2. *Королева О. А.* Ирония в «малой» прозе Джерома К. Джерома и английская литературная традиция : автореф. дис. … канд. филол. наук. Н. Новгород, 2006.

3. *Мирзоева Л. Ю.* Категориальные особенности оценки и их представление в современной лингвистике // Вестник Карагандинского университета. Сер. : Филология. 2012. №2. С. 4—9.

4. *Слепцова М. А.* Ирония как косвенный речевой акт отрицательной оценки : автореф. дис. … канд. филол. наук. СПб., 2008.

5. *Трипольская Т. А.* Эмотивно-оценочный дискурс: когнитивный и прагматический аспекты. Новосибирск, 1999.

6. *Гоголь Н. В.* Мертвые души. М., 1972.

7. *Gogol N.* Die toten Seelen. Berlin, 1978.

Об авторе

Елена Васильевна Булатая — асп., Балтийский федеральный университет им. И. Канта, Калининград.
E-mail: bulataya87@mail.ru

About the author

Elena Bulataya, PhD student, I. Kant Baltic Federal University, Kaliningrad.
E-mail: bulataya87@mail. ru

УДК 81’255

Р. Гайда

ЯЗЫК ПРОГРАММИРОВАНИЯ И ПЕРЕВОД

Рассмотрены некоторые вопросы перевода технических текстов, в частности, в области программирования. Внимание уделяется проблеме перевода текстов IT-сферы с учетом фактора многозначности некоторых англоязычных терминов. Особую сложность представляет их адаптация к системе принимающего языка. От переводчиков, работающих в сфере информационных технологий, требуется высокий уровень компетентности в области информатики. Только это гарантирует правильный выбор стратегий перевода, направленных на прагматическую адаптацию текста.

The article deals with some issues of translating technical, in particular, programming texts. The author pays special attention to the polysemy of certain English IT terms. An important problem is adapting such terms to the system of the target language. Translators working in IT are required to have extensive expertise in informatics. Only this will guarantee the choice of a translation strategy adequate for the task of pragmatic adaptation.

Ключевые слова: язык программирования, перевод кода, способы и приемы перевода, прагматическая адаптация, англоязычные заимствования, IT-терминология.

Key words: programming language, translation of code, translation methods and techniques, pragmatic adaptation, English borrowings, IT terminology.

В повседневной жизни мы постоянно сталкиваемся с разного рода текстами, которые переведены с других языков. Например, работая с компьютером, пользователь имеет дело с результатом технического перевода различных программ и операционных систем. Покупая бытовую технику зарубежного производства, мы одновременно получаем точную инструкцию, переведенную с языка оригинала. Технический перевод играет также важную роль в бизнесе, поскольку успешной международной компании необходим качественный перевод договоров, спецификаций, отчетов технических служб, стандартизирующей документации и т. п.

Технический перевод — это переводческая деятельность, связанная с изучением особенностей и закономерностей технических текстов, которая направлена на передачу «средствами одного языка технической информации, выраженной средствами другого языка» [7, с. 191]. Л. Л. Нелюбин под этим термином подразумевает «перевод специальных текстов (документов) научно-технического характера, осуществляемый специфическими методами, отличными от перевода художественного произведения» [6, с. 117—118]. Характерные признаки данного типа перевода — «его информативность (содержательность), логичность (строгая последовательность, четкая связь между основной идеей и деталями), точность и объективность и вытекающие из этих особенностей ясность и понятность» [4, с. 110]. На процесс перевода текстов научно-технического стиля значительно влияет специальная лексика, а именно термины. Слова могут употребляться исключительно в рамках данного стиля или иметь специальное, наряду с общеупотребительным, значение. Термин, по мнению В. Н. Комиссарова, должен точно указывать на реальные явления и объекты, иметь конкретное толкование, «указанное его определением, во всех случаях его употребления в любом тексте, чтобы пользующимся термином не надо было каждый раз решать, в каком из возможных значений он здесь употреблен» [4, с. 111].

Язык программирования — один из наименее изученных объектов в области технического перевода. Перевод таких текстов — довольно сложная сфера творческой деятельности, где профессиональное выполнение работы гарантирует только качественно подготовленный специалист [14, с. 132]. Существуют разные методы технического перевода. Выбор того или иного способа в большей степени зависит от желаний заказчика и его предпочтений. Так как даже переводчик, знающий язык оригинала в совершенстве и обладающий нужными сведениями, не в состоянии перевести технический текст достоверно, возникает необходимость в редакции такого перевода. Поэтому в теории перевода был разработан процесс, который состоит из четырех этапов:

— перевод исходного текста переводчиком;

— редакция, которую делает сам переводчик или другой редактор;

— верификация, во время которой внимание уделяется соответствию с оригиналом;

— стилистическая редакция [12, с. 37].

