https://doi.org/10.25559/SITITO.020.202403.740-747
УДК 004.435

Оригинальная статья

Количественная оценка преимуществ языка описания задачи по сравнению с языком Perl и оптимизация транслятора

Е. А. Доренская
ФГБУ «Национальный исследовательский центр «Курчатовский институт»», г. Москва, Российская Федерация
Адрес: 123182, Российская Федерация, г. Москва, пл. Академика Курчатова, д. 1
dorenskaya@itep.ru

Аннотация

В статье проведено сравнение текстов, написанных на языке описания задачи (CDTL) с аналогичными кодами алгоритмического языка высокого уровня Perl. Для того, чтобы лучше, понять преимущества языка описания задачи (ранее язык описания проблемы), были произведены количественные расчёты по таким важным показаниям как сложность написания программы, количество ошибок и коэффициент сжатия текста описания CDTL по сравнению с кодами программ на алгоритмическом языке программирования высокого уровня Perl. Оценка сокращения сложности разработки программ на CDTL проводилась с использованием метрик Холстеда и Джилба. Количество сокращения ошибок в программах рассчитывалось с помощью метрики Холстеда и простой интуитивной модели.

Ранее созданный транслятор с языка CDTL в Perl нуждался в доработке, к тому же имелись и возможности для значительного его улучшения. Поскольку база данных алгоритмов (специальная таблица базы данных для хранения описаний программных модулей определённой проблематики и ссылок на их код) была создана и добавлена в открытый доступ, нужно было открыть к ней доступ людям, которые будут использовать модули в своих целях и оставлять на них отзывы. С помощью информации из отзывов можно находить и исправлять ошибки в кодах программ, находящихся в базе. Для решения задачи быстрого анализа отзывов на модули была написана специальная программа для ЭВМ.

Также был проведён экспериментальный анализ похожих отзывов и описаны его результаты. В результате этого эксперимента программа распознала верно 92% отзывов, в распознании 4% отзывов допустила ошибку и 4% отзывов она распознать не смогла. Помимо этого, был проведён эксперимент по генерации кода транслятора из CDTL в Perl с помощью самого этого транслятора, который закончился весьма успешно. Длина описания задания на CDTL составляет 10 строк. Длина кода транслятора, написанного программистом – 771 строка. Результаты всех этих работ представлены в данной статье.

Ключевые слова: транслятор, язык CDTL, расчёт сокращения числа ошибок, расчёт сложности написания программ

Конфликт интересов: автор заявляет об отсутствии конфликта интересов.

Для цитирования: Доренская Е. А. Количественная оценка преимуществ языка описания задачи по сравнению с языком Perl и оптимизация транслятора // Современные информационные технологии и ИТ-образование. 2024. Т. 20, № 3. С. 740-747. https://doi.org/10.25559/SITITO.020.202403.740-747

Original article

Quantification of the Advantages of the Task Description Language Compared to Perl and Optimization of the Translator

E. A. Dorenskaya
National Research Centre «Kurchatov Institute», Moscow, Russian Federation
Address: 1 Academician Kurchatov Square, Moscow 123182, Russian Federation
dorenskaya@itep.ru

Abstract

In order to better understand the advantages of the Task Description Language (formerly the problem description language), we performed quantitative calculations to compare it with a high-level algorithmic programming language for such important indications as the complexity of writing a program, the number of errors and the compression ratio of the CDTL description text compared to program codes in an algorithmic language. Evaluation of the reduction in the complexity of developing programs on CDTL was carried out using Halstead and Gilb metrics. The amount of error reduction in the programs was calculated using the Halstead metric and a Simple Intuitive Model.

The previously created translator from CDTL to Perl needed to be improved, besides there were opportunities for its significant improvement. Since the algorithm database was created and added to the public domain, it was necessary to open access to it to people who would use the modules for their own purposes and leave feedback on them. With the help of information from reviews, you can find and correct errors in the codes of programs in the database. An experimental analysis of similar reviews was carried out and its results were described.

Keywords: translator, CDTL language, calculation of error reduction, calculation of the complexity of writing programs

Conflict of interests: The author declares no conflict of interests.

