” – poprzedni.



Rysunek A.3: Widok menu „Edycja”

Podsumowanie

1. Podgląd pamięci tłumaczeniowej prezentuje użytkownikowi aktualny kontekst aktywnego segmentu. Aktualnie przetwarzany segment prezentowany jest w kolorze czerwonym.
2. Aby przejść do kolejnego segmentu pamięci tłumaczeniowej należy:
 - wcisnąć przycisk „Następny”,
 - z menu „Edycja” wybrać „Przetwórz następny segment”,
 - użyć skrótu klawiszowego „Ctrl+N”.

3. Aby przejść do poprzedniego segmentu pamięci tłumaczeniowej należy:

- wcisnąć przycisk „Poprzedni” ,
- z menu „Edycja” wybrać „Przetwórz poprzedni segment” ,
- użyć skrótu klawiszowego „Ctrl+P” .

4. Zmiana aktywnego segmentu automatycznie aktualizuje pozostałe części aplikacji, umożliwiając tworzenie tłumaczeń opartych na tym segmencie.

Przełączanie segmentów zsynchronizowane jest z częścią okna aplikacji, prezentującą aktualny segment pamięci tłumaczeniowej.

A.3.2. Aktywny segment pamięci tłumaczeniowej

Większość pracy użytkownika aplikacji „Kreator słowników” związana jest z częścią nazwaną „Aktywnym segmentem pamięci tłumaczeniowej”. Wszystkie tłumaczenia tworzone w tej części aplikacji zapisywane są do słownika domyślnego. Wybór tego słownika odbywa się za pomocą opcji „zarządzanie słownikami” opisaną w dalszej części instrukcji.

„Aktywny segment pamięci tłumaczeniowej” składa się z dwóch list, umożliwiających kojarzenie wyrazów w pary, które następnie zapisywane są do słownika. Listy zawierają wyrazy składające się na aktywny segment pamięci tłumaczeniowej: lista znajdująca się po lewej stronie (lista źródłowa) zawiera listę wyrazów segmentu źródłowego, lista po prawej (lista tłumaczeń) zawiera natomiast wyrazy tłumaczenia tego segmentu.

Użytkownik, używając myszki, wybiera z list słowa. Po wyborze słowa z listy źródłowej, aplikacja podświetli na liście tłumaczeń tłumaczenia znajdujące się w obsługiwanych przez nią słownikach. Podświetlenie to ma kolor żółty.

Aby stworzyć nowe tłumaczenie po wyborze słowa z listy źródłowej, należy wybrać jego tłumaczenie na liście tłumaczeń. Wybrane słowo będzie podświetlone kolorem zielonym.

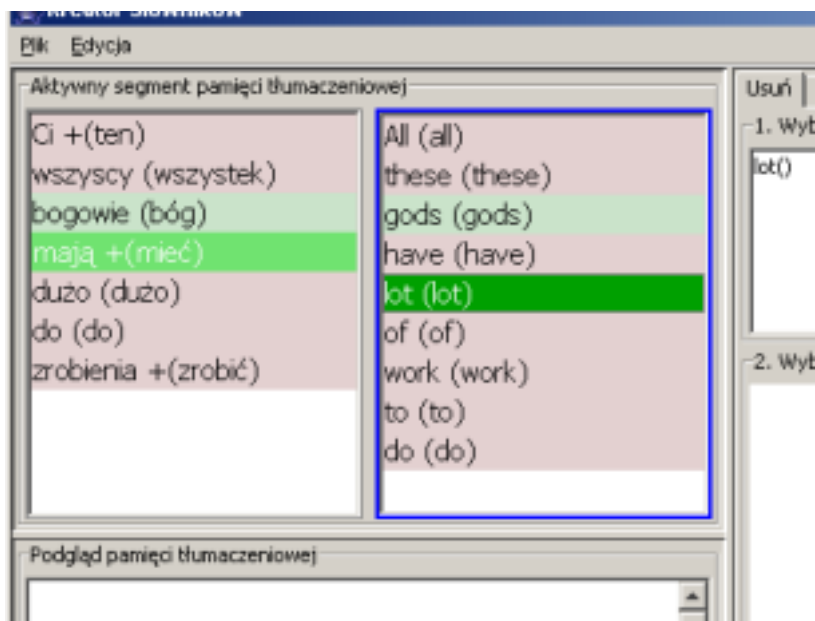
Poniższy obrazek (A.4) pokazuje stworzone przez użytkownika tłumaczenie: z listy źródłowej wybrane zostało słowo „bogowie”, a następnie na liście tłumaczeń słowo „lot”.

W powyższym przykładzie, użytkownik pomylił się, wybierając niepoprawne tłumaczenie wyrazu „mieć”. Aby naprawić ten błąd, należy ponownie wybrać słowo „lot” z listy tłumaczeń. W ten sposób wcześniejszy wybór użytkownika zostanie cofnięty. Można w ten sposób usunąć tłumaczenie już istniejące w domyślnym słowniku. Z dowolnego słownika można również usunąć tłumaczenie przy pomocy obszaru aplikacji znajdującego się po prawej stronie głównego okna (patrz „Edycja słowników”).