М. Милковский в своей статье предлагает руководствоваться следующими принципами редактирования технического текста:

— принцип понятности лексики;
— принцип семантической верности;
— принцип бритвы Оккама;
— принцип синтаксической верности;
— принцип проверки адекватности;
— принцип избегания макаронизмов;
— принцип выбора терминологической базы [12, с. 45].

Впрочем, как утверждает автор, указанный список не является бесспорным и законченным. Переводчик должен применять эти правила в соответствии с характером текста, который подлежит переводу.

В данном контексте актуальной сферой исследований представляются лингвистические основы информатики, в том числе создание искусственных языков, например, языков программирования, веб-разработки.

Естественный язык не может быть в полной мере формализован для коммуникации в компьютерных средах. В связи с этим в научной литературе, посвященной лингвистическим основам информатики, говорится о необходимости разработки для этой цели специальных искусственных языков, что входит в сферу интересов лингвистики. Современный словарь определяет искусственные языки как знаковые системы, создаваемые для использования в тех областях науки и техники, где применение естественного языка ограничено, менее эффективно или невозможно [5, с. 1004]. Как отмечает Е. Соснина [9, с. 29], любой искусственный язык по сравнению с естественным всегда ограничен (по словарю, синтаксису, семантике слов) и служит для решения определенных задач.

Существует следующая классификация этих языков:

1. Неспециализированные языки общего назначения (например, эсперанто, волапюк);

2. Специализированные языки различного назначения (например, символические языки наук (математика, логика, химия, физика)) [9, с. 29].

В настоящее время существенную роль играют языки человеко-машинного (компьютерного) общения и реализации компьютерных и информационных технологий (языки программирования, операционных систем, информационных систем и т. п.). Лингвист, в том числе лингвист-переводчик, должен иметь представление о том, какие бывают языки, какова их структурная организация, как создаются искусственные языки и как их анализировать и переводить. Язык программирования как лингвистический объект имеет свою структуру. «Первый низший уровень — *символьный*, его элементы алфавита — буквы, спецсимволы (по аналогии с естественным языком — графематический уровень). Второй уровень — это уровень слов, например, зарезервированных слов, выражений (в естественном языке — это лексический уровень). Третий уровень — *операторный* (командный), где синтаксические конструкции имеют повелительный характер (в естественном языке — аналог синтаксического уровня), и последний — уровень *программы*,

всегда являющейся синтаксически и семантически законченной последовательностью предписаний-команд. Программа — это структурно строгий текст, записанный по формально заданным правилам искусственного языка программирования» [9, с. 30]. На примере апплета (апплет (англ. applet от application) — это существенный компонент ПО, функционирующий в контексте другого, полновесного приложения, предназначенный для одной узкой задачи и не работающий в отрыве от основного приложения [15, с. 20; 16, с. 19]) попытаемся рассмотреть все три уровня, описанные выше.

Символьный уровень начинается фразой:


```
import javax.swing.JApplet;
public class Aplet extends JApplet {
```

Следующий этап — лексический уровень:

```
public Aplet () { // Конструктор апплета }
public void init () { // Инициализация апплета… }
public void start () { // Запуск апплета… }
publicvoidstop () { // Приостановление апплета… }}
gDC. drawString (“Первый апплет”, 100, 50);
<param name=”текст1” value=”Анна”>
<param name=”текст2” value=”Иван”>
</applet>
import javax.swing.JApplet;
public class Aplet extends JApplet { public void init() { String tek1 = getParameter(“текст1”);
String tek2 = getParameter(“текст2”);
```

Последний уровень — командный:

```
public void mouseClicked(MouseEventevt) { int button = evt. getButton();
switch(button) { case MouseEvent. BUTTON1: tekst = “Нажать на кнопку 1, “;break;
case MouseEvent.BUTTON2: tekst = “Нажать на кнопку 2,“;break;
case MouseEvent.BUTTON3: tekst = “Нажать на кнопку 3, “;break;
```