For citation: Dorenskaya E.A. Quantification of the Advantages of the Task Description Language Compared to Perl and Optimization of the Translator. *Modern Information Technologies and IT-Education.* 2024;20(3):740-747 https://doi.org/10.25559/SITITO.020.202403.740-747

Введение

Раньше задача программирования решалась комбинированием команд процессора. Поэтому для ещё большего сокращения человеческого участия нашей командой было решено попробовать комбинировать крупные программные модули[1] [1-4]. Многообразие таких модулей будет более чем на порядок-два больше множества команд процессора. Крупные программные модули должны размещаться в отдельных базах данных соответствующей области проблематики. Пример такой базы был создан, он называется банк алгоритмов[2] [5]. Модули, находящиеся в нём, могут применяться для защиты Web-сервера.

Для замены описания алгоритма описанием задачи был предложен специальный язык Creating Description Task Language (CDTL, ранее PDL) [1]. В этой статье описывается его оценка и проверка с помощью различных методов. Для перевода описаний в код на языке программирования высокого уровня был использован специальный транслятор из CDTL в Perl [1]. Но для того, чтоб улучшить работу транслятора, необходимо было провести его оптимизацию [6-14].

Цель исследования

Целью данного исследования является анализ и усовершенствование языка описания задачи CDTL (ранее язык описания проблемы, PDL), описанного в статье [1]. Для оценки преимуществ язык CDTL по сравнению с языком Perl, необходимо провести их сравнения разными способами. В процессе анализа и оценки языка CDTL и транслятора были выявлены некоторые их «точки роста», поэтому возникла необходимость в оптимизации.

Материалы и методы

Оптимизация транслятора из CDTL в Perl

Во время доклада на семинаре ИСП РАН с целью представления языка CDTL, было предложено попробовать с помощью транслятора из CDTL в Perl воссоздать сам этот транслятор. Данная задача в итоге была успешно реализована. Для её реализации было написано специальное задание на CDTL для формирования транслятора, с помощью которого был сгенерирован программный код на языке Perl. Полученная программа была сравнена с кодом, написанным человеком. Длина описания задания на CDTL – 10 строк. Длина кода транслятора, написанного человеком – 771 строка[3] [15]. Сформированный транслятор опробован при генерации 5 программных модулей по их описаниям.

Желательно, чтобы модули в банке имели минимальное число ошибок. Для этой цели в данном исследовании был применен метод «Множества глаз». Для реализации данного метода, нужно, чтобы пользователи могли применять код программных модулей для своих целей и находить в нём ошибки. А также сообщать о найденных ошибках разработчикам.

Метод определения контекста слов был применён для анализа отзывов внешних пользователей на модули из банка алгоритмов[4]. Так как отзывы написаны на естественном языке, их анализ должен проводиться автоматически. При таком анализе часто бывает нужно определять контекстные значения отдельных слов и фрагментов текста. Смысл текста отзывов на модули банка алгоритмов распознаётся программой с помощью анализа специальных слов из базы данных, которые были найдены в тексте. Для этой цели был использован алгоритм определения контекста, изложенный в работе[5]. По контексту слов создана база данных на MySQL куда занесены слова, которые указывают на контекст отзыва

С помощью метода, описанного в данной работе, можно определять контекст различных слов, например, по тематике Web-безопасности (программа, сеть, протокол, канал, порт и др.). Для контекстного анализа этих слов уже были созданы специальная база данных и программа.

С применением метода, описанного работах[6] [16, 17] был создан и специальный анализатор отзывов. Дата, за которую необходимо проанализировать отзывы, указывается пользователем при запуске программы. На выходе программа отображает список, в котором указаны названия модулей, имена авторов, положительность/отрицательность отзывов и информация о найденных ошибках. Все эти сведения, за исключением результата анализа на положительность/отрицательность передаются напрямую из формы.

Анализ происходит за счёт вычисления контекстного значения [16], положительности и отрицательности отзыва и их сравнения между собой. Для расчёта используется специальная база данных тематических слов, которые могут встречаться в тексте отзыва. Для предварительного расчёта эффективности анализа было взято 25 положительных отзывов на произвольные программные модули и 25 отрицательных. Среди положительных программа неверно определила один отзыв и один не определила вообще. Такие ошибки связаны

[1] Бычков А. В., Доренская Е. А. Подход к минимизации количества программных ошибок в алгоритмических модулях, разработка «языка описания проблем» // Труды 60-й Всероссийской научной конференции МФТИ. 20-26 ноября 2017 г. Нано-, био-, информационные, когнитивные и социогуманитарные науки и технологии. М.: МФТИ, 2017. С. 7-8 [Электронный ресурс]. URL: https://abitu.net/public/admin/mipt-conference/INBIKST.pdf (дата обращения: 27.04.2024).