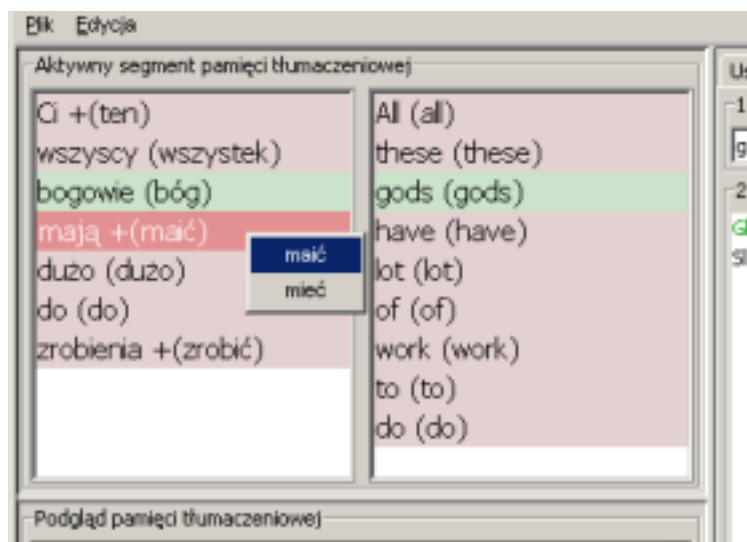
Program „Kreator Słowników” przeprowadza analizę morfologiczną słów występujących w przetwarzanych segmentach. Zdarza się, że wyrazy te posiadają więcej niż jedną formę podstawową. W takich przypadkach użytkownik ma możliwość wyboru odpowiedniej formy. Sytuacja, w której wyraz ma więcej niż jedną formę sygnalizowana jest użytkownikowi przez dodanie znaku „+” przed formą zapisaną w nawiasie po każdym z wyrazów umieszczonych na liście. Aby wybrać inną formę, należy wybrać słowo, a następnie prawym przyciskiem myszy wywołać menu podręczne z listą dostępnych form. Przykład zmiany formy słowa pokazano na rysunku A.5.

Podsumowanie

1. Tworzenie tłumaczenia należy rozpocząć wyborem słowa z listy źródłowej; każdy wybór słowa z listy tłumaczeń związany jest z tym właśnie słowem.



Rysunek A.4: Wybranie błędnego tłumaczenia



Rysunek A.5: Zmiana formy bazowej słowa

2. Na liście tłumaczeń, aplikacja prezentuje tłumaczenia wybranego słowa źródłowego. Tłumaczenie znajdujące się w słowniku podświetlane jest na żółto. Tłumaczenie wybrane przez użytkownika podświetlane jest na zielono.
3. Aby usunąć wybrane przez użytkownika tłumaczenie należy wybrać je ponownie (klikając na nie myszką).
4. Aby zmienić formę bazową słowa na inną, należy wybrać odpowiednią formę z menu podręcznego wywoływanego prawym przyciskiem myszy na wybranym słowie.

A.4. Edycja słowników

Omawiana część aplikacji umożliwia edycję wszystkich otwartych przez użytkownika słowników (w przeciwieństwie do „Listy tłumaczeń aktywnego segmentu pamięci tłumaczeniowej” umożliwiającego szybką edycję słownika domyślnego). Dostępne są tutaj dwie zakładki, umożliwiające usuwanie i dodawanie tłumaczeń. Operacje na obu z nich związane są z aktualnie wybranym słowem z listy źródłowej aktywnego segmentu tłumaczeń i polegają na wykonaniu trzech czynności:

1. wyborze słowa do usunięcia lub wpisaniu słowa do dodania,
2. wyborze słownika(ów), na którym(ch) będzie wykonywana operacja,
3. zatwierdzeniu operacji przez wciśnięcie przycisku „Usuń” /”Dodaj”.

A.4.1. Usuwanie tłumaczeń ze słownika

Poniższy rysunek (A.6) przedstawia część aplikacji umożliwiającą usunięcie tłumaczenia ze słowników.

Na liście tłumaczeń znajdują się wszystkie tłumaczenia słowa źródłowego. W pierwszym kroku należy wybrać tłumaczenie, które użytkownik zamierza usunąć. Następnie należy wybrać słownik(i), z którego tłumaczenie będzie usunięte. Aby usunąć tłumaczenie z wybranych słowników, należy wybrać „Usuń”.

UWAGA: Aby usunąć tłumaczenie z kilku słowników, należy wybrać je wszystkie z listy, trzymając przycisk „CTRL”.