Следует отметить, что в структуре языка программирования важную роль играют кавычки. С точки зрения общей теории перевода все, что находится в кавычках, следует перевести. В языке программирования кавычками обозначаются строковые типы, например, дефиниции переменных или инструкции, которые пользователь программы увидит на экране. Поэтому программисты и переводчики текстов IT-сферы используют два типа кавычек — машинописные двойные кавычки (") и апостроф ('). В некоторых языках программирования используются оба типа кавычек, они играют сходную роль, но их применение не идентично. Так, в языке C++ апостроф используется только для обозначения одного символа (например, 'a'). Машинописные двойные кавычки применяются в случае нескольких знаков, то есть "абв и г". В языке PHP апостроф и двойные кавычки применяются в случае использования

строковой переменной. Разница заключается в том, что переменные в двойных кавычках заменяются конкретными значениями, а в апострофе — нет.

Объектом нашего дальнейшего рассмотрения послужат книги *Изучаем Java* [8] и *Паттерны проектирования* [10].

В них упомянуты паттерны проектирования, то есть документированный многими программистами опыт разработки объектноориентированных систем. Речь идет об описании задач, часто возникающих в работе программиста, а также принципов их решения таким образом, чтобы можно было использовать данное решение много раз, ничего не изобретая заново. Однако это не готовые решения, которые можно непосредственно трансформировать в код. Они лишь показывают взаимодействия и отношения между классами и объектами, не определяя при этом их конечной структуры. Паттерны представляют общее, абстрактное описание задач объектноориентированного проектирования, которые можно (часто даже следует) модифицировать и использовать в конкретных архитектурных ситуациях.

Ниже приведены в качестве примеров предложения на русском языке, в которых присутствуют названия паттернов проектирования, взятые из английского языка

Паттерн Наблюдатель чрезвычайно полезен и принадлежит к числу наиболее часто используемых паттернов [10, с. 71].

Начнем, как обычно, с описания паттерна Декоратор [10, с. 123].

Строго говоря, Простая Фабрика не является паттерном проектирования, скорее это идиома программирования [10, с. 149].

Паттерн Фабричный метод определяет интерфейс создания объекта, но позволяет субклассам выбрать класс создаваемого экземпляра [10, с. 166].

Макрокоманды — простое расширение паттерна Команда, позволяющее выполнять цепочки из нескольких команд [10, с. 258].

В паттерне Фасад подсистема остается открытой для непосредственного использования [10, с. 285].

Паттерн Состояние управляет изменением поведения объекта при изменении его внутреннего состояния [10, с. 430].

Паттерн Мост позволяет изменять реализацию и абстракцию, для чего они размещаются в двух разных иерархиях классов [10, с. 629].

Паттерн Одиночка направлен на создание уникальных объектов, существующих только в одном экземпляре [10, с. 199].

Следующей проблемой служит тот факт, что при переводе любого пользовательского интерфейса переводчику приходится сталкиваться с многозначностью некоторых терминов. Обратимся в данной связи к ряду фрагментов из книги К. Сьерра и Б. Бейтса *Изучаем Java* [8]:

User interface contain: ***buttons***, ***flags***, ***boxes***, *etc.*

В это окно вы будете помещать все элементы интерфейса: ***кнопки***, ***флажки***, ***поля ввода*** *и т.п.* [8, с. 384].

Перевод первого слова не вызывает у нас никаких сомнений, так как большинство двуязычных словарей дает такой же вариант [13, с. 476; 17, с. 92, 266, 273; 18].

Вторая лексема намного интереснее с точки зрения перевода. *Современный толковый словарь* Т.Ф. Ефремовой объясняет ее как *уменьш. к сущ. флаг, отт. маленький флаг, используемый при сигнализации или для украшения* [3, с. 907]. Е.Ю. Ваулина в своем труде *Информатика. Толковый словарь* фиксирует данное слово как «элемент окна диалога для включения или выключения указанного в его названии параметра (в системах с графическим пользовательским интерфейсом)» [1, с. 413]. Заметим, что ни один из доступных на рынке двуязычных словарей (англо-русский или англо-польский) не учитывает нового значения данной лексемы.

Рассмотрим еще один пример:

JFrame со строкой меню и двумя ***виджетами****: кнопкой и переключателем* [8, с. 384].

Англо-русский и англо-польский словари дают буквальный перевод *виджет*, а в русском и польском толковых словарях такого слова нет. Это слово представляет собой буквальный перевод английского термина *widget*, обозначающего основной элемент интерфейса, его компонент, элемент управления, который позволяет пользователю работать с компьютерной программой. Все три выражения — *элемент интерфейса, компонент, элемент управления* – одновременно являются синонимами подразумеваемого термина и могут быть использованы в переводе без утраты эквивалентности.

