МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Гуманитарно-педагогический институт
(наименование института полностью)

Кафедра «Теория и практика перевода»
(наименование кафедры)

45.03.02 Лингвистика
(код и наименование направления подготовки, специальности)

Перевод и переводоведение
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему Стратегии передачи типологических характеристик текстов в сфере-IT при переводе с английского языка на русский

Студент Л. А. Осипова _____
(И.О. Фамилия) (личная подпись)

Руководитель С. П. Анохина _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.филол.н., доцент С. М. Вопияшина _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____»_____20_____г.

Тольятти 2019

АННОТАЦИЯ

Актуальность данной работы обусловлена активным развитием информационных технологий и повышением интереса к данной сфере, а также необходимостью правильного перевода статей научно-технического стиля в сфере IT.

Объектом исследования являются англоязычные научно-технические тексты, посвящённые IT-сфере, с таких ресурсов как Hakernoon, Medium, Habr, Alispart, Tonsky.me, Keycdn, Ecom.

Предметом исследования являются типологические характеристики на уровне лексики и грамматики научно-технических текстов в аспекте их перевода.

Цель данного исследования – определить стратегии передачи типологических характеристик на уровне лексики и грамматики с английского языка на русский.

В соответствии с поставленной целью в работе обозначены следующие **задачи**: 1) рассмотреть научно-технический стиль; 2) описать существующие методики анализа научно-технического стиля; 4) проанализировать научно-технические тексты сферы IT; 5) выявить их типологические характеристики; 6) выявить стратегии перевода типологических характеристик.

Структура. Работа состоит и введения, двух глав, заключения и списка используемой литературы.

В **первой главе** рассматриваются научно-технический стиль, особенности научно-технического стиля, выявляются типологические характеристики статей сферы IT, также производится обзор основных особенностей перевода научно-технических текстов. Во **второй главе** проводится анализ статей (в объёме 300 058 знаков), посвящённых IT-сфере с выявлением основных стратегий передачи лексико-грамматических типологических характеристик.

Список используемой литературы включает 45 научных источников, три из которых иноязычные. **Общий объем** работы составляет 42 страницы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время активно развиваются информационные технологии. IT-сфера очень обширна и разнообразна, и включает в себя достаточно большое количество областей исследования.

Информационные технологии направленны на обработку и хранение информации, передачу данных. В большинстве случаев в основе IT-технологий лежит программирование, которое помогает создавать программы, программные обеспечения, мобильные приложения и многое другое.

Программирование, в свою очередь, основывается на использование языков программирования для написания текста программ, который называется код.

Существует огромное количество языков программирования, которые, как и человеческие языки могут меняться, устаревать и даже умирать. Поэтому, как и в лингвистике в данной сфере есть тенденция к постоянному изменению и развитию, что приводит к быстрому устареванию информации. Таким образом, на сегодняшний день особенно актуальны различного рода интернет ресурсы, как например онлайн-лекции, статьи, вебинары, подкасты и онлайн-курсы доступных не только для специалистов, но и всех желающих повысить свой уровень знаний в данной сфере.

Наиболее распространенным жанром научно-технического стиля в сфере-IT являются статьи. Статьи пишутся непосредственно программистами для таких же специалистов, с целью поделиться опытом в каком-либо вопросе и помочь другим найти решение проблемы.

На сегодняшний день существует необходимость правильного перевода данного типа статей, с учётом всех ее особенностей.

Актуальность данной работы обусловлена активным развитием IT-технологий, в частности область программирования. Дополнительным фактором данной работы является необходимость правильного перевода статей научно-технического стиля.

Объектом исследования являются англоязычные научно-технические

тексты, посвящённые IT-сфере, с таких ресурсов как Hakernoon, Medium, Habr, Alispart, Tonsky.me, Keycdn, Ecom.

Предметом исследования являются типологические характеристики на уровне лексики и грамматики научно-технических текстов в аспекте их перевода.

Цель данного исследования – определить стратегии передачи типологических характеристик на уровне лексики и грамматики в процессе перевода научно-технических текстов с английского языка на русский.

В соответствии с поставленной целью в работе обозначены следующие **задачи**:

1) рассмотреть научно-технический стиль;
2) изучить характерные черты научно-технического стиля;
3) описать типологические характеристики статей научно-технического стиля сферы IT;
4) выявить способы сохранения типологических характеристик при переводе и описать их.

Для решения задач были использованы следующие **методы исследования**: 1) методы анализа и синтеза, с помощью которых был собран и обобщен теоретический материал по исследуемой теме; 2) метод сплошной выборки; 3) метод лингвостилистического анализа, с помощью которого были выявлены типологические характеристики; 4) метод трансформационного анализ выявления комплекса переводческих трансформаций в переводе и объяснения причин и механизмов трансформации исходной единицы.

Теоретической базой работы послужили научные исследования таких ученых, как Нечаева Т. А., Занина А. Л., Попова Т.Г., Лату М. Н., Валиева А. В.

Практическая значимость бакалаврского исследования заключается в выявлении стратегий передачи типологических характеристик в процессе перевода на русский язык, а также в возможности использования данных, полученных в результате исследования, в практических целях.

Материалом исследования послужили англоязычные тексты IT-

сферы, посвященные теме программирования объемом 101 058 знаков, 25 000 из них были переведены.

Апробация результатов исследования. Основные положения и выводы бакалаврской работы были апробированы на Всероссийской студенческой научно-практической конференции «МОЛОДЁЖЬ. НАУКА. ОБЩЕСТВО», в Тольяттинском Государственном Университете (опубликована статья в сборнике научных трудов: «Молодежь. Наука. Общество»: Всероссийская студенческая научно-практическая междисциплинарная конференция. Тольятти: Изд-во ТГУ, 2018. С. 449-451), а также на конференции в ТГУ «Студенческие дни науки».

Структура и основное содержание работы.

Логика исследования и последовательность решения поставленных з а-дач обусловили структуру работы, которая состоит из введения, двух глав, заключения и списка используемой литературы.

Во **введении** обосновывается выбор темы и ее актуальность, определяются объект и предмет исследования, характеризуются цели, задачи, методы, определяется практическая значимость исследования.

В **первой главе** освещаются основные характеристики научно-технического стиля, приводятся классификации жанров научно-технического стиля, даются определения основных понятий, выявляются типологические характеристики статей научно-технического стиля, а также освещается теория, посвящённая переводу текстов научно-технического стиля.

Во **второй главе** проводится анализ примеров из статей научно-технического стиля IT-сферы, выявляются стратегии перевода типологических характеристик статей научно-технического стиля.

В **заключении** обобщаются результаты исследования.

Список используемой литературы состоит из 45 научных источников, два из которых на иностранном языке.

ГЛАВА I. НАУЧНЫЙ СТИЛЬ В СИСТЕМЕ СТИЛЕЙ АНГЛИЙСКОГО ЯЗЫКА

1.1 Общая характеристика научно-технического стиля

Наряду с официально-деловым, публицистическим и научным, научный стиль относится к книжным стилям. В рамках научного стиля различаются научно-технические, научно-гуманитарные, естественнонаучные, научно-популярные, учебные тексты [29, С. 71].

Научный стиль используется в области исследований, теорий и доказательств, для объяснения научных фактов, гипотез. Принято различать академическую научную литературу (научные статьи, диссертации), предназначенную для подготовленного читателя (специалиста) в определенной области; научно-учебную (учебники, учебные пособия), предназначенную для будущего специалиста, и научно-популярную, предназначенную для непрофессионала. Техническая литература – это литература, относящаяся к области техники и производства (каталоги изделий, инструкции по эксплуатации, обслуживанию и ремонту, каталоги деталей, патенты и т. п.) [3, С. 73].

Следовательно, научно-технический текст сочетает в себе признаки научного и технического стилей. Таким образом, в текстах данного стиля обязательно должна присутствовать последовательность и строгость в изложении [29, С. 72]. Выводы должны вытекать из содержания и не противоречить друг другу. Что касается повествования, то оно направлено либо от частного к общему, либо от общего к частному. Помимо этого, текст должен быть ясным и точным, чтобы реципиенту была понятна и доступна информация, во избежание расхождения между означаемым и определением [37]. Необходимо отметить, что в текстах научно-технического стиля, по сравнению с научным и научно-популярным существует высокая частотность использования терминов, а также слов в прямом значении, что снижает вероятность возникновения двусмысленности и недопонимания смысла высказывания [42, С. 200].

Таким образом, в научно-техническом стиле должны преобладать:

1. Профессиональная лексика (термины, сокращения)
2. Стремления к объективности, логичности и ясности
3. Нормативный стиль речи

Как уже было сказано ранее, на лексическом уровне в текстах научно-технического стиля преобладают термины, как общенаучные, так и узкоспециальные. Это связано с тем, что термин несет логическую информацию большого объема [19]. В нормативном научно-техническом стиле, идеальный термин, в отличие от обычной лексики, является однозначным, в составе определенной терминологии и не нуждающимся в контексте, а также может употребляться изолировано [42, С. 108].

Отсюда следует, что норма научно-технического стиля не предполагает синонимии терминов, во избежание двусмысленности и искажения смысла высказывания. Тем не менее, синонимия присутствует в текстах научно-технического стиля. Основными причинами являются активное развитие различных областей научного знания [39] и отражение динамики развития научно-технической мысли [4, С. 43]. Помимо этого, одна из возможных причин состоит в том, что, специалисты, как правило, заимствуют термины из другого языка, вследствие, возможного отсутствия аналога в родном языке, однако термин остаётся немотивированным и безассоциативным, что приводит к необходимости создания синонима в родном языка [34]. Например, транслитерированный термин скринсейвер также представлен вариантами его перевода хранитель экрана и экранная заставка; транслитерированный термин конвертация (файла) имеет синоним преобразование [5, С. 85].

Более того, в тексте допускаются общеупотребительные слова (contractor, documentation, list, equipment), которые приобретают свои специальные значения непосредственно в тексте. Причина этого в том что, термины функционирую в определенной терминологии, поэтому когда общеупотребительные и многозначные слова попадают в такую же систему они приобретают однозначность [12, С. 87].

В текстах научно-технического стиля употребляются простые термины (flare, tank), термины-словосочетания (superheavy lifts, storage tanks), термины, образованные с помощью словообразовательных аффиксов. Например, с помощью префиксации (*un*-equal, *in*-put); суффиксации: существительные (heat-*er*, cens-*or*), глаголы (to energ-*ize*, theor-*ise)*, прилагательные (period-*ic*, measur-*able*), наречия (direct-*ly*) [14].