A.4.2. Dodawanie tłumaczeń do słownika

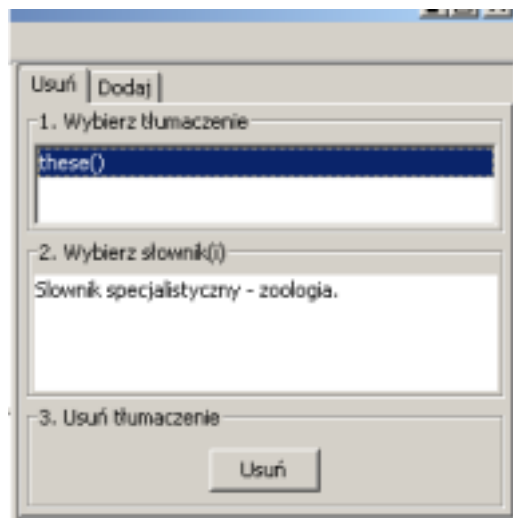
Poniższy rysunek (A.7) przedstawia część aplikacji, umożliwiającą dodanie tłumaczenia do słowników.

W pierwszym kroku należy wprowadzić tłumaczenie, które użytkownik zamierza dodać. Następnie należy wybrać słownik(i), do którego tłumaczenie będzie dodane, a następnie wybrać ”Dodaj”.

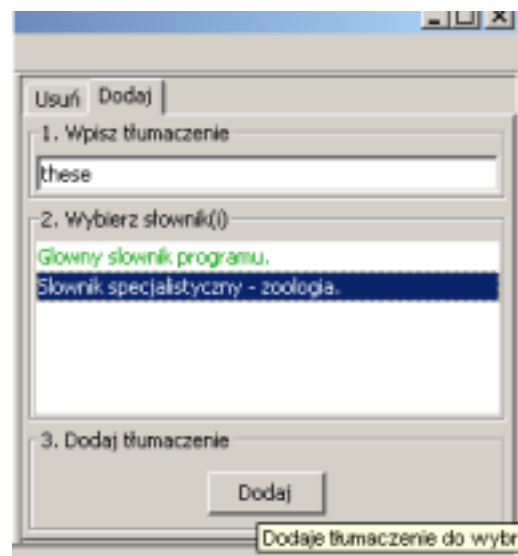
UWAGA: Aby dodać tłumaczenie do kilku słowników, należy wybrać je wszystkie z listy, trzymając przycisk „CTRL”.

A.5. Otwieranie pamięci tłumaczeniowej

Aby otworzyć plik pamięci tłumaczeniowej, należy w menu „Plik” wybrać polecenie „Otwórz plik pamięci tłumaczeniowej” A.8 lub zastosować skrót „Ctrl-T”. Po wyborze tej opcji pojawi

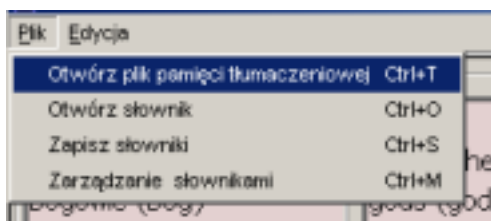


Rysunek A.6: Usuwanie słów ze słownika

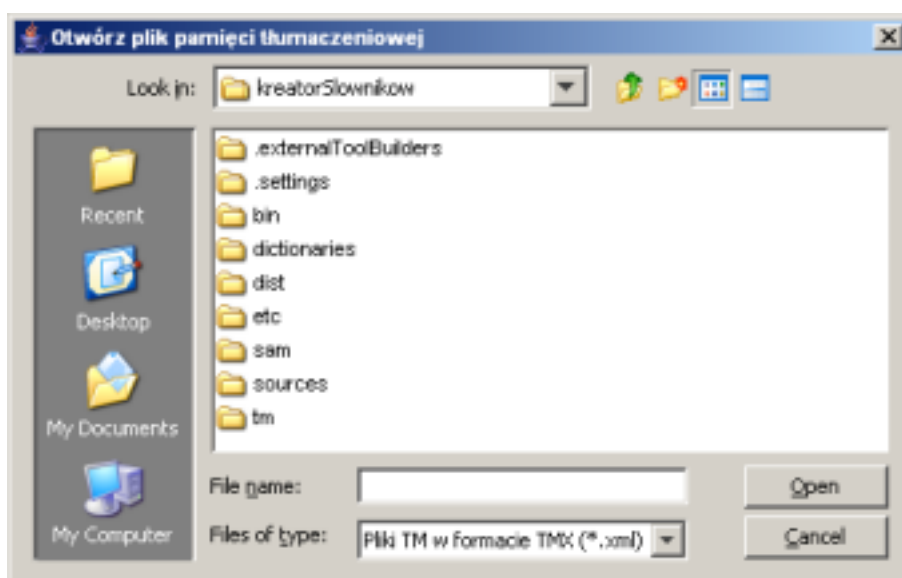


Rysunek A.7: Ręczne dodawanie słów do słownika

się okno dialogowe, umożliwiające wybór pliku do otwarcia A.9. Wybór słownika należy zatwierdzić przyciskiem „Open”.