[2] Банк алгоритмов [Электронный ресурс]. URL: http://144.206.148.37/cgi-bin/algodb.cgi (дата обращения: 27.04.2024).

[3] Доренская Е. А. Методы создания изображений и программ с использованием описаний и функций // Решение. 2022. Т. 1. С. 206-209. EDN: PTSERV

[4] Отзыв на какой-либо модуль банка алгоритмов может быть отправлен через форму по URL: http://144.206.148.37/feedback/. В данной форме указываются в обязательном порядке название модуля, на который пишется отзыв, ФИО автора отзыва, достоинства/недостатки программы и имеется возможность занесения информации о найденных ошибках. Для этой цели предусмотрено специальное поле. Анализ таких отзывов необходим для выявления и устранения ошибок в программах, находящихся в банке алгоритмов. Сейчас этот банк доступен по URL: http://144.206.148.37/cgi-bin/algodb.cgi и программные модули может скачать любой желающий.

[5] Патент № 2685044 C1 Российская Федерация, МПК G06F 17/27. Способ определения контекста слова и текстового файла : № 2018124219 : заявл. 03.07.2018 : опубл. 16.04.2019 / Е. А. Доренская, Ю. А. Семенов ; заявитель Федеральное государственное бюджетное учреждение «Институт теоретической и экспериментальной физики имени А.И. Алиханова Национального исследовательского центра «Курчатовский институт»» (НИЦ «Курчатовский институт» – ИТЭФ). EDN: ZDSZRJ

[6] Там же.

с отсутствием тематических слов в отзыве или малым их количеством. В результате программа распознала верно 92% отзывов, в распознании 4% отзывов допустила ошибку и 4 % отзывов не распознала совсем (см. рис. 1). Это говорит о том, что программа может распознавать отзывы с большой точностью, но, в редких случаях, ошибается и не может распознать точный смысл. Подобные ошибки довольно маловероятны и могут аналогичным образом допускаться человеком.

Р и с. 1. Доля не распознанных и неверно распознанных отзывов на программные модули банка алгоритмов
F i g. 1. The percentage of unrecognized and incorrectly recognized reviews for software modules in the algorithm bank

Источник: здесь и далее в статье все таблицы и рисунки составлены автором.
Source: Hereinafter in this article all tables and figures were made by the author.

Оценка сокращения сложности разработки программ на CDTL с помощью метрики Холстеда и Метрики Джилба

Использование метрик программного кода даёт возможность оценивать разнообразные характеристики ПО. С помощью них можно рассчитать сложность или надежность кода программы[7] [18]. Это очень важно, потому что из-за большой сложности могут серьёзно увеличиваться время и усилия, затраченные на разработку программного продукта.

Чем проще написать программный код, тем меньше человек в нём сделает ошибок. Сложность программ защиты Web-сервера на метаязыке CDTL и на Perl, была вычислена с помощью 2-х метрик разного вида. Это метрика Джилба и метрика Холстеда[8]. Они являются количественными, поэтому предназначены для работы с исходным кодом и сложность с их помощью вычислить достаточно просто. Сложность программ на Perl по метрике Холстеда равна 104,92, что выше аналогичного показателя программ CDTL (23,61) на 77,5%. Сложность программ на Perl по метрике Джилба равна 153, что выше аналогичного показателя программ CDTL (4) на 97,39%.

Увеличение разницы в значениях при вычислениях с помощью метрики Джилба связано с тем, что там сравнивается количество операторов типа IF-THEN-ELSE, а в метрике Холстеда используются любые операторы и операнды. Это говорит о том, что в метаязыке CDTL меньше операторов IF-THEN-ELSE, где вероятнее совершить ошибки программисту. Из этих показателей следует, что программный код на CDTL написать проще, чем на Perl. Поэтому при использовании CDTL человек будет совершать меньше ошибок.