Существует тенденция к использованию много компонентных терминов (adjacent base pitch error, lever gun welding head, jacquard circular knitting machine), которые образованы без какого-либо грамматического оформления и представляют собой последовательность слов [12, С. 88].

Сокращения являются обязательным атрибутом научно-технического стиля. В большинстве случаев, их фиксируют в специализированных источниках [44]. Следует заметить, что в таком направлении, как IT-сфера процесс возникновения сокращений привел к перенасыщению сокращениями, понятными только для специалистов узкого круга, а также к отсутствию таких сокращений в специализированных словарях [31, С. 100].

Более того, появляются такие явления как аббревиатурная полисемия и синонимия, акронимия.

Аббревиатурная полисемия означает существование нескольких вариантов расшифровки сокращения, которые относятся к различным явлениям и понятиям (многозначная аббревиатура). А именно, **BN** имеет значение*: 1) Bridge Number; 2) Backbone Node; 3) BECN Cell* [38].

Аббревиатурная синонимия касается сокращений, в которых начальная форма и значение аббревиатуры остается неизменным, но меняется один ее термин, который образует синонимичную расшифровку [31]. Например, **CAS,** аббревиатура которую можно расшифровать как *Column Address Strobe* и *Column Access Strobe [7]*.

Явление акронимии имеет отношение к сокращениям, схожим по звучанию и написанию с обычным словом (**AIR** - *Additive Increase Rate*)[39].

Принято считать, что в нормативном научно-техническом тексте не допускается использование фразеологизмов, идиом и метафор, несмотря на это, авторы достаточно часто использую выразительные средства в своих работах [24]. Например, Bull market, to bell the cat, dead halt. Так как, метафорическая коммуникация – хороший способ соотнести сложные процессы с простыми [29, С. 73].

Что касается грамматического аспекта текстов научно-технического стиля, то необходимо отметить, что абстрактные понятия или природные явления, зачастую построены по именному типу (*The yard is a unit of linear measure* equal to 3 feet)[12, С. 88]. Помимо этого, используются двусоставные безличные(it was decided, it was mentioned) и неопределенно-личные предложения, где например неопределенно-личное местоимение - one выступает, в качестве слова заменителя (The way of solving the problem is more common than that *one*), клише (From above mentioned..., Fargoing possibilities, Favourable solution)[18], использование инфинитивных (the properties *to be expected*, Brakes are used *to slow or to stop* the car) или герундиальных (There is *no denying* the seriousness of the problem) оборотов [22]. Это говорит о том, что авторы текстов должны стремиться к объективному изложению фактов.

В свою очередь, норма научно-технического стиля предполагает использование пассивных форм (Welding processes *are classified…*), отглагольных прилагательных с предлогами (to be attendant on, to be conducive to, to be destructive of), допускается замена наречий предложно именными сочетаниями (accurately − with accuracy) [26, С. 90].

Так как авторы текстов данного стиля стремятся к краткости и сжатости изложения, наиболее распространены эллиптические конструкции (a remote crane = a remote-operated crane), возможны случаи пропуска определенных артиклей (_*First uranium mine* in the region was…). Артикли обычно пропускают перед названиями научных областей или при описании конкретных деталей (…in such fields as civil *engineering, telecommunication,…*/Traps

have long-live parts, *valve and seat*) [14]. Для логического выделения отдельных смысловых элементов в английской технической литературе часто используется нарушение прямого порядка слов (инверсия)[20] (*In Table I are listed* the data obtained).

Наряду с этим, такие атрибуты, как логичность и последовательность изложения, подразумевают использование причинно-следственных союзов и связок (since, therefore, it follows that)[26]. Еще одна особенность — это использование множественного числа у неисчисляемых существительных (fats, greases, sands, wools)[12, С. 91].

Таким образом, принято считать, что научно-технический стиль – это сухой лишенный какой-либо образности и эмоциональности текст. Ведь цель такого текста – это донести научную информацию, поэтому текст обладает точностью, логичностью, а также в нем должна присутствовать последовательность. Изложение информации не должно противоречить само себе, быть понятным и доступным, также не допускается выражение авторской точки зрения или каких-либо комментариев относительно изложенных фактов. Все стремиться к тому чтобы, информация была беспристрастной и объективной. Тем не менее, в текстах присутствует синонимия, возможно употребление выразительных средств, метафоризация терминов.

Также, в таких текстах распространены клише, номинативная структура, безличные, и эллиптические конструкции. И конечно же в этом стиле процент научных и узкопрофессиональных терминов намного выше, чем в текстах научного или научно-популярного стилей, что также является особенностью этого стиля.

1.2 Обзор типологических характеристик научно-технических текстов IT-сферы в английском языке

В предыдущем параграфе мы рассмотрели, характеристику научно-технического стиля. Мы пришли к выводу, о том, что, несмотря на суще-

ствующие формальные характеристики, присущие данному стилю, авторы, в большинстве случаев отступают от нормы.

В этом разделе мы хотели бы рассмотреть особенности научно-технического стиля в рамках IT-сферы и рассмотреть типологические характеристики текстов IT-сферы.

В основном статьи IT-сферы посвящены языкам программирования. Это связано с тем, что в основе техники, программного обеспечения, интернета, а также иногда и телефонии лежит программирование. Поэтому очень важной областью исследования для многих специалистов являются языки программирования.

Что касается языковой составляющей данных статей, то на лексическом уровне остаётся неизменным наличие терминов [1]. Например, использование общенаучных терминов, которые являются важной составляющей любой статьи, касающейся программирования: *method, version, function, array, symbol, class, source, repository[60]* и т.д. Широко распространены многого компонентные термины и термины словосочетания, например,: *type-driven approach[61], conditional types feature, single-page application[72], redux-related code[61], user-concerned acceptance criteria.* Также, в статьях IT-сферы используются узкоспециальные термины: *code, code review, framework, refactoring, back-end, front-end, implementation* [60].

Присутствует тенденция к метафоризации общеупотребительных слов, в качестве терминов [2]. Например, *to build* a program, в значении создать программу, *cookies* в значении небольших фрагментов данных, *to wrap* object, что означает объект обтекания. Особенно, эта тенденция наблюдается у глаголов, например, *to break* в значении дать сбой, *to write* в значении создавать, *to call* a function в значении вызов функции, а также может использоваться в значении называть функцию, метод или массив, что говорит и полисемии термина. Помимо этого, метафоризация используется в названиях визуальных компонентов, например, *carousel, breadcrumbs, dropdowns, grid system.* Метафоризация в этих случаях делает термин мотивированным, и лишает его

многозначности, что способствует лучшему пониманию смысла высказывания.

Также неотъемлемой частью научно-технических статей IT-сферы являются сокращения [6]. Например, в статье можно встретить такие сокращения как: *var и val[61]* от variables и value, *const* – constant, *parametrs[61]* – params, *proc* и *proc* – procedure и proceedings. Широкое употребление данных сокращений объясняется тем, что написание полного слова занимает много времени и памяти в коде. Так, как программисты стараются оптимизировать и излишне не нагружать код, то стремление к краткости, можно наблюдать непосредственно в статьях.

Аббревиатуры, широко распространены в данной сфере, например, *API - Application programming interface, URL - Uniform Resource Locator[65], TTI - Time To Interactive, DOM - Document Object Model*. Поэтому следует отметить, что в текстах проявляется явления синонимии (*ILMI – Interim Local Management Interface или Integrated Local Managment Interface)*, полисемии (*EFS - End of frame sequence или Error Free Seconds*), которые уже были описаны ранее[5].

Необходимо обратить внимание на структуру научно-технических статей в сфере IT [8]. В анализируемых нами статьях, заголовок обладает не меньшей метафоричностью, чем у терминов [25]. В большинстве случаев они очень яркие, выразительные и привлекающие внимание, что делает их приближенными к научно-популярному стилю. Например, “*Death of the for Loop”[70], “From URL to Interactive”[65], “After two years with TypeScript – was it worth it?”[74], “JSX is syntactic sugar”[60]*. Не менее важным и структурирующим элементом являются подзаголовки. Зачастую, подзаголовки могут быть еще более выразительными. Например, может быть использована идиома *First things first – setup [48]*, иногда может использоваться ирония *Mappy Map Map[61]*, использование вопроса в качестве подзаголовка это частое явление в статьях, посвященных IT, *Why should we place limits on how we write software?/ What’s happening under the hood?[65]*

Возвращаясь к структуре статей научно-технического стиля в сфере IT, следует обратить внимание на способ организации текста [13]. Практически всегда, вначале статьи и после краткого вступления следует скриншот кода, который затем, автор рассматривает более подробно, также в статье могут появляться выноски, для того, чтобы сделать акцент на более важной и необходимой информации [27]. Названия функций, циклов, методом и т.п. практически всегда выделяются, либо при помощи изменённого шрифта, либо при помощи выделения цветом. Помимо этого, сохраняются специальные символы (0, _, #) и операторы (-, +, =, \, *), которые являются частью кода и остаются неизменными.

Что касается грамматических особенностей научно-технических статей IT-сферы, следует отметить в первую очередь распространение пассивного залога (function *that has been completely separated* from our cat business logic!/ , …a CompleteUser *can now be definitively expressed/ …it's well tested…*). Также употребляются безличные конструкции (*Is it possible to code/ …it will produce the same output 100% of the time. / It was not until functional components showed up…) [10]*. Широко распространены герундиальные и инфинитивные обороты (but tell TypeScript to use our soon *to be created function* DOMcreateElement for that.[74]/ TypeScript *needs to know how to compile*…/ we can even create our own methods *using* recursion!/ what do browsers do when *tearing down* a page./ There's steam *rising* from the hood[65].)

Одной из особенностей статей в научно-техническом стиле IT-сферы, является использование условных предложений [11].

- *If you think about it, our whole industry depends on our faith in a handful of "black boxes" few of us fully understand: browsers[65].*
- *If the element is a string, we create a regular DOM Node. For that we call document.createElement[60].*

Также существует тенденция использования вопросно-ответной формы изложения, которая снова приближает статью данной сферы к научно-популярному [15].

- *Ever wanted to write to the DOM directly, but gave up because it's so unwieldy? Come on, document.createElement is probably easy, but you have to do a ton of calls to the DOM API to get what you can achieve so easily by writing HTML[60].*
- *If you think that types are "the game changer", what would you say about interfaces? Interfaces help you write much better code, because they finally allow you to define contracts between objects[74].*

Помимо этого, достаточно часто авторы используют вводные конструкции, что тоже не свойственно научно-техническому стилю [16].