Rysunek A.8: Menu „Plik”



Rysunek A.9: Otwieranie pliku pamięci tłumaczeniowej

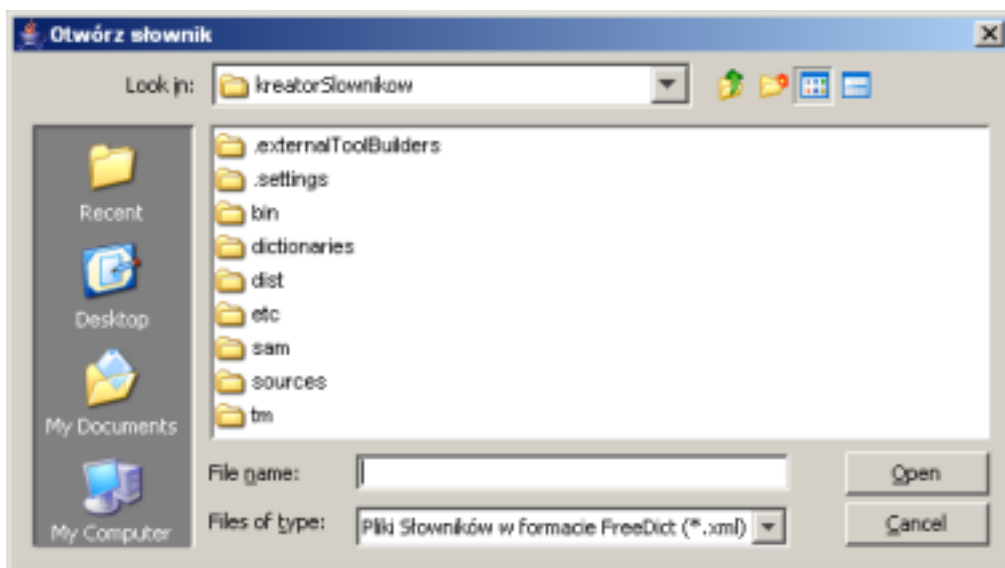
A.6. Otwieranie słownika

Aby otworzyć słownik, należy w menu „Plik” wybrać polecenie „Otwórz słownik” A.8 lub zastosować skrót „Ctrl-O”. Po wyborze tej opcji pojawi się okno dialogowe, umożliwiające wybór słownika do otwarcia A.10. Wybór słownika należy zatwierdzić przyciskiem „Open”.

A.7. Zapisywanie słowników

W każdym momencie pracy z programem, użytkownik ma możliwość zapisania aktualnej zawartości słowników do plików. Zapisywanie zawartości wszystkich słowników możliwe jest przez wybór opcji „Zapisz słowniki” z menu „Plik” (rysunek A.8) lub zastosowanie skrótu „Ctrl-S”.

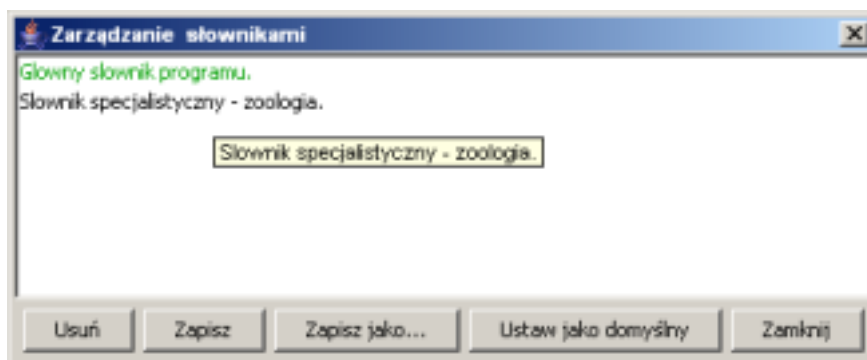
Aby zapisać poszczególne słowniki należy użyć „Menadżera słowników”.



Rysunek A.10: Otwieranie słownika

A.8. Menadżer słowników

Aby przejść do okna umożliwiającego zarządzanie słownikami, należy w menu „Plik” wybrać „Zarządzanie słownikami” A.8 lub zastosować skrót „Ctrl-M”. Wybór tej opcji wyświetli okno dialogowe, umożliwiające zamknięcie oraz wybór domyślnego słownika (słownika, do którego zapisywane są tłumaczenia stworzone za pomocą „aktywnego segmentu pamięci tłumaczeniowej”) (rysunek A.11).