Таким образом, недостаток существующих словарей, как уже отмечалось выше, — отсутствие информации о том, что данные слова функционируют также в контексте программирования.

Как пишет Н.И. Герасимова, «переводчику следует различать собственно термины и названия программных продуктов IT-сферы. Тот факт, что названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм, требует их сохранения в исходной форме (.NET, C++, VisualBasic, Java, MFC, Windows и др.). В сфере информационных технологий терминологическая система проявляет большую подвижность в связи с быстрым обновлением программного продукта и появлением новых средств его номинации в английском языке. Необходимость адаптации англоязычных заимствований к системе русского языка, мы полагаем, делает чрезвычайно актуальным исследование средств адаптации, которые используют переводчики при работе с англоязычными заимствованиями IT-терминосистемы» [1, с. 26].

Вполне очевидно, что каждая программа в той или иной степени обрабатывает различные данные и сохраняет результат этой работы, давая, тем самым, лингвистам возможность рассматривать лексику, связанную со структурой данных и сохранением объектов. Таким образом, знание шаблонов проектирования — неотъемлемая часть работы лингвиста-переводчика с любым языком программирования. Подобные шаблоны фиксируют проверенные способы решения проблем, часто появляющихся во время разработки программного обеспечения. Все это свидетельствует о том, что переводчик должен быть особо внимательным, когда работает с текстом, посвященным тематике программирования.

Подводя итог вышесказанному, следует констатировать, что главная цель настоящего исследования заключается в попытке актуализации слов и выражений, связанных со структурой объектно ориентированного языка и паттернами проектирования. Разумеется, в рамках данной работы автор смог уделить внимание лишь отдельным аспектам проблемы, которая нуждается в дальнейшем глубоком изучении и открывает перед лингвистами-переводчиками возможности всесторонней разработки анализируемой лексики в связи с постоянным появлением новых версий уже существующих языков и, соответственно, расширением терминологической базы. Важно, что настоящая статья может стать определенным стимулом к созданию словаря англо-русско-польских терминов в области программирования.

Список литературы

1. *Ваулина Е.Ю.* Информатика. Толковый словарь. М., 2005.

2. *Герасимова Н.И.* Прагматическая адаптация IT-текстов информационного дискурса при переводе. Тюмень, 2015.

3. *Ефремова Т.Ф.* Новый словарь русского языка. Толково-словообразовательный : в 2 т. М., 2001.

4. *Комиссаров В.Н.* Теория перевода (лингвистические аспекты). М., 1990.

5. *Большой* толковый словарь русского языка / под ред. С.А. Кузнецова. СПб., 2008.

6. *Нелюбин Л.Л.* Толковый переводческий словарь. М., 2003.

7. *Раренко М.Б.* Основные понятия переводоведения (Отечественный опыт). Терминологический словарь-справочник. М., 2010.

8. *Сьерра К., Бейтс Б.* Изучаем Java. М., 2012.

9. *Соснина Е.П.* Введение в прикладную лингвистику. Ульяновск, 2012.

10. *Фримен Э., Бейтс Б., Сьерра К.* Паттерны проектирования. М., 2011.

11. *Chomsky N.* Syntactic structures. Berlin, 2002.

12. *Milowski M.* Przekład tekstów informatycznych na język polski. Warszawa, 2012.

13. *Prowit M., Szarski J.* Большой русско-польский политехнический словарь. Warszawa, 1968.

14. *Voellnagel A.* Jak nie tłumaczyć tekstów technicznych. Warszawa, 1974.

15. *Angielsko-polski* słownik informatyczny, WNT. Warszawa, 2004.

16. *Polsko-angielski* słownik informatyczny z wymową, WNT. Warszawa, 2013.

17. *Modern* English-Russian Dictionary / под ред. В.К. Мюллер. М., 1998.

18. *Русско-английский* технический словарь (онлайн версия). URL: http://www.classes.ru/dictionary-russian-english-tech-term-36874.htm (дата обращения: 11.11. 2015).

Об авторе

Радослав Гайда — канд. филол. наук, адъюнкт, Педагогический университет, Краков, Польша.
E-mail: radgajda@wp.pl

About the author

Dr Radoslaw Gajda, Adjunct Professor, Pedagogical University, Krakow, Poland.
E-mail: radgajda@wp.pl