- *You can go to the source, examine the code and extract valuable information (because your code is always clean and fast to read)*...[65]

Авторы используют эллиптические добавления, которые относятся к публицистическому стилю [17].

- *They simply may be incorrect or outdated. Or missing [65].*

Ролевые акты в предложениях наряду с вопросно-ответной формой изложения, являются одной из особенностей статей IT сферы [18].

- *Shall we take a look?[72]*
- *What if I told you there was a way to do this in Ruby?[72]*
- *Of course you can use forgiving "any" type, which means "I don't care about the type of that thing, it could be a number[60].*

Помимо этого, авторы используют: повторения и восклицательные предложения [21].

1. Восклицательные предложения [43] :

- *Let's step through it!/ Let's try![72]*
- *That's all there is! But the API is a lot nicer![60]*

2. Повторения [44] :

- *Not that I advocate that, no no no, just that it does have some use in enabling my particularly…[72]*

3. Междометия [33] :

- *Oh that's where we can have some fun [72].*
- *Well what does send take? Oh yes.*
- *Oh jeez. What the heck just happened?[65]*

Таким образом, приведенные примеры раскрывают особенности, грамматических конструкций. Использование авторами таких средств как добавление, вопросно-ответная форма изложения говорит о том, что авторы стремятся к ясности. Данные стилистические приёмы помогают достигнуть понимания за счет образности, создания автором атмосферы общения и взаимодействия с читателем. Несмотря на то что авторы статей отдаляются от научно-технического стиля в сторону научно-популярного, это не мешает достигать коммуникативной цели в передаче опыта относительно информационных процессов, связанных с программированием. Напротив, образность и выразительные средства помогают понять, как работает та или иная функция, как она поведет себя в той или иной ситуации, и как решать трудные задачи возникающие на этапе написания кода.

Таким образом, анализ научно-технических статей сферы IT показал, что авторы статей достаточно свободно и творчески подходят к написанию статей. Несмотря на то, что весь спектр терминологии сохраняется в текстах данного стиля, авторы могут использовать средства выразительности, чтобы сделать текст привлекательнее и интереснее. Например, использовать сравнения (*OpenStruct can act like a Hash [70], it's quite the trip down the ol' black magic rabbit hole [72])*. Иногда, могут использовать иронию (*…just that it does have some use in enabling my particularly hacky REPL Ruby habits which should never see the light of production, amen [72]*), или фразеологизмы (*I was completely dumbstruck [61]*). Фактически это нарушение норм научно-технического стиля, которое может стать проблемой понимания текстов данной сферы. Тем не менее, использование данных средств, добавляет красочности и выразительности, что помогает лучше понять изложенный материал. Помимо использования выразительных средств стоит отметить, что в научно-технических статьях сферы-IT существует тенденция к использованию со-

кращений и аббревиатур. Особенности заключаются также и в особой организации текста, где, изложение информации происходит от общего к частному. Большое внимание уделяется заголовкам, сноскам, дополнительной иллюстративной информации. Вышеперечисленные типологические характеристики упрощают восприятие информации и предотвращают проблему неправильного понимания текста.

1.3 Особенности перевода текстов научно-технического стиля

В предыдущем параграфе мы рассмотрели, как формируется научно-технический стиль, а также изучили типологические характеристики научно-технических текстов IT-сферы в английском языке. При переводе статей необходимо учитывать эти особенности.

Перевод любого текста с исходного языка (ИЯ) на другой предполагает необходимость учитывать принадлежность оригинала к определенному функциональному стилю с принятыми для него языковыми средствами, которые не всегда совпадают с характерными для аналогичного стиля языковыми средствами в переводящем языке (ПЯ) [41]. Так в частности, специфика научно-технического стиля определяется его информативностью, логичностью, объективностью и, как следствие, ясностью и понятностью [23]. Это, в свою очередь, определяет выбор языковых средств, где, прежде всего, следует отметить ведущую роль терминологии [9]. Слова могут употребляться исключительно в рамках данного стиля или иметь специальное, наряду с общеупотребительным, значение [10]. Поэтому перевод научной и технической литературы надо рассматривать как с языковедческих, так и научных и технических позиций, с преобладанием первых при исследовании общеязыковых вопросов и вторых - при рассмотрении узкой терминологии [35].

Перевод научно-технических текстов должен отвечать следующим требованиям: эквивалентность, адекватность, информативность, логичность и четкость изложения [20]. Чтобы перевод научно-технического текста был адекватным и эквивалентным, т.е. качественным, переводчику необходимы

общие и специфические навыки, умения и следующие знания: теоретические (лексические единицы, грамматические правила, словообразование); практические (виды переводческих трансформаций и соответствий: транслитерация, калькирование, замены, перестановки, добавления, опущения, способы описательного и антонимического перевода); экстралингвистические знания (владение достаточной информацией для перевода специализированного текста), недостаточной информацией для перевода специализированного текста), необходимые в процессе переложения текста и построения осмысленных и адекватных предложений на языке перевода [20].

Что касается непосредственно текстов IT-технологий, относящихся к сфере программирования, то существуют определённые стратегии при переводе типологических характеристик научно-технических текстов в сфере-IT [7].

При переводе термины (customization, refactoring, framework) и лексику общенаучного описания (library, specification, component, object) следует передавать с помощью эквивалентов (hardware – жесткий диск, code review – инспекция кода, algorithm – алгоритм, language – язык) или вариантными соответствиями (web service–веб-сервис, веб-служба, web-site – веб-сайт, интернет-сайт), также как идиомы (core dump – дамп ядра, storage dump trap – ловушка для распечатки памяти) и фразеологизмы (sound loop – звуковая петля, external memory – внешняя память, infrared port - инфракрасный порт) все лексические средства выразительности следует передавать, сохраняя особенности данных средств [24]. В некоторых случаях стратегия перевода позволяет вольный или антонимический перевод, главное передать функцию воздействия [3].

Как было отмечено ранее, авторы научно-технических статей стремятся передать информацию более понятно и доступно, поэтому зачастую при описании технологий авторы используют вводные конструкции (For extra credit we could curry takeFirst and use it to create other functions (more on currying in another article)), вопросно-ответные формы изложения (If a number, why the

hell you just added string “id_” in front of it? TypeScript code looks a lot like other popular typed languages and there is a chance, that you will receive better and more precise code review of your commits [69]), условные предложения (It’s fun but when would we ever use things like this in actual code? When it’s well tested, commented, documented, and becomes an understood idiom of your code base [31]; If a number, why the hell you just added string “id_” in front of it? TypeScript code looks a lot like other popular typed languages and there is a chance, that you will receive better and more precise code review of your commits[74].), которые создают связь между автором и адресатом. Этот прием обязателен к сохранению, так как он определяет стилистику текста [36]. Данные приёмы, зачастую имеют простую форму и понятное содержание, поэтому допускается дословный перевод.

Помимо этого, авторы используют повторения (Not that I advocate that, no no no, just that it does have some use in enabling my particularly…[72]) и восклицательные предложения (That’s all there is! But the API is a lot nicer![61]), которые создают атмосферу живого общения и приближают, таким образом, текст к научно-популярному, данные типологические характеристики необходимо сохранять в тексте. Это поможет создать адекватный перевод и сохранить прагматику текста.

Выводы по первой главе

Несмотря на то, что указанные особенности приближают статьи научно-технического стиля IT-сферы к научно-популярному, считать такие статьи научно-популярными было бы неправильно. Прежде всего, необходимо задуматься о реципиенте данных статей. Реципиентом является специалист, но так как IT-сфера активно развивается, и в настоящее время большинство людей выбирают эту профессию, вне зависимости от опыта и знаний, то уровень варьируется от базового до специалиста. Поэтому термины, в отличие от научно-популярного стиля не объясняются. Аббревиатуры, используемые в научно-техническом стиле не общеизвестные, а узкоспециальные, также, как

и сокращения. Даже несмотря на то что существует явление метафоричности терминов, которое достаточно сильно развито, это лишь делает статьи научно-технического стиля еще более интересными и понятными. Несомненно, использование разговорных элементов, таких как повторения, междометия приближают научно-технический стиль к научно-популярному, но это необходимо, чтобы не только информация была понятная специалистам базового уровня, но и интересна более опытным специалистам.

Помимо этого, автор статей – это IT-специалист, который хочет поделиться опытом по какому-либо вопросу, поэтому уровень в этом случае, также варьируется.

Что касается организации текста, авторы большое внимание уделяют заголовкам, которые в большинстве очень яркие и бросающиеся в глаза, а также подзаголовкам. Повествование ведётся от общего к частному, этот принцип чаще всего используется в статья научно-технического стиля в сфере IT. Иллюстративный материал представлен в виде скриншота кода, который автор детально разбирает. Комментарии, эллиптические добавления и вопросно-ответная форма изложения создают эффект общения и взаимодействия между читателем и автором, что делает статьи еще более привлекательными, понятными и интересными для изучения.

Следовательно, анализ лексических единиц показал, что большинство переводчиков, применяют методы модуляции или прибегают к дословному переводу. Это связано с тем, что авторы статей используют наравне с терминологией разговорную лексику. Особую сложность составляют фразеологизмы идиомы, которые переводятся с помощью подбора эквивалентов, также, как и большинство терминов.

Анализ грамматических конструкций показал, что осложненные длинные предложения разбивают на более короткие и простые, которые лучше воспринимают широкой аудиторий. А также вопросно-ответная форма изложения, комментарии переводят методом калькирования или, с помощью модуляции.

Специфика перевода научно-технических текстов, направлена на адаптацию теста для читателя и сохранение прагматического потенциала текстов.

ГЛАВА II. СТРАТЕГИИ ПЕРЕДАЧИ ТИПОЛОГИЧЕСКИХ ХАРАКТЕРИСТИК ТЕКСТОВ В СФЕРЕ-IT ПРИ ПЕРЕВОДЕ С АНГЛИЙСКОГО ЯЗЫКА НА РУССКИЙ

2.1 Передача типологических характеристик на уровне лексики научного-технического текста в сфере IT

В первой главе мы рассмотрели, как формируется научно-технический стиль, изучили типологические характеристики и стратегии перевода, на которые необходимо обратить внимание переводчику при работе с данным типом текстов. Во второй главе мы более подробно рассмотрим лексический и грамматический аспект перевода, что поможет сформировать целостную картину и понять, как достигнуть адекватного перевода.