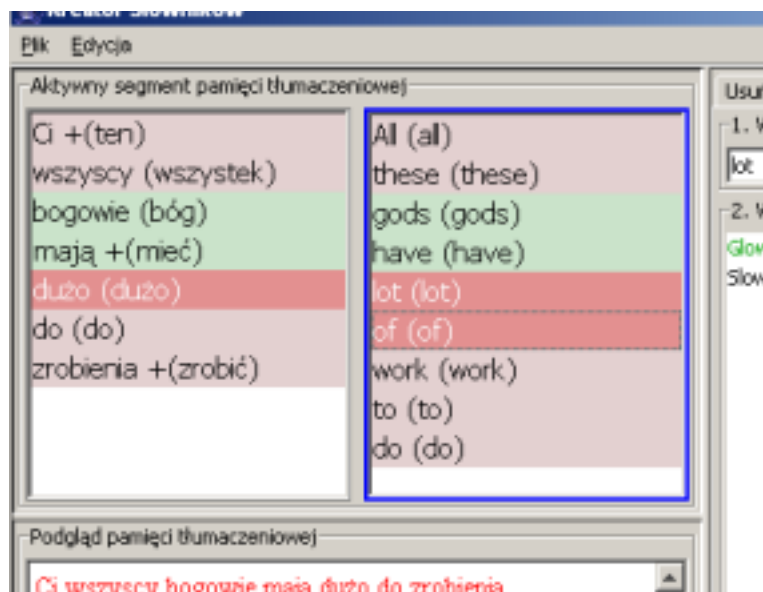
Rysunek A.11: Menadżer słowników

Okno dialogowe składa się z listy otwartych słowników z wyróżnionym na zielono słownikiem domyślnym. Aby usunąć słownik z listy, należy wybrać ten słownik, a następnie wybrać przycisk „Usuń”. Ustawienie domyślnego słownika polega na jego wyborze oraz wyborze przycisku „Ustaw jako domyślny”. Aby zapisać zawartość wybranego słownika do pliku należy wybrać przycisk „Zapisz”. Przycisk „Zamknij” zamyka okno „Zarządzanie słownikami”.

A.9. Łączenie wyrazów

Czasami niezbędne jest utworzenie ze słów znajdujących się na liście słów segmentów źródłowego/docelowego związków frazeologicznych. Program umożliwia łączenie wyrazów za pomocą skrótów „Ctrl-1” oraz „Ctrl-2” łączących zaznaczone wyrazy odpowiednio w segmencie źródłowym i docelowym. Opcje te można również wybrać z menu „Edycja”.

Aby połączyć wyrazy należy je wybrać myszką trzymając równocześnie przycisk „Ctrl”. Następnie należy wybrać odpowiednią komendę łączącą wyrazy. Ilustracje obrazujące ten proces znajdują się na rysunkach A.12 oraz A.13.



Rysunek A.12: Łączenie słów: stan przed połączeniem słów „lot” i „of”

A.10. Tworzenie słownika krok po kroku

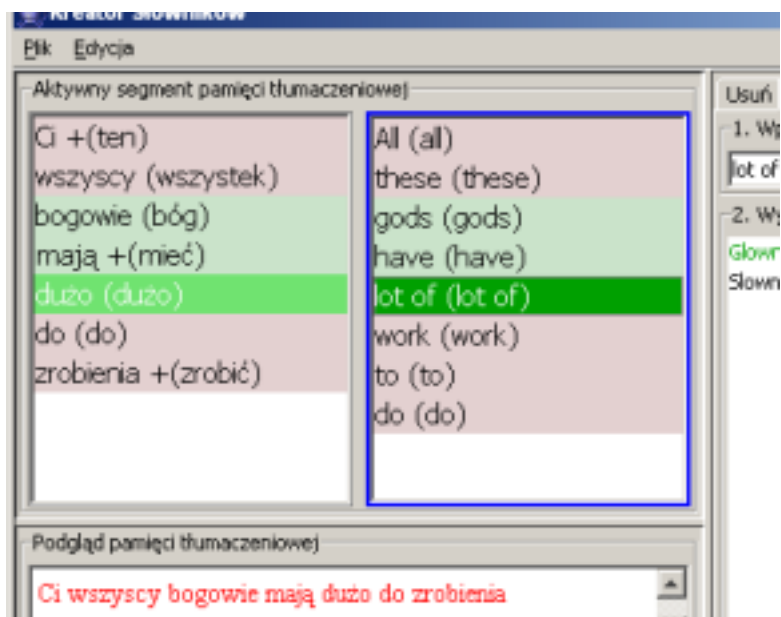
Poniżej opisana została przykładowa sesja użytkownika z programem. Zostały tu uporządkowane czynności, z którymi spotyka się każdy użytkownik programu.

1. Wczytywanie słowników

Po uruchomieniu program automatycznie wczytuje słownik systemowy (znajdujący się w pliku „etc/systemdictionary.tei”). Dodatkowo, wczytywane są wszystkie słowniki znajdujące się w podkatalogu „dictionaries” oraz wszystkie słowniki otwarte podczas ostatniej sesji aplikacji. Jako domyślny, ustawiany jest słownik, który ustawiony był jako domyślny podczas poprzedniej sesji aplikacji.

Jeśli użytkownik chce wczytać dodatkowe słowniki, może to zrobić wybierając z menu „Plik” opcję „Otwórz słownik” lub stosując skrót „Ctrl-O”.

Użytkownik ma dostęp do pełnej listy dostępnych słowników, które może otworzyć/zamknąć w dowolnym momencie działania systemu. Podczas analizy poszczególnych segmentów do odszukania kolejnych tłumaczeń służą wszystkie otwarte słowniki.