Оценка сокращения количества ошибок в программах при применении метаязыка CDTL с помощью метрики Холстеда и простой интуитивной модели

Основным преимуществом CDTL по сравнению с алгоритмическими языками является минимизация человеческого участия в генерации кода программы [5], [19-25]. Благодаря этому сокращается и количество ошибок, которые могут возникнуть, а также время, которое затрачивается на написание программы. Сам процесс программирования упрощается, благодаря чему большее количество людей могут принимать в нём участие.

Снижения числа ошибок при использовании CDTL происходит в основном за счёт сокращения количества программного кода. Поэтому для определения сокращения количества программных ошибок, которое должно произойти благодаря применению CDTL, (наиболее удобно использовать простую интуитивную модель и метрику Холстеда).

Правда надо заметить, что Метрика Холстеда имеет небольшой недостаток, в ней учитываются только те ошибки, которые пользователь создаёт в описании. Но данный недостаток устраняется за счёт использования интуитивной модели. Она была применена для дополнительной проверки результатов метрики Холстеда. С помощью неё сравнивался код на Perl, написанный вручную, с кодом на Perl, сгенерированным транслятором. Такое сравнение было необходимо, чтобы убедиться, что в коде программ, сгенерированных из описаний на CDTL не будет столько же ошибок, как и в написанных на Perl. Ошибки могут быть в программных модулях, примитивах или самом трансляторе. И если их много, они могут повлиять на результаты работы программы. Чтобы проверить, насколько существенно такое влияние, была использована простая интуитивная модель.

Сокращение ошибок, при расчёте с помощью интуитивной модели получилось даже более существенным, чем при использовании метрики Холстеда. Однако отличия небольшие. Результаты расчетов по данной методике представлены в таблице 4.3.

[7] Ледовских И. Н. Метрики сложности кода. Технический отчет 2012-2. М.: ИСП РАН, 2021. 22 с. [Электронный ресурс]. URL: https://www.ispras.ru/preprints/docs/prep_25_2013.pdf (дата обращения: 27.04.2024).

[8] Там же.

Таблица 1. **Расчёт сокращения количества ошибок с помощью простой интуитивной модели и метрики Холстеда в пакете программ защиты Web - сервера при использовании CDTL**

Table 1. **Calculating Error Reduction Using a Simple Intuitive Model and Halstead Metrics in a Web Server Security Suite Using CDTL**

Количество строк кода		Объём в битах		Приблизительное количество ошибок, шт.		Сокращение ошибок в % при использовании CDTL	
в описании задачи на CDTL	в программе на Perl	Описания проблемы на CDTL	в программе на Perl	в описании задачи на CDTL	в программе на Perl	Метрика Холстеда	Простая интуитивная модель
52	335	29600	105064	9,87	35,02	71,8	86,7

Конечно, возникает такой вопрос, а что будет, если программные модули, примитивы или сам транслятор будут написаны с большим количеством ошибок? Авторы уверены, что даже такая ситуация будет со временем решена, поскольку исходные коды будут добавляться в банк алгоритмов, который будет находиться в открытом доступе. Следовательно, этими модулями смогут пользоваться все желающие и сообщать о найденных ошибках разработчику. И так ошибки будут сокращены до минимума.

Оценка сжатия текста, написанного программистом, при использовании CDTL

Для того, чтобы оценить насколько уменьшается описание, написанное человеком на языке CDTL, по сравнению с кодом на языке Perl был вычислен коэффициент сжатия. Он был рассчитан для более чем 10 описаний на языке CDTL и аналогичных программ на Perl. Данный коэффициент показывает во сколько раз сокращается текст при использовании CDTL. Была построена зависимость вероятности коэффициента сжатия текста при использовании CDTL (см. рис. 2).

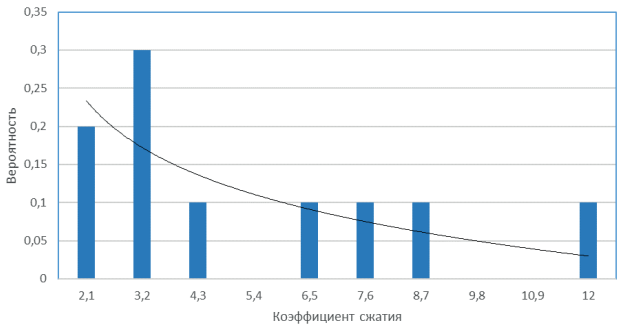

Рис. 2. Зависимость вероятности коэффициента сжатия текста при использовании CDTL

Fig. 2. The dependence of the compression ratio on the probability of text using CDTL

Минимальный коэффициент сжатия при использовании CDTL в данной выборке равен 2,1. Максимальный коэффициент = 12. Для данных значений была рассчитана дисперсия, которая оказалась равна 9,8. Авторы предполагают, что величина сжатия будет сильно варьироваться в зависимости от задачи.