В качестве материала были проанализированы статьи из таких интернет журналов, как: Tonsky.me, Keycdn, Ecom, Medium, Hackernoon, Alistapart, Habr.

Статьи данных интернет источников связаны с научно-технической тематикой, цель автора статьи осветить некоторые особенности в работе с языками программирования, представить новые способы решения определённых задач, рассказать о тенденциях развития IT-сферы. Следовательно, такой текст насыщен терминами, как общенаучными, так и узко специальными, как уже было указано в первой главе.

Сопоставительный анализ статей с сайтов, приведенных выше показал, что переводчики придерживаются стратегии подбора эквивалентов при передачи общенаучных терминов (Таблица 1).

Таблица 1 – Примеры передачи общенаучных терминов (эквиваленты)

1. So the use of the slash in the value allows you to create *curved corners* that are not symmetrical. [66]	Использование slash-а при указании значений позволит вам создавать несимметричные *скругленные углы*. [54]

2. The *two values* that are often forgotten, however, are bolder and lighter. [66]	Но два возможных *значения* часто забывают — bolder и lighter. [54]
3. Let's get back to the above code snippet and try to understand how the code executes inside the JavaScript *engine.* [58]	Давайте теперь вернемся к фрагменту кода выше и попробуем понять, как *движок* JavaScript его выполняет. [52]

Также существует тенденция передавать с помощью соответствий общенаучные термины, которые уже стали общеизвестными и не нуждаются в пояснении или подборе эквивалента (Таблица 2).

Таблица 2 – Примеры передачи общенаучных терминов (соответствия)

1. Notice that the demo includes 12 nested 'box' *elements* with different font-weight values, including "bolder" and "lighter" so you can see how these affect the weight of the text in different inheritance contexts. [66]	В демке используется 12 вложенных box-*элементов* с различными значениями font-weight включая bolder и lighter, так что эффект от применения этих ключевых слов очевиден. [54]
2. The :first-letter *pseudo-element* allows you to style the first letter of an element, letting you do a drop-cap effect which has been common in print for many years. [66]	*Псевдо-элемент* :first-letter позволяет вам стилизовать первую букву элемент (примечание переводчика — привет, капитан!), например, красиво выделить её как в печатных книгах много лет назад. [54]
3. Learn *syntax* of the new programming language. [63]	Освоить *синтаксис* языка. [57]

Что касается узкоспециальных терминов, также существует тенденция передавать такой тип терминов с помощью соответствий. В первую очередь, это связано с тем, что реципиентом являются специалисты, которые хорошо знакомы с тематикой данных статей. Помимо этого, стоит отметить, что жанр статьи позволяет переводчикам использовать профессионализмы, так как этот приём придаёт статьям неофициальный оттенок и не приравнивает их к научным работам (Таблица 3).

Таблица 3 – Примеры передачи узкоспециальных терминов (соответствия)

1. On the right is the resultant *assembly*	Справа — получившийся *ассемблерный*

code. [69]	*код*. [47]
2. I've focused mainly on *recurrent neural networks*. [64]	До этого я больше фокусировался на *рекуррентных нейронных сетях.[49]*
3. The *Internationalization* API is native and pretty well supported. [62]	Но, API *интернационализации* – встроенная возможность и отлично поддерживается браузерами. [48]
4. That's why we built Figma, a collaborative *interface* design tool that we recently launched, as *a* browser-based cloud service. [71]	Вот почему мы создали Figma, командный инструмент работы над дизайном *интерфейсов,* как облачный сервис, распространяемый в виде веб-приложения. [55]
5. The JavaScript *interpreter* will assign variable declarations a default value of undefined during what's called the "Creation" phase. [68]	*Интерпретатор* JavaScript назначает объявленным переменным значение undefined во время фазы называемой "Создание". [51]
6. The basic *logarithm* for quicksort then becomes very simple. [74]	Основной *логарифм* для быстрой сортировки довольно прост. [57]
7. *Compile* it into WebAssembly (giving you a binary .wasm file). [69]	*Скомпилируйте* этот код в WebAssembly (это даст вам бинарный wasm-файл). [47]
8. WAT exists because we humans generally have a hard time making sense of straight *binary code*. [69]	WAT существует потому, что мы, люди, обычно испытываем трудности с пониманием чистого *бинарного кода*. [47]
9. Because they're so tightly tied together, this *pattern* is suitable for one-offs, but it's still not really unit-testable or reusable. [67]	Поскольку функции сильно связаны друг с другом, этот *паттерн* пригоден для каких-то единичных случаев, но он по-прежнему не пригоден для тестирования или переиспользования. [50]
10. Believe it or not, the following is *valid code*: [66]	Верите или нет, но следующий *код валиден*: [54]

Очень часто в текстах оригинала можно встретить сокращения. Авторы статей зачастую могут заменять некоторые длинные термины, хорошо известными и укрепившимися в сообществе IT-специалистов, сокращениями. Однако в русском языке такой тенденции не наблюдается, поэтому при передаче данных сокращений переводчики расшифровывают и предают сокращения, подходящим эквивалентом (Таблица 4).

Таблица 4 – Примеры передачи сокращений (эквиваленты)

1. If you've never seen that, it might seem a little confusing, so here's the explanation from the *spec*: [66]	Такой подход немного обескураживает в первый раз. Вот объяснение из *спецификации: [54]*
2. Projects that can now be integrated into websites or *apps. [69]*	Эти проекты теперь могут быть интегрированы в сайты или *приложения*. [47]
3. In each case, I'm going to *err* on the side of bare-bones simplicity. [69]	Во всяком случае, я собираюсь сделать всё на самом базовом уровне, чтобы избежать *ошибок*. [47]
4. Banking apps, *e-commerce websites*, stock exchange platforms, we interact with money daily. [62]	Банковские приложения, *интернет-магазины*, платформы для фондовых бирж, мы встречаемся с деньгами каждый день. [48]

Относительно сокращений, которые используются авторами в качестве примера и названия того или иного компонента кода, переводчики оставляют данные компоненты в том виде, как они выглядят в оригинале (Таблица 5).

Таблица 5 – Примеры передачи сокращений названий (сохранение оригинала)

1. First, let's compare *var and let. [68]*	Для начала, давайте сравним *var и let. [51]*
2. However, the only difference is that once you've assigned a value to a variable using *const*, you can't reassign it to a new value. [68]	Однако есть одно отличие: если вы однажды присвоили значение используя *const*, вы не сможете его изменить на другое. [51]
3. The variable declaration, *var message* whose scope is the *function hoist()*, is hoisted to the top of the function. [68]	Объявление переменной *var message*, область видимости которой заканчивается в функции *hoist()*, «поднимается» наверх функции. [51]

Стоит отметить, что аналогичный подход стоит применять для передачи названий свойств (Таблица 6).

Таблица 6 – Примеры передачи названий свойств (сохранение оригинала)

1. The *border-radius property* can use "slash" syntax. [66]	В *свойстве border-radius* можно использовать slash-синтаксис. [54]

2. The *font-weight property* accepts relative keywords. [66]	*Свойство font-weight* может принимать «относительные» значения. [54]
3. *The vertical-align* property is defined on the input element. [66]	*Свойство vertical-align* установлено у поля ввода input. [54]

Функций (Таблица 7):

Таблица 7 - Примеры передачи названий функций (сохранение оригинала)

1. The *second()* function finishes, so it's popped off the stack. [58]	После этого мы вызываем функцию *second()*, поэтому она помещается на вершину стека. [52]
2. When the *processImage()* function completes, it's removed from the stack. [58]	Когда функция *processImage()* выполнена она удаляется из стека. [52]
3. So you see, we have to wait until the function (such as *processImage() or networkRequest( ))* has finished. [58]	Как вы видите, нам нужно ждать пока функция (такие как *processImage()* или *networkRequest()*) завершится. [52]
4. For example, the previous animation uses a timing function of *linear*. [66]	В вышеупомянутом примере используется функция с линейной зависимостью от времени — *linear*. [54]

Названий языков программирования (Таблица 8):

Таблица 8 – Пример передачи названий языков программирования (сохранение оригинала)

1. Now that we have a basic idea about the call stack, and how the synchronous *JavaScript* works, let's get back to the asynchronous *JavaScript*. [58]	Теперь, когда мы имеем общее представление о стеке вызовов и о том, как работает синхронный *JavaScript*, давайте вернемся к асинхронному *JavaScript*. [52]
2. Searching for "*Vue.js*" trends gives us a curve that starts in 2014 and then climbs from there so that looks good, and there is definitely no other "Vue.js" that could be adding unwanted results. [73]	Поиск по трендам *«Vue.js»* даёт нам кривую, которая начинается в 2014 году, а затем поднимается выше, и безусловно нет какого-либо другого «Vue.js», который мог бы привести к неожиданным результатам. [53]
3. I started reading about the syntax of *Go* and found an awesome beginners	Так я оказался на пороге изучения другого языка программирование — *Go*. [57]

language tour on their official website. [63]	
4. Right after I learnt the syntax of *Perl*, I started contributing to open source projects. [63]	Сразу после того, как я освоил синтаксис *Perl*, я начал вносить вклад в различные opensource проекты. [57]
5. ES2015 (or ES6) introduced two new ways to create variables, let and const. [68]	ES2015 (или ES6) представил нам 2 новых способа для создания переменных, let и const. [51]

Названий программ (Таблица 9):

Таблица 9 - Примеры передачи названий программ (сохранение оригинала)

1. In this experiment, I used the user's camera as a controller for playing a small JavaScript clone of *Mortal Kombat 3*. [64]	В этом эксперименте я использовал пользовательскую камеру, чтобы играть в небольшую JavaScript-копию игры *Mortal Kombat 3*. [49]
2. When we set out to build *Figma* we knew it would be a challenge. [71]	Когда мы решились создать *Figma*, мы знали, что это будет серьезный вызов. [55]
3. We recorded 5 videos with *QuickTime* on my *MacBook Pro,* each of which contains 2-4 punches and 2-4 kicks. [64]	Мы записали 5 видео на *QuickTime* с моего *MacBook Pro*, каждое из которых содержало 2–4 удара рукой и 2–4 удара ногой. [49]

Авторы очень часто используют аббревиатуры в своих статьях. Как уже было отмечено ранее, статьи научно-технического стиля сферы IT, направлены на передачу информации для специалистов. Таким образом, существует тенденция сохранять аббревиатуры в том виде, как они выглядят в оригинале (Таблица 10).