Rysunek A.13: Łączenie słów: stan po połączeniu słów „lot” i „of”

2. Wybór słownika domyślnego

Tłumaczenia, które użytkownik chce zapamiętać zapisywane są do tzw. słownika domyślnego. Użytkownik może zmienić słownik domyślny w dowolnym momencie działania programu. Aby zmienić słownik domyślny, należy z menu „Plik” wybrać „Zarządzanie słownikami”, następnie wybrać z listy słownik i przycisk „Ustaw jako domyślny”.

3. Wczytywanie pamięci tłumaczeniowej

Podstawowym interfejsem do edycji słowników jest interfejs oparty na pamięci tłumaczeniowej. Składa się on z dwóch obszarów: listy źródłowej i tłumaczeń. Aby móc skorzystać z tego interfejsu, należy wczytać pamięć tłumaczeniową (TM). Aby to zrobić, należy z menu „Plik” wybrać „Otwórz plik pamięci tłumaczeniowej”, następnie z rozwijanej listy wybrać typ pliku (obecnie obsługiwane są formaty TMX (domyślny) oraz PO), wybrać plik oraz wcisnąć „Otwórz”.

4. Nawigacja po pamięci tłumaczeniowej

Do tworzenia tłumaczeń w słowniku domyślnym służą dwie listy: źródłowa i tłumaczeń – zawierają one aktywny segment wczytanej pamięci tłumaczeniowej. Do nawigacji po pamięci tłumaczeniowej służą przyciski „Poprzedni” oraz „Następny”.

5. Podgląd wszystkich tłumaczeń aktywnej frazy segmentu źródłowego

Po wyborze słowa z listy źródłowej, po prawej stronie okna aplikacji wyświetlane są wszystkie znane (znajdujące się w otwartych słownikach) tłumaczenia.

6. Przyporządkowywanie tłumaczeń z segmentu docelowego (tłumaczenia)

Aby dodać do słownika nowe tłumaczenie, należy wybrać słowo z listy tłumaczeń. Para składająca się z wyrazów zaznaczonych na listach źródłowej i docelowej dodawane są do słownika domyślnego.

Użytkownik ma możliwość dodania własnego tłumaczenia do wybranego słownika. Aby to zrobić, należy panel „Edycja słownika” przełączyć na zakładkę „Dodaj”, wpisać tłumaczenie, wybrać słowniki oraz zatwierdzić przyciskiem „Dodaj”.

A.11. Pytania i odpowiedzi (FAQ)

1. Dlaczego podświetlenie tłumaczenia znika, kiedy wybieram źródłowy wyraz tłumaczenia?

Podświetlone jest zawsze tłumaczenie aktualnie zaznaczonego wyrazu z segmentu źródłowego. W momencie, gdy wybierzesz nowy wyraz, podświetlone zostanie tłumaczenie właśnie tego słowa (jeśli w słowniku nie znajduje się informacja o tłumaczeniu wybranego wyrazu, wszystkie podświetlenia w segmencie docelowym znikną).

2. Zazaczyłam tłumaczenie, jak z niego zrezygnować?

Aby zrezygnować z wpisu w słowniku, masz dwie możliwości:

- zaznaczyć ponownie niechciane tłumaczenie
- użyć opcji usuwania wpisów ze słownika przy pomocy prawej części okna głównego.

3. Po skojarzeniu pary słów, z segmentu źródłowego i segmentu docelowego, tłumaczone słowo podświetliło się na zielono. Kiedy po stworzeniu kilku tłumaczeń w kolejnych segmentach wróciłam do tego segmentu, tłumaczenie podświetlone było na żółto i nie mogłam go odznaczyć. Dlaczego?

W momencie zmiany segmentu, wszystkie pary słów skojarzone podczas pracy z tym segmentem zostają zapisane do słownika. Po powrocie do segmentu, w którym dokonano parowania słów, tłumaczenia oznaczone są kolorem żółtym, co oznacza, że nie są już tłumaczeniami roboczymi, a zatwierdzonymi tłumaczeniami słów ze słownika.

4. W tłumaczeniu segmentu znalazłam błąd. Jak wpisać do słownika poprawne tłumaczenie?

Do modyfikacji (usunięcia i dodania poprawnego) tłumaczenia służy prawa część głównego okna aplikacji.

5. Co to jest słownik domyślny?

Podczas pracy z programem jeden z otwartych słowników jest słownikiem domyślnym. To do tego słownika zapisywane są wszystkie pary słów skojarzone przez ich wybór z list słów segmentów. Możesz zmienić domyślny słownik, wybierając menu 'Plik -> Zarządzanie słownikami'.

6. Czy do pracy z programem muszę korzystać z myszki?

Nie. Myszka jest tylko narzędziem ułatwiającym pracę. Jeśli nie chcesz, lub nie możesz korzystać z myszki, możesz posługiwać się wyłącznie klawiaturą. Poniżej zamieszczona została tabela (A.2) prezentująca stosowane skróty klawiaturowe.