Математическое ожидание коэффициента сжатия текста описания CDTL по сравнению с языком PERL составляет 5,3±3,1. Приблизительно во столько раз наиболее вероятно, что текст описания задачи будет меньше кода программы на Perl при использовании метаязыка CDTL. Данный результат является довольно значительным и это показывает, что при использовании CDTL написание программы для человека существенно упростится. Помимо этого, коэффициент сжатия будет, скорее всего, увеличиваться при создании новых модификаций метаязыка CDTL. Среднеквадратичное отклонение коэффициента сжатия σ получилось довольно небольшим – 3,1.

Результаты исследования

1. Был создан транслятор для транслятора из CDTL в Perl. Длина описания задания на CDTL составляет 10 строк. Длина кода транслятора, написанного программистом – 771 строку;
2. Была создана и апробирована программа для анализа отзывов на модули, находящиеся в базе данных алгоритмов. По итогам апробации было выяснено, что программа верно распознаёт 92% отзывов;
3. Текст описания на языке описания задачи в среднем получается в 5,3 раза короче, чем код алгоритмического языка, а ошибок в 5 раз меньше;
4. По метрике Холстеда при применении CDTL сложность разработки кода сокращается на 77,5%, а по метрике Джилба на 97,4%;
5. По метрике Холстеда при применении CDTL количество ошибок сокращается на 71,8%, а по простой интуитивной модели, на 86,7%.

Обсуждение и заключение

Было проведено сравнение текстов, написанных на языке описания задачи (CDTL) с аналогичными кодами алгоритмического языка высокого уровня Perl. Это сравнение показало, что язык описания задачи обладает многими преимуществами. Был оптимизирован транслятор из CDTL в Perl таким образом, что количество ошибок, допускаемых человеком максимально сокращается.

Список использованных источников

[1] Доренская Е. А., Куликовская А. А., Семенов Ю. А. Язык описания проблемы и исследование его возможностей // Современные информационные технологии и ИТ-образование. 2020. Т. 16, № 3. С. 653-663. https://doi.org/10.25559/SITITO.16.202003.653-663

[2] Доренская Е. А., Семенов Ю. А. О технологии программирования, ориентированной на минимизацию программных ошибок // Современные информационные технологии и ИТ-образование. 2017. Т. 13, № 2. С. 50-56. https://doi.org/10.25559/SITITO.2017.2.226

[3] Dorenskaya E. A., Semenov Y. A. New Methods of Minimizing the Errors in the Software // CEUR Workshop Proceedings. 2018. Vol. 2267. P. 150-154. URL: https://ceur-ws.org/Vol-2267/150-154-paper-27.pdf (дата обращения: 27.04.2024).

[4] Семенов Ю. А., Овсянников А. П., Овсянникова Т. В. Разработка банка алгоритмов и основ языка описания проблем с целью минимизации числа программных ошибок // Труды НИИСИ РАН. 2016. Т. 6, № 2. С. 96-100. EDN: ZCCBKT

[5] Куликовская А. А., Доренская Е. А., Семенов Ю. А. Разработка банка алгоритмов и метода поиска программ в соответствии с требованиями пользователей // Современные информационные технологии и ИТ-образование. 2020. Т. 16, № 1. С. 81-89. https://doi.org/10.25559/SITITO.16.202001.81-89

[6] Chen M. Trust, understanding, and machine translation: the task of translation and the responsibility of the translator // AI & SOCIETY. 2024. Vol. 39. P. 2307-2319. https://doi.org/10.1007/s00146-023-01681-6

[7] Jose M. A., Cozman F. G. A multilingual translator to SQL with database schema pruning to improve self-attention // International Journal of Information Technology. 2023. Vol. 15, No. 6. P. 3015-3023. https://doi.org/10.1007/s41870-023-01342-3