Таблица 10 – Примеры передачи аббревиатур (сохранение оригинала)

1. The *MDN* generator is the only one I've found that does this. [66]	Из всех найденных таких генераторов только *MDN* это умеет. [54]
2. *CSS* has lots of fancy text layout algorithms but no way to customize the algorithms or to read back the result of what the browser did so the text layout algorithm can be used as part of another algorithm. [71]	*CSS* предоставляет ряд прекрасных алгоритмов рендеринга/раскладки текста, но не дает настроить их, или получить результат того, что сделал браузер, дабы использовать алгоритм раскладки текста как часть другого алгоритма. [55]

3. The call stack has a *LIFO* structure which means that the items can be added or removed from the top of the stack only. [58]	Структура *LIFO* означает, что элементы могут добавляться и удаляться только с вершины стека. [52]
4. But, it has changed so much along the way to its official *MVP* release in February, that when you first get started learning about it, it's easy to drown in a sea of details. [69]	Но он настолько сильно изменился по пути к официальному *MVP* в феврале, что когда вы впервые узнаете о нём, то легко можете утонуть в море деталей. [47]
5. I won't go into much detail about either of these, but you do need to know at least a little bit about the *WAT* file in order to follow the next steps. [69]	Я не буду разбирать результат компиляции подробно, но вы должны знать хотя бы немного о *WAT*-файле, чтобы двигаться дальше. [47]
6. A trained *CNN*, similar to *VGG*-16, which recognizes a large set of images would have hidden layers that have discovered many features from its training set. [64]	Обученная *CNN*, аналогичная *VGG*-16, распознающая большой набор изображений, будет иметь скрытые слои, которые уже умеют распознавать многие характеристики изображений на основании предыдущего тренировочного сета. [49]
7. *HTML* and *SVG* contain a lot of baggage and are often much slower than the 2D canvas *API* due to *DOM* access. [71]	*HTML* и *SVG* тащат за собой много лишнего багажа и часто работают намного медленнее, чем 2D canvas *API* из-за использования *DOM*. [55]

Стоит отметить, что *общеупотребительные слова*, в текстах *приобретают в системе терминов IT сферы узкое специализированное значение*. Так, глагол to build, приобретает значения создать, например, программу, приложение или веб-сайт. Глагол to declare приобретает значение объявлять, например, функцию, переменную, класс. Или глагол to run в системе терминов IT приобретает значение запускать, например, команду, программу, приложение (Таблица 11).

Таблица 11 – Примеры передачи общеупотребительных слов

1. Now, I really wanted *to build* something using Perl so that I could apply my knowledge of the language and practice various concepts of the language. [63]	После этого появилось огромное желание *собрать* что-нибудь, используя Perl, чтобы применить и обкатать свои новообретённые навыки. [57]
2. Above we try to access a variable outside of the function it was *declared*. [68]	Выше мы попытались получить доступ к переменной снаружи функции, в которой она была *объявлена*. [51]
3. *To run* the above command, you may first need to install ffmpeg on your computer. [64]	Прежде, чем *запустить* эту команду, установите ffmpeg на ваш компьютер. [49]
4. When a *call* to first() is encountered, it's pushed to the top of the stack. [58]	Когда встречается *вызов* функции first(), он так же добавляется на вершину стека. [52]

Подводя итог выше сказанному, хочется отметить, что IT-специалисты образуют своё комьюнити и имеют определённые площадки, где делятся опытом. Это обусловлено тем, что языки программирования, так же, как и обычные языки, имеют тенденции к устареванию и развитию. Комьюнити, в этом случае, помогает оставаться в курсе всех изменений и инноваций. Поэтому авторы, в своих статьях могут использовать жаргонизмы (сбилдить – to build, пофиксить – to fix, хакнуть – to hack, протестить – to test, прод – production, сборка – assembly [46]) и профессионализмы (дебажить – debugging, рефакторинг – refactoring, компиляция – compilation, имплементация – implementation, верификация – verifying, валидность – validation, рекурсия – recursion [45]), которые будут понятны таким же специалистам.

Таким образом, основными приёмами при переводе типологических характеристик на уровне лексики является перенос оригинала, подбор эквивалентов, передача с помощью соответствий. Особенно данные стратегии касаются узкоспециальных терминов. Поэтому, чтобы передать эффект неформальности, переводчики используют соответствия вместо эквивалентов.

Жанр статьи позволяет использовать данный метод, так как он делает термины, короче и информативнее, чем их эквиваленты.

Что касается общенаучных терминов, то сопоставительный анализ статей, показал, что переводчики подбирают подходящий эквивалент. Сокращения также следует передавать с помощью эквивалентов или вариантными соответствиями. Но, что касается сокращений, используемых автором, в качестве элемента кода, то необходимо сохранить оригинал. Такой же способ, касается наименований функций, переменных, объектов, массивов следует сохранять оригинал. Обычно такие элементы легко распознать в тексте, так как они имеют пояснительное слово или изменённый шрифт. Подобным образом, необходимо поступать при передаче названий языков программирования и программ.

2.2 Передача типологических характеристик на уровне грамматики научного-технического текста в сфере IT

В предыдущем параграфе мы рассмотрели, каких стратегий придерживаются переводчики при работе с типологическими характеристиками на уровне лексики. Мы выяснили, что основными стратегиями являются перенос оригинала, подбор эквивалентов, метод транслитерации.

Что касается, грамматического аспекта, то проведенный нами анализ статей показал, что авторы чаще всего используют условные предложения, вводные конструкции, ролевые акты, восклицательные предложения и вопросно-ответную форму изложения.

Авторы очень часто используют, *условные предложения* в своих статьях. Это связанно с тем, что условные предложения – это предложения, где есть определенное условие, при котором действие произойдет или могло бы произойти/не произойти. Так как цель специалиста поделиться опытом, относительно какой-либо проблемы, данный тип предложений помогает создать определённый алгоритм действия при решении трудной задачи.

В большинстве случаев, данный тип предложений не составляет особых сложностей при переводе, так как в русском языке существует аналог.

Поэтому при переводе таких предложений возможен дословный перевод (Таблица 12).

Таблица 12– Примеры передачи условных предложений (дословный перевод)

1. *If values are given before* and after the slash, then the values before the slash set the horizontal radius and the values after the slash set the vertical radius. [66]	*Если значения указаны* и «до», и «после» слэша, то значения «до» устанавливают горизонтальный радиус, а значения «после» «вертикальный». [54]
2. *So if we log the declaration* variable, we get undefined. [68]	*Итак, если мы вывели переменную declaration*, то мы получили undefined. [51]
3. *If the variable statement occurs* inside a FunctionDeclaration, the variables are defined with function-local scope in that function. [68]	*Если объявление переменной* происходит внутри объявления функции, переменная определяется в локальной области видимости этой функции. [51]
4. *If the left pointer is less than* or equal to the right pointer, then swap the values at these locations in the array. [74]	*Если левый указатель меньше или* равен правому указателю, поменять значения в этих местах в массиве. [56]
5. *Everything will be much clearer if we step back and look at a list* of the steps involved in implementing WebAssembly in a project. [69]	*Всё станет намного понятнее, если мы сделаем шаг назад и рассмотрим список действий для внедрения* WebAssembly в проект. [47]
6. *If all of this makes sense to you,* then you can skip to the next section. *But if you found yourself* scratching your head a bit and want a more detailed explanation, then continue reading. [69]	*Если вы поняли, о чём идёт речь*, то можете пропустить следующую часть. *Но если вы обнаружили себя* почёсывающим голову и хотите получить подробное объяснение, то продолжайте читать. [47]
7. *If we train the model with only* 600 photos taken in the same environment, with the same people, we'll not be able to achieve a very high level of accuracy. [64]	*Если мы обучим модель только на* 600 фотографиях, сделанных в одном и том же месте с одними и теми же людьми, то аккуратность этой модели будет не высокой. [49]

8. *If we wanted to take that even* further, we could create similar configurations for the success and error handlers. [67]	*Если бы мы хотели пойти еще* дальше, то мы бы создали аналогичные функции для создания обработчиков успеха и ошибок. [50]
9. *If you want to add two monetary* values with different currencies, you need to convert them first. [62]	*Если вам потребуется сложить две* денежные величины, нужно сначала их конвертировать. [48]

Одна из основных конструкций, которая создаёт эффект взаимодействия между автором и реципиентом является вопросно-ответная форма изложения. Он отражает ход мысли и размышления автора в процессе. Данный приём помогает привлечь внимание к теме и идее текста, побуждает читателя к диалогу с автором, вовлекает в процесс поиска решения проблемы вместе с автором. Данный приём обязателен к сохранению, так как он является особенностью статей IT сферы.

Анализ, используемых нами статей показал, что основными методами при переводе таких приёмов и конструкций является дословный перевод (Таблица 13).

Таблица 13 – Примеры передачи вопросно-ответной формы изложения (дословный перевод)

1. Money is nothing more than a numeric value, right? Wrong. [62]	Деньги ведь число, верно? Неверно. [48]
2. Why did I pick this specific layer? Empirically. [49]	Как я выбрал именно этот слой? Эмпирическим путем. [47]
3. Why did I pick 1024 units for the second layer and 1e-6 learning rate? Well, I tried a couple of different options and saw that 1024 and 1e-6 work best. [49]	Почему я выбрал 1024 юнита для второго слоя и 1e-6 для скорости обучения? Я просто попробовал несколько разных настроек и увидел, что 1024и 1e-6 показали лучшие результаты. [47]
4. Let’s jump right to the good stuff – how is Vue doing? Sure, it’s not a competition, but still… let’s add in React and Angular results. [50]	Давайте сразу перейдём к интересному – как дела у Vue? Конечно, это не соревнование, но всё же… давайте добавим результаты по React и Angular. [48]
5. Are the variables declared inside of it	Доступны ли переменные объявленные

accessible outside of it? Turns out, they are. [49]	внутри него за его пределами? Оказывается доступны.(модуляция) [47]
6. We can contribute to an open source project only if we are expert in that language, right? The answer is No. [49]	«Я могу привнести что-то в опенсорс проект только будучи матёрым разработчиком, не так ли?». Отвечаю – Нет.(модуляция) [48]

Как было отмечено ранее, авторы статей, посвящённых языкам программирования, стремятся передать информацию более понятно и доступно, поэтому зачастую при описании устройства или технологии авторы добавляют вводные конструкции, которые, в большинстве случаев содержат важные сведения, дополняющие основное содержание.