7. Czy mogę zmienić skróty klawiaturowe przypisane do poszczególnych akcji w programie?

Skrót klawiaturowy	Przypisana akcja
Ctrl-N	Przechodzenie do kolejnego segmentu pamięci tłumaczeniowej
Ctrl-P	Przechodzenie do poprzedniego segmentu pamięci tłumaczeniowej
Ctrl-Q	Wyjście z programu
Ctrl-T	Wyświetlenie okna dialogowego umożliwiającego otworzenie pamięci tłumaczeniowej
Ctrl-O	Wyświetlenie okna dialogowego umożliwiającego otworzenie pliku słownika
Ctrl-S	Zapisanie wszystkich słowników
Ctrl-M	Wyświetlenie okna umożliwiającego zarządzanie słownikami
Ctrl-1	Połączenie zaznaczonych słów z listy źródłowej
Ctrl-2	Połączenie zaznaczonych słów z listy docelowej
Tab	Przechodzenie do kolejnego komponentu interfejsu użytkownika (panele, listy, przyciski)
Shift-Tab	Przechodzenie do poprzedniego komponentu interfejsu użytkownika (panele, listy, przyciski)
Space	Wybór aktywnego elementu (element na liście, przycisk)

Tabela A.2: Skrótów klawiaturowe używane w programie

Tak. Wraz z programem w katalogu *locale* znajduje się plik *captions_LN.properties* zawierające zlokalizowane komunikaty programu oraz przypisanie skrótów klawiaturowych do funkcji programu. W tym pliku można zmienić przypisane skróty na takie, które są dla użytkownika wygodne. Każda wersja językowa ma swój plik konfiguracyjny, można więc stosować różne skróty w różnych wersjach językowych. Na przykład dla polskiej wersji językowej należy zmienić konfigurację zapisaną w pliku *captions_pl.properties*.

8. Na jakich zasadach mogę korzystać z programu?

Program „Kreator Słowników” jest programem udostępnianym na zasadach licencji GNU. Treść licencji można znaleźć pod adresem:
<http://www.gnu.org.pl/text/licencja-gnu.html>.

Dodatek B

Wybrane formaty plików

B.1. debiandoc-sgml

Format *debiandoc-sgml* używany jest w projekcie Debian Documentation Project do formatowania tworzonej dokumentacji. Pliki w tym formacie mają dość przejrzystą budowę i dają się łatwo edytować. Stworzonych jest szereg narzędzi umożliwiających generowanie na podstawie plików w tym formacie stron HTML oraz plików PDF.

Listing B.1: Plik w formacie debiandoc-sgml

```
<!doctype debiandoc system>
<debiandoc>

<book>

  <title>TEST HOWTO</title>

  <author>
    <name>Marcin Świnoga</name> <email>marcin.swingoa@gmail.com</email>
  </author>

  <version>1.0.1</version>

  <abstract>Streszczenie podręcznika</abstract>

  <copyright>
    <copyrightsummary>Copyright &copy; 2006 Marcin Świnoga</copyrightsummary>
    <p>Niniejszy podręcznik dostępny jest na zasadach licencji GNU</p>
  </copyright>

  <toc>

  <chapt>Wprowadzenie
    <p>
      Przykład wywołania programu java:
    <p>
      <example>
        java -jar nazwa_biblioteki.jar
      </example>
    </chapt>

</book>

</debiandoc>
```

B.2. docbook

Format szeroko używany do tworzenia dokumentacji. DTD docbooka jest bardzo rozbudowane, dlatego często korzysta się z podzbioru tego języka. Podobnie jak w przypadku debiandocsgml istnieje szereg narzędzi umożliwiających konwersję plików napisanych w tym formacie do innych, np.: HTML i PDF. Poniżej znajduje się przykład FAQ zapisanych w formacie docbook.

Listing B.2: Plik w formacie DocBook

```
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
<article class="faq" id="index">

<artheader>
<title>Linux-RAID FAQ</title>

<author>
<firstname>Gregory</firstname>
<surname>Leblanc</surname>
</author>

<abstract>
<para>This is a FAQ for the Linux-RAID mailing list, hosted on
vger.kernel.org. vger.rutgers.edu is gone, so don't bother
looking for it. It's intended as a supplement to the existing
Linux-RAID HOWTO, to cover questions that keep occurring on the
mailing list. PLEASE read this document before your post to the
list.</para>
</abstract>
</artheader>

<qandaset>
<qandadiv>
<title>General</title>

<qandaentry>
<question>
<para>Where can I find archives for the linux-raid mailing
list?</para>
</question>
<answer>
<para>My favorite archives are at <ulink
url="http://www.geocrawler.com/lists/3/Linux/57/0/">http://www.geocrawler.com/lists/3/
Linux/57/0/</ulink>.</para>
<para>Other archives are available at <ulink
url="http://marc.theaimsgroup.com/?l=linux-raid&r=1&w=2">http://marc.
theaimsgroup.com/?l=linux-raid&r=1&w=2</ulink></para>
<para>Another archive site is <ulink
url="http://www.mail-archive.com/linux-raid@vger.rutgers.edu/">http://www.mail-archive.
com/linux-raid@vger.rutgers.edu/</ulink></para>
</answer>
</qandaentry>

</qandadiv>
</qandaset>
</article>
```