Этот прием обязателен к сохранению, так как он является стилистической особенностью IT текстов. Данный приём, зачастую имеет простую форму и понятное содержание, поэтому допускается дословный перевод, если необходимо применяется метод модуляции (Таблица 14).

Таблица 14 – Примеры передачи вводных конструкций (дословный перевод)

1. The outline property is pretty well known due to its ability to help in debugging *(it doesn't affect page flow).* [49]	Свойство outline довольно широко известно из-за использования при дебагинге *(из-за него страница не «расползается»)*. [48]
2. In this case, the animation will run a half time (that is, it will stop halfway through its first iteration). [49]	В этом случае анимация будет проиграна лишь на половину первой итерации. [48]
3. Last year, I got an offer for full-time job from Booking.com and I knew that I'd be working on Perl *(which is their primary language for backend). [49]*	В прошлом году я получил оффер от Booking.com. Я знал, что мне предстоит писать на Perl *(который является основным языком для бэкенд части в Booking). [48]*
4. We have an internal API called IndirectBuffer *(which we are open sourcing here)* that references an external typed array and makes it available to C++. [50]	Мы создали внутренний API «IndirectBuffer» *(заопенсорсили тут),* который позволяет ссылаться на внешний типизированный массив и делает его доступным в C++. [48]

5. Some people just love it, and some think it's nasty. It's a matter of style and preference. *(For the record, the people who think pumpkin flavored beer is nasty are correct). [50]*	Некоторые люди просто любят его, а другие думают, что это противно. Это вопрос стиля и предпочтений. *(Для справки, люди, которые считают, что тыквенное пиво противно, правы). [48]*
6. For our project, we'll be writing a simple function in C++ *(don't worry if you're not familiar with C++, it'll be* extremely *simple). [49]*	Для нашего проекта мы напишем простую функцию на C++ *(не беспокойтесь, если вы не знакомы с этим языком, код будет максимально простым).* [47]
7. The VGG-16 network recognizes 1000 classes of images. It has 16 layers (*we don't count the output and the pooling layers). [49]*	Сеть VGG-16 может распознавать до 1000 классов изображений. Она имеет 16 слоёв *(мы не учитываем выходной и pooling слои). [47]*
8. This allows you to work with integers only, which means safe calculations *(until you hit big numbers)* and great performances**. [**49**]**	Это позволит работать с целыми числами, что означает безопасные вычисления *(пока не доберётесь до больших чисел)* и отличную производительность**. [**47**]**

Восклицательные предложения позволяют автору выразить своё отношение к описываемому или побудить читателя к действию. В примере 1, была использована модуляция, а в примере, 2 переводчик сделал добавление, в остальных примерах был произведен дословный перевод. Можно сделать вывод, что при переводе данных типов предложений основным приёмом является использование дословного перевода, но в зависимости от контекста переводчик, применять и другие стратегии для достижения адекватного перевода (Таблица 15).

Таблица 15 – Примеры передачи восклицательных предложений (дословный перевод)

1. But Wait, There's More! [50]	Но подождите, это ещё не всё! [48]
2. Very colorful! [50]	График выглядит очень колоритно! [48]
3. Finish up and run it! [71]	Заканчивайте и запускайте! [55]
4. Now we can use squarer() as a regular JavaScript function! [49]	Теперь мы можем использовать squarer() как обычную JavaScript-функцию! [47]

5. It works! You're calling a function written in C++! [49]	Работает! Вы вызываете функцию, написанную на C++! [47]
6. Now let us run it in the browser and wire it up together with MK.js! [49]	Теперь давайте запустим её в браузере и подключим к MK.js! [47]

Несмотря на стиль статей, в текстах довольно часто авторы используют риторические вопросы. На риторические вопросы автор отвечает сам, либо хочет, чтобы над вопросом подумали читатели. Зачастую данная конструкция не составляет труда для перевода, поэтому существует тенденция передавать с помощью дословного перевода (Таблица 16).

Таблица 16 – Примеры передачи риторических вопросов (дословный перевод)

1. So what does this actually tell us? [50]	Так что на самом деле он нам говорит? [48]
2. Let's jump right to the good stuff – how is Vue doing? [50]	Давайте сразу перейдём к интересному – как дела у Vue? [48]
3. Now that you understand the difference between var and let, what about const? [49]	Теперь вы понимаете разницу между var и let, что же о const? [47]

Авторы также часто использую риторические обращения, выраженные повествовательными предложениями. Повествовательные предложения могут дать общую информацию перед тем как дать более подробную. Или обратить внимание читателей на важную информацию. При передаче также используется дословный перевод (Таблица 17).

Таблица 17 – Примеры передачи риторических обращений (дословный перевод)

1. Let's discuss them one by one. [49]	Существует несколько аспектов, давайте обсудим их по порядку. [48]
2. Please keep in mind that the setTimeout is not a part of the JavaScript engine, it's a part of something known as web APIs (in browsers) and C/C++ APIs (in node.js). [48]	Пожалуйста, помните, что *setTimeout* не является частью движка JavaScript, это часть так называемого web API(в браузере) и C/C++ APIs (в node.js). [48]

При передаче эллиптических добавлений переводчики иногда опускают данный элемент, так как считают его неуместным. Но, в большинстве

случаев, переводчики сохраняют эту особенность применяя дословный перевод (Таблица 18).

Таблица 18 – Примеры передачи эллиптических добавлений (дословный перевод)

1. That might be the first time in W3C specification history that something is hard to understand. LOL JK. [49]	Возможно, это первый раз в истории спецификации W3C, когда что-то так сложно понять.(опущение) [48]
2. Surprised? Some of you may be thinking “But wait, open source is hard”. [49]	Удивлены? Некоторые из вас подумали – "Погоди, опенсорс – это же сложно. (дословный перевод) [48]
3. Cool, so anytime you want a variable to be immutable, you can declare it with const. Well, not quite. [49]	Отлично, теперь когда вы захотите, чтобы ваша переменная была неизменна вы можете объявить её при помощи const. Или не совсем. (дословный перевод) [47]

Пассивный залог широко употребителен в современном английском языке. Особенно часто он используется в тех случаях, когда нам неизвестно, кто исполняет действие, или этот момент нам не важен. Основными приёмами, которых придерживаются переводчики являются передача с помощью страдательного залога, перевод в активный залог, и передача с помощью глагола с инфинитивом пассивного залога (Таблица 19).

Таблица 19 – Примеры передачи пассивного залога

1. The two values that *are often forgotten*, however, are bolder and lighter. [49]	Но два возможных значения *часто забывают* bolder и lighter.(перевод в активный) [48]
2. The emscripten compiler targets the asm.js JavaScript subset which provides a way to get JavaScript JITs to emit predictable, compact machine code and *is widely supported* in all modern browsers. [50]	Компилятор emscripten нацелен на asm.js, *подмножество JavaScript поддерживаемое* во всех современных браузерах, которое позволяет получить от JIT-компилятора JavaScript предсказуемый, компактный машинный код.(страдательный залог) [48]
3. Once *all this is done*, the array is returned as the result. [50]	Как только *все это сделано*, массив возвращается в качестве результата. (страдательный залог) [48]

4. Also, all three functions *are still cursed* with the need to be kept inside of the route. [49]	Однако, все три функции *по-прежнему глубоко поражены* проблемой необходимости держать их внутри обработчика маршрута. (страдательный залог) [47]
5. The library *was designed* to be immutable and chainable. [49]	Библиотека *сконструирована* так, чтобы все объекты были неизменяемыми и могли складываться в цепочки вызовов. (страдательный залог) [47]
6. When this code *is executed,* a global execution context is created (represented by main()) and pushed to the top of the call stack. [48]	Когда код *начал выполняться*, был создан глобальный контекст выполнения(представленный как *main()*) и добавлен на вершину стека вызовов.(глагола с инфинитивом пассивного залога) [48]

Безличные конструкции при переводе следует переводить с помощью страдательного причастия, модального глагола (мочь) (Таблица 20).

Таблица 20 – Примеры передачи безличных конструкций (модальный глагол мочь)

1. *It works* only on inline or inline-block elements. [49]	*Применимо* только к inline или inline-block элементам. [48]
2. *It can be affected* by text/font settings like line-height and by the size of adjacent inline or inline-block elements. [49]	На свойство *могут влиять* настройки текста или шрифта, такие как line-height и размер смежных inline и inline-block элементов. [48]

Инфинитивные обороты переводят сочетанием: *«чтобы + инфинитив»* (пример 1). Также, передают с помощью инфинитива (примеры 2, 3, 4, 5) (Таблица 21).

Таблица 21 – Примеры передачи инфинитивных оборотов

1. *To really succeed* it would have to provide a high-fidelity editing experience that professionals would accept and would have to work consistently	*Чтобы действительно преуспеть*, необходимо предоставить высокоточный инструмент редактирования, который будет принят профессионалами, а так же будет

everywhere. [50]	работать одинаково хорошо в любом окружении.[48]
2. So the use of the slash in the value *allows you to create* curved corners that are not symmetrical.[49]	Использование slash-а при указании значений *позволит вам создавать* несимметричные скругленные углы. [48]
3. Most border-radius generators do not *allow you to set* these optional values. [49]	Большинство генераторов border-radius *не позволяют устанавливать* дополнительные значения. [48]
4. But before we actually dive into the differences between var, let, and const, there are some prerequisites you need to know first. [49]	Но прежде чем мы углубимся в различия между *var*, *let* и *const*, имеются некоторые темы, которые вам следует узнать в первую очередь. [47]
5. In this case, I chose to use the floor function, but you could just as well use the ceiling function or round function with some slightly different logic. [50]	В данном случае, я решил использовать функцию Math.floor, но вы можете точно также использовать функцию Math.ceil или Math.round с немного другой логикой. [48]

Герундиальные обороты являются особенностью английского языка. При анализе статей нами была выявлена тенденция к передаче герундиального оборота с помощью деепричастного оборота или существительным (Таблица 22).

Таблица 22 – Примеры передачи герундиальных оборотов

1. I'll walk you through the steps to create and implement an extremely simple project, removing complexity wherever possible. [49]	Я покажу вам несколько шагов для реализации простого проекта, исключая сложность, где это возможно.(деепричастный оборот) [47]
2. Training an image classification TensorFlow.js model in Node.js and using it in the browser. [49]	Тренировка классификационной модели TensorFlow.js с Node.js и использование её в браузере.(существительное) [47]
3. Knowing this, we should be able to find a way to build a function which holds some information, and returns a preconfigured function ready to be used by something else. [49]	Зная это, мы должны иметь возможность найти способ создания функции, которая содержит некоторую информацию, и возвращает предварительно сконфигурированную функцию, готовую к использова-

	нию чем-то другим.(деепричастный оборот) [47]
4. When you want to successively perform several operations on a single object, chaining provides an elegant notation and concise syntax. [49]	Если вы хотите последовательно выполнить несколько операций над одним объектом, объединение их в цепочку вызовов обеспечивает элегантную и краткую запись.(существительное) [47]

Анализ перевода типологических характеристик на уровне грамматики показал, что основной стратегией при переводе является дословный перевод.

Это связано с тем, что в текстах, посвящённых IT сфере, переводчикам приходится сталкиваться с элементами публицистического стиля: эллиптические добавления, вопросно-ответная форма изложения, восклицательные предложения, риторические вопросы и предложения, которые имеют простую форму и содержание, поэтому не составляют труда при переводе. Условные предложения, переводчики передают таким же типом приложения. Что касается герундиальных оборотов, то мы выявили тенденцию к передаче с помощью существительного и деепричастного оборота. Инфинитивные конструкции передают с помощью инфинитива или сочетанием «чтобы + инфинитив».

Выводы по второй главе

Таким образом, мы рассмотрели все особенности, которые переводчику важно учитывать при передачи типологических характеристик на уровне лексики и грамматики.

Реципиентом является специалист, но так как IT-сфера активно развивается, и в настоящее время большинство людей выбирают эту профессию, вне зависимости от опыта и знаний, то уровень варьируется от базового до специалиста. Но несмотря на это, термины, в отличие от научно-популярного стиля не объясняются. Аббревиатуры, используемые в научно-техническом стиле не общеизвестные, а узкоспециальные, также, как и сокращения. Даже

несмотря на то что существует явление метафоричности терминов, которое достаточно сильно развито, это не является проблемой данных текстов.

Несомненно, использование стилистических элементов, таких как риторические обращения, вопросно-ответная форма-изложения, восклицательные предложения, которые приближают научно-технический стиль к научно-популярному, но это необходимо, чтобы не только информация была понятная специалистам базового уровня, но и интересна более опытным специалистам.

Таким образом, можно сделать вывод о том, что авторы статей стараются упросить сложную научно-техническую информацию, за счёт данных стилистических элементов.

Следовательно, анализ лексических единиц показал, что аббревиатуры следует сохранять в оригинале, так же, как и названия методов, классов функций, программ, языков программирования. Что касается сокращений то переводчики расшифровывают их и подбирают советующий эквивалент. Но это не касается сокращений, которые являются примерами кода. Их следует также оставлять в оригинале. Узкоспециальные термины, переводчики используют соответствия вместо, его эквивалента, так как это позволяет жанр статьи, а также площадка на которой была размещена данная статья. Что касается общеупотребительных слов, то переводчику важно помнить, что многие слова в терминологической системе IT сферы меняет своё значение. Поэтому важно подбирать эквивалент в соответствии с контекстом.

Анализ перевода грамматических особенностей, показал, основным приёмом является дословный перевод. Несмотря на стиль статей, авторы используют достаточно большое количество элементов присущих публицистическому стилю, такие как вопросно-ответная форма изложения, эллиптические добавления, условные предложения, риторические вопросы. Зачастую эти элементы не составляют труда при переводе. Однако они обязательны для сохранения так как они придают тексту неформальный стиль повествования и создают связь между автором и реципиентом герундиальных оборо-

тов, инфинитивных конструкций, то переводчики применяют общепринятые правила передачи данных конструкций. Таким образом, стратегия передачи типологических характеристик на уровне лексики и грамматики, является совокупность данных приёмов, которые помогут создать правильный перевод. Это поможет передать текст с учётом всех особенностей, а также сохранить прагматический потенциал текста.

ЗАКЛЮЧЕНИЕ

Научно-технический стиль, используется во многих известных журналах, распространен за счет ясности, четкости и точности.

Естественно в текстах такого типа присутствует терминология и общенаучная лексика, существует логичность при изложении материала, точность формулировок.

Формально авторы статей, должны соблюдать правила при написании научно-технических статей. Однако на практике авторы отступают отправил и сухого наполненного фактами текста. Это происходит потому, что сухой научно-технический текст сложен для восприятия. Выразительные средства делают термины мотивированными и легко запоминающимися.

Следовательно, анализ лексических единиц показал, что большинство переводчиков применяют перенос оригинала, транслитерацию и подбор эквивалентов. Перенос оригинала необходим для передачи названий (функции, методы, программы, языки программирования). Использование соответсвий при передаче узкоспециальных терминов вместо соответствующих эквивалентов позволяет сохранить оттенок неофициальности данным статьям. Подбор подходящих эквивалентов используют для передачи, общеупотребительных слов, которые приобретают своё значение в определённой терминологии. Следовательно, стратегиями при переводе типологических характеристик на уровне лексики выражены приёмами, перечисленными выше.

Анализ грамматических конструкций показал, что авторы часто используют условные предложения, вопросно-ответная форма изложения. Несвойственные научно-техническому стилю риторические вопросы и обращения, восклицательные предложения, вводные конструкции. Все перечисленные грамматические конструкции имеют простую форму и содержание, что позволяет переводчикам использовать дословный перевод.

Что касается пассивных и инфинитивных конструкций, герундиальных оборотов, в этом случае переводчики придерживаются общих правил передачи данных типов конструкций.

Проанализировав статьи научно-технического стиля сферы IT, мы пришли к выводу, что перевод такого типа текста несмотря на кажущуюся легкость, имеет свои трудности. Причина кроется в том, что тексты данной сферы направлены на специалиста. Авторы могут использовать жаргонизмы, профессионализмы, узкоспециальные термины, которые не всегда просто найти в словарях, человеку незнакомому с данной сферой. Помимо этого, существует опасность неправильно трактовать некоторые элементы, которые необходимо сохранять в том виде в каком они есть в оригинале. Также переводчик должен учитывать все типологические характеристики на уровне лексики данных текстов и применять стратегии, которые помогут ему добиться адекватного перевода. Также важно помнить, что необходимо сохранять все грамматические типологические характеристики, особенно, что касается вопросно-ответной формы изложения, риторических вопросов, вводных конструкций, которые придают текстам неформальность и создают иллюзию общения.

Из всего вышеизложенного можно сделать вывод, что стратегия пере передачи типологических характеристик на уровне лексики и грамматики, является совокупность данных приёмов, которые помогут создать правильный перевод. Помимо этого, переводчикам необходимо учитывать существование многоязычных IT-коммьюнити, к которым можно обратиться в затруднительных случаях. Это также поможет переводчику сделать правильный и адекватный перевод.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Алексеевичева С. Ю. К определению понятия научного-технического типа текста: // Cyberlenika. 2017. URL: https://cyberleninka.ru/article/v/k-opredeleniyu-ponyatiya-nauchno-populyarnyy-tip-teksta. (Дата обращения: 02.04.2019).
2. Береговская Э. М. Стилистика в подробностях: учебник. М. : Либроком, 2015. - 232 с.
3. Бобылев И. И. Жанры и стратегии перевода научно-технических текстов // Актуальные вопросы современности глазами молодых исследователей. 2018. № 3. С. 17-21.
4. Валиева А. В. О жанровой принадлежности научно-технических текстов и ее влиянии на перевод // Филологические науки. Вопросы теории и практики. 2015. № 11 (53). С. 33-35.
5. Володькова С. А. Проблемы расшифровки английских сокращений в информатике и их перевод на русский язык // Филологические науки. 2008. № 6 (47). С. 99-100.
6. Воронова А. В. Научно-технические тексты как объект функционально-стилистического анализа // Русский и иностранные языки и методика их преподавания. 2016. № 2. С. 7–11.
7. Гайда Р. Язык программирования и перевод // Вестник Балтийского федерального университета им. И. Канта. Филология, педагогика, психология. 2016. № 1. С. 42-48.
8. Гайда С. Актуальные задачи стилистики // Актуальные проблемы стилистики. 2015. № 1. С. 11–22.
9. Гладун Д. Ш. Перевод научных статей: проблема качества // Журналистика, лингвистика, переводоведение и филология. 2016. № 14. С. 138–142.

10. Гришечкина Г. Ю. Способы раскрытия термина в научно-популярном тексте // Вопросы когнитивной лингвистики. 2011. № 2. С. 92–100.

11. Губарева О. Н. Перевод при смешении научно-технического и научно-учебного стилей // Перевод и сопоставительная лингвистика. 2010. № 14. С. 37–45.

12. Давыдов Д. А Лингвистические особенности английских научно-технических текстов // Московский авиационный институт (национальный исследовательский университет). 2016. С. 9-14.

13. Ельчищева Л. Д. Ориентиры, стимулирующие восприятие научно-технического текста и формирование речевого уровня // Научные труды Дальневосточного государственного технического рыбохозяйственного университета. 2007. № 19. С. 468-474.

14. Занина А. Л. Лингвистические особенности технической документации как жанра научного стиля // Вестник научного общества студентов, аспирантов и молодых ученых. 2015. № 3. С. 72-75.

15. Занина М. А. Проблемы понимания метафоры в англоязычном научно-техническом тексте // Вестник Челябинского государственного университета. Филология. Искусствоведение. 2013. № 1 (292). С. 210–214.

16. Ивлиева Е. А. Метафора в компьютерной терминологии испанского языка // Филология. 2011. № 127. С. 154-158.

17. Киселев А. Ю. Адресные стратегии в научно-техническом дискурсе // Языкознание. № 5 2014. С. 762–764.

18. Клёстер А. М. Лексико-семантические особенности научно-технических текстов // Филологические науки. 2017. № 3. С. 41-44.

19. Комарова Л. Н. Отличительные особенности научного стиля // Образование и наука в современных реалиях. 2017. С. 239-241.

20. Коняева Л. А. О некоторых трудностях научно-технического перевода // Перевод и сопоставительная лингвистика. 2015. № 11 С. 50–54.

21. Круглов А. С. Особенности сообщения как элемента коммуникативного взаимодействия в социальных медиа // Наука и мир. 2016. № 2. С. 149–152.