B.3. wml

Pliki *wml* stanowią szablony zawierające strony HTML wraz ze skryptami umożliwiającymi wstawianie dynamicznych elementów. Pliki te można również zintegrować z pakietem *gettext* umieszczając w nich odpowiednie znaczniki - *<gettext>*:

Listing B.3: Plik w formacie WML

```
#use wml::debian::common_tags

<define-tag debinternational whitespace=delete>
  <gettext>Debian&nbsp;International</gettext>
</define-tag>
<define-tag partners whitespace=delete>
  <gettext>Partners</gettext>
</define-tag>
<define-tag debianweeklynews whitespace=delete>
  <gettext>Debian Weekly News</gettext>
</define-tag>
<define-tag weeklynews whitespace=delete>
  <gettext>Weekly&nbsp;News</gettext>
</define-tag>

<HTML>
  <TITLE><debinternational/></TITLE>
  <BODY>
    To jest <debianweeklynews/>
  </BODY>
</HTML>
```


Bibliografia

- [SJP01] *Słownik języka polskiego*, Wydawnicwo Naukowe PWN, <http://sjp.pwn.pl/>.
- [WIK01] Wikipedia, *Bitext*, <http://en.wikipedia.org/wiki/Bitext>.
- [WIK02] Wikipedia, *Machine translation*,
http://en.wikipedia.org/wiki/Machine_translation.
- [WIK03] Wikipedia, *Computer-assisted translation*,
http://en.wikipedia.org/wiki/Computer-assisted_translation.
- [WIK04] Wikimedia, *Wikipedia Machine Translation Project*,
http://meta.wikimedia.org/wiki/Wikipedia_Machine_Translation_Project.
- [NMPT] Łukasz Degurski, *Prezentacja konkordancera z analizą morfologiczną*,
lista dyskusyjna NMPT, MIM UW.
- [JHU01] John Hutchins, *Machine translation publications*,
<http://ourworld.compuserve.com/homepages/wjhutchins/>.
- [JHU02] John Hutchins, *The development and use of machine translation systems and computer-based translation tools*, International Symposium on Machine Translation and Computer Language Information Processing, 26-28 June 1999, Beijing, China.
- [OCR01] Olivia Craciunescu, *Machine Translation and Computer-Assisted Translation*,
<http://accurapid.com/journal/29computers.htm>.
- [GGA01] Gerald Gazdar, *Machine translation: early/mid history*, course web pages,
<http://www.cogs.susx.ac.uk/research/nlp/gazdar/teach/nlp/nlpnode164.html>.
- [BRI01] Encyclopedia BRITANICA, *Weaver, Warren*,
<http://concise.britannica.com/ebc/article-9382454/Warren-Weaver>.
- [TER01] Terminotix web pages,
<http://www.terminotix.com/eng/index.htm>.
- [GET01] (gettext) History,
[http://docsrv.sco.com:8457/cgi-bin/info2html?\(gettext\)History](http://docsrv.sco.com:8457/cgi-bin/info2html?(gettext)History).
- [JAV01] *JAVA TECHNOLOGY: THE EARLY YEARS*, Sun Developer Network,
<http://java.sun.com/features/1998/05/birthday.html>.
- [JAV02] *To 1.1 – And Beyond!*, Kathy Walrath i Mary Campione,
<http://journals.ecs.soton.ac.uk/java/tutorial/post1.0/index.html>.

- [XLI01] Witryna internetowa OASIS,
<http://www.oasis-open.org/cover/SavourelI18NExcerpts.html>.
- [POA01] Man pages for po4a, <http://po4a.alioth.debian.org/pl/man7/po4a.7.html>.
- [KHE01] Karl Heinz Freigang, *Automation of Translation: Past, Presence, and Future*, <http://www.fti.uab.es/tradumatica/revista/num0/articles/khfreigang/central.htm>.
- [ALP01] ALPAC report, *Language and Machines: Computers in Translation and Linguistics (1966)*, The National Academy Press.
- [RFC01] *RFC 2229: A Dictionary Server Protocol*,
<http://www.faqs.org/rfcs/rfc2229.html>.
- [MOR01] Morphix-NLP, <http://morphix-nlp.berlios.de/>.
- [KNR01] Knorpora, http://sslmit.unibo.it/~baroni/welcome_to_knorpora.html.
- [COL01] Tomasz Okniński, *Narzędzia przetwarzania tekstów w języku Java*,
Praca magisterska (2005).
- [GNI01] Grzegorz Niksiński, *Programy Wspomagające Pracę Tłumacza*,
Praca magisterska (2003).