22. Лату М. Н. Когнитивные аспекты образования синонимии в терминологии // Филология. Искусствоведение. Вестник Челябинского государственного университета. 2011. № 24 (239). С. 84–86.

23. Макарич М. В. Особенности логико-семантической структуры английских научно-технических текстов // Вестник Мозырского государственного педагогического университета им. И.П. Шамякина. 2013. № 1. С. 115-120.

24. Минакова Н. А. О языке и стиле научно-технического текста // Русский и иностранные языки и методика их преподавания. 2015. № 4. С. 27–34.

25. Митюкова Е. А. Термины-метафоры информационных технологий в английском языке // Языковая личность и эффективная коммуникация в современном поликультурном мире. 2018. С. 124-131.

26. Михиенко С. А. Стилистика научно-технических текстов в английском языке: особенности перевода // Язык и культура (Новосибирск). 2015. № 19. С. 86-92.

27. Мякишева Е. А. Стилистические особенности научно-технического текста // Гуманитарные научные исследования. 2018. № 5 [Электронный ресурс]. URL: http://human.snauka.ru/2018/05/25006 (дата обращения: 23.09.2018).

28. Мякшин К. А. Явление метафоризации в терминологии (на примере английской фонетической терминологии) // Молодой ученый. –2015. – №7. – С. 490-494.

29. Нечаева Т. А. Выразительные средства языка научных текстов // Технические науки. Известия ЮФу. 2013. № 10 (147). С. 71-77.

30. Переверзева И. В. Прагматическая адаптация эллиптических предложений при переводе // Филологические науки. Вопросы теории и практики. 2015. № 5. С. 139–141.

31. Попова Т. Г. Научно-технический текст в функционально-техническом ракурсе // Вестник РУДН. Языки и специальность. 2007. № 4. 59-63.

32. Попова Т. Г. Параметры научно-технической статьи // Романо-германская филология. 2004. № 11. С. 149-153.

33. Родичева А. А., Зайцева Т.А. Диалогичность и эмоционально-оценочная информация в медицинских научно-технических текстах // Филология. 2015. №8. С. 86–90.

34. Уланович О. И. Функционально-стилистическая вариативность научных текстов // Вестник БДУ. 2012. № 1. 54-58.

35. Хоменко С. А. Основы теории и практики перевода научно-технического текста с английского языка на русский. М. : БНТУ, 2013. 203 с.

36. Хомутова Т. Н. Стратегии научного дискурса // Лингвистика. 2015. №3. С. 15–22.

37. Хомутова Т. Н., Петров С.Г. Научно-технический текст: интегральная модель // Лингвистика. 2016. № 2. С. 37–41.

38. Шапкина Е. В. Особенности перевода научной статьи: аннотация // Лингвистика. 2015. № 2. С. 10–14.

39. Шереметьева С. О. К вопросу об электронных ресурсах профессиональной лексики // Вестник ЮУрГУ. Лингвистика. 2014. № 1. С. 57-61.

40. Шимановская Л.А. Переводческие исследования и их специфика // Лингвистика. 2016. №3. С. 450–456

41. Ясницкая Е. С. Особенности перевода английской научно-технической литературы [Текст] // Актуальные вопросы филологических наук: материалы IV Междунар. науч. конф. (г. Казань, октябрь 2016 г.). — Казань: Бук, 2016. — С. 77-81. — URL https://moluch.ru/conf/phil/archive/232/11114/ (дата обращения: 21.12.2018).

42. Gibbons A. Contemporary Stylistics / A. Gibbons. Edinburgh University Press, 2018 – 288 p.

43. Phadtare A. Scientific and Technical Writing: A Manual of Style / A. Phadtare. - Auris Reference, 2016. – 218 p.

44. Tillewein B. Electronic Literature and Its Influence on Print Stylistics / B. Tillewein - LAP Lambert Academic Publishing, 2014. – 104 p.

Словари

45. Ахманова О.С. Словарь лингвистических терминов. М.: Либроком, 2012. - 576 с.

46. Казаков Б.Н. Словарь научных терминов: Справочное пособие. – Казань: КГУ, 2008. - 32 с.

Иллюстративный материал

47. Ергин А. Начало работы с WebAssembly, используя только 14 строк на JavaScript: // Medium. 2018. URL: https://medium.com/devschacht/daniel-simmons-get-started-with-webassembly-using-only-14-lines-of-javascript-89960df71498. (Дата обращения: 23.04.2019).

48. Камышев И. Как работать с денежными значениями в JavaScript: // Medium. 2018. URL: https://medium.com/devschacht/how-to-handle-monetary-values-in-javascript-bb0706840f0e. (Дата обращения: 23.04.2019).

49. Миранович М. Играем в Mortal Kombat с TensorFlow.js: перенос обучения и дополнение данных: // Medium. 2018. URL: https://medium.com/devschacht/transfer-learning-tensorflow-js-data-augmentation-59823ab00b34. (Дата обращения: 23.04.2019).

50. Мэй Ш. Рефакторинг в NodeJS с помощью функций высшего порядка: // Medium. 2018. URL: https://medium.com/devschacht/refactoring-node-with-higher-order-functions-8f637c9d82da. (Дата обращения: 23.04.2019).

51. Панасович Н. Когда использовать var, let и const в Javascript: // Habr. 2019. URL: https://habr.com/ru/post/438880/. (Дата обращения: 23.04.2019).

52. Панасович Н. Разбираемся с асинхронностью в JavaScript: // Habr. 2019. URL: https://habr.com/ru/post/439620/. (Дата обращения: 23.04.2019).

53. Пыльцин А. Состояние JavaScript в 2018 году: обзор Vue.js: // Medium. 2018. URL: https://medium.com/devschacht/the-state-of-javascript-2018-the-view-on-vue-b1606c452344. (Дата обращения: 23.04.2019).

54. Станислав К. 12 малоизвестных фактов о CSS: // Habr. 2015. URL: https://habr.com/ru/post/263169/. (Дата обращения: 23.04.2019).

55. Уаллейс Е. Разработка инструмента веб-дизайнера на основе веб-приложения (Figma): // Habr. 2017. URL: https://habr.com/ru/post/343442/. (Дата обращения: 23.04.2019).

56. Хвостов В. Информатика в JavaScript: Быстрая сортировка (Quicksort): // Medium. 2017. URL: https://medium.com/devschacht/nicholas-c-zakas-computer-science-in-javascript-quicksort-afa07c0a47f0. (Дата обращения: 23.04.2019).

57. Чикирёв С. Нестандартный способ обучения новому языку программирования: // Habr. 2017. URL: https://habr.com/ru/company/wrike/blog/329006/. (Дата обращения: 23.04.2019).

58. Arora S. Understanding Asynchronous JavaScript: // Medium. 2018. URL: https://blog.bitsrc.io/understanding-asynchronous-javascript-the-event-loop-74cd408419ff. (Дата обращения: 23.04.2019).

59. Arsenault C. How to Remove Unused CSS for Leaner CSS Files: // Keycdn. 2018. URL: https://www.keycdn.com/blog/remove-unused-css. (Дата обращения: 12.12.2018).

60. Baumgartner S. JSX is syntactic sugar: // Fettblog. 2018. URL: https://fettblog.eu/jsx-syntactic-sugar. (Дата обращения: 21.12.2019).

61. Carver D. A Type-Driven Approach To React and Redux Using TypeScript: // Medium. 2019. URL: https://medium.com/func-shui/a-type-driven-approach-to-react-and-redux-using-typescript-f93470b0d724. (Дата обращения: 23.04.2019).

62. Dayan S. How to Handle Monetary Values in JavaScript: // Frontstuff. 2018. URL: https://frontstuff.io/how-to-handle-monetary-values-in-javascript. (Дата обращения: 23.04.2019).

63. Dua S. Unconventional way of learning a new programming language: // Hackernon. 2017. URL: https://hackernoon.com/unconventional-way-of-learning-a-new-programming-language-e4d1f600342c. (Дата обращения: 23.04.2019).

64. Gechev M. Playing Mortal Kombat with TensorFlow.js. Transfer learning and data augmentation: // GitHub. 2018. URL: https://blog.mgechev.com/2018/10/20/transfer-learning-tensorflow-js-data-augmentation-mobile-net/. (Дата обращения: 23.04.2019).

65. Gustafson A. From URL to Interactive: // Alistapart. 2019. URL: https://alistapart.com/article/from-url-to-interactive. (Дата обращения: 23.04.2019).

66. Lazaris L. 12 Little-Known CSS Facts (The Sequel): // Sitepoint. 2015. URL: https://www.sitepoint.com/12-little-known-css-facts-the-sequel. (Дата обращения: 23.04.2019).

67. May S. Refactoring Node with Higher-Order Functions: // Rangle. 2018. URL: https://rangle.io/blog/refactoring-node-with-higher-order-functions/. (Дата обращения: 23.04.2019).

68. McGinnis T. var vs let vs const in JavaScript: // Tylermcginnis. 2019. URL: https://tylermcginnis.com/var-let-const/. (Дата обращения: 23.04.2019).

69. Simmons D. Get started with WebAssembly — using only 14 lines of JavaScript: // freeCodeCamp. 2018. URL: https://medium.freecodecamp.org/get-started-with-webassembly-using-only-14-lines-of-javascript-b37b6aaca1e4. (Дата обращения: 23.04.2019).

70. Thoms J. Rethinking JavaScript: Replace break by going functional: // Hackernoon. 2019. URL: https://hackernoon.com/rethinking-javascript-break-is-the-goto-of-loops-51b27b1c85f8. (Дата обращения: 23.04.2019).

71. Wallace E. Building a professional design tool on the web: // Figma. 2015. URL: https://www.figma.com/blog/building-a-professional-design-tool-on-the-web. (Дата обращения: 23.04.2019).

72. Weaver B. Destructuring Methods in Ruby: // Medium. 2019. URL: https://medium.com/rubyinside/destructuring-in-ruby-9e9bd2be0360. (Дата обращения: 23.04.2019).

73. Willoughby J. The State of Javascript 2018: The View on Vue: // Telerik. 2018. URL: https://www.telerik.com/blogs/the-state-of-javascript-2018-the-view-on-vue. (Дата обращения: 23.04.2019).

74. Zagrabski K. After two years with TypeScript – was it worth it?: //Ecom. 2018. URL: https://ecom.software/en/tytul/ . (Дата обращения: 21.12.2018).