[8] On-Demand Triggered Memory Management Unit in Dynamic Binary Translator / B. Xie [et al.] // Advanced Parallel Processing Technologies ; ed. by C. Li, Z. Li, L. Shen, F. Wu, X. Gong. APPT 2023. Lecture Notes in Computer Science. Vol. 14103. Singapore: Springer, 2024. P. 297-309. https://doi.org/10.1007/978-981-99-7872-4_17

[9] Mutation analysis for evaluating code translation / G. Guizzo, J. M. Zhang, F. Sarro [et al.] // Empirical Software Engineering. 2024. Vol. 29, No. 1. Article number: 19. https://doi.org/10.1007/s10664-023-10385-w

[10] Even-Mendoza K., Cadar C., Donaldson A. F. CsmithEdge: more effective compiler testing by handling undefined behaviour less conservatively // Empirical Software Engineering. 2024. Vol. 27, No. 6. Article number: 129. https://doi.org/10.1007/s10664-022-10146-1

[11] Qurzon: A Prototype for a Divide and Conquer-Based Quantum Compiler for Distributed Quantum Systems / T. Chatterjee, A. Das, S. I. Mohtashim [et al.] // SN Computer Science. 2022. Vol. 3, No. 4. Article number: 323. https://doi.org/10.1007/s42979-022-01207-9

[12] Automatic Target Description File Generation / H. N. Geng, F. Lyu, M. Zhong [et al.] // Journal of Computer Science and Technology. 2023. Vol. 38, No. 6. P. 1339-1355. https://doi.org/10.1007/s11390-022-1919-x

[13] Preventing Vulnerabilities Caused by Optimization of Code with Undefined Behavior / R. V. Baev, L. V. Skvortsov, E. A. Kudryashov [et al.] // Programming and Computer Software. 2022. Vol. 48, No. 7. P. 445-454. https://doi.org/10.1134/S0361768822070027

[14] Enhancing embedded systems development with TS / X. Yu, W. Tang, T. Xiong [et al.] // Automated Software Engineering. 2024. Vol. 31, No. 1. Article number: 6. https://doi.org/10.1007/s10515-023-00404-x

[15] Куликовская А. А, Доренская Е. А., Семенов Ю. А. Количественные характеристики безопасности программ // Современные информационные технологии и ИТ-образование. 2022. Т. 18, № 4. С. 855-860. https://doi.org/10.25559/SITITO.18.202204.855-860

[16] Доренская Е. А., Семенов Ю. А. Метод определения контекстных значений слов и документов // Современные информационные технологии и ИТ-образование. 2018. Т. 14, № 4. С. 896-902. https://doi.org/10.25559/SITITO.14.201804.896-902

[17] Доренская Е. А., Семенов Ю. А. Улучшенный алгоритм вычисления контекстного значения слов в тексте // Современные информационные технологии и ИТ-образование. 2019. Т. 15, № 4. С. 935-942. https://doi.org/10.25559/SITITO.15.201904.954-960

[18] Methods and software tools to support combined binary code analysis / V. A. Padaryan, A. I. Getman, M. A. Solovyev [et al.] // Programming and Computer Software. 2014. Vol. 40. P. 276-287. https://doi.org/10.1134/S0361768814050077

[19] Василенко Н. В., Макаров В. А. Модели оценки надежности программного обеспечения // Вестник Новгородского государственного университета им. Ярослава Мудрого. 2004. № 28. С. 126-132. EDN: PHYITB

[20] Лебедева Т. Н., Носова Л. С. Формализация данных в языке программирования 1С // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2015. № 3. С. 113-121. EDN: UBKNKP

[21] Казакова А. Е. Особенности семантики языков программирования // Вестник Московского университета. Серия 7: Философия. 2007. № 6. С. 69-75. EDN: JWPOYP

[22] Наумов Р. В. Актуальные языки программирования // ACADEMY. 2016. № 1. С. 49-50. EDN: VIQSND

[23] Ferrara P., Cortesi A., Spoto F., From CIL to Java bytecode: Semantics-based translation for static analysis leveraging // Science of Computer Programming. 2020. Vol. 191. Article number: 102392. https://doi.org/10.1016/j.scico.2020.102392

[24] Zaytsev V. Modelling of language syntax and semantics: The case of the assembler compiler // Journal of Object Technology. 2020. Vol. 19, No. 2. P. 1-22. https://doi.org/10.5381/jot.2020.19.2.a5

[25] Rahman M. M., Watanobe Y., Nakamura K. Source Code Assessment and Classification Based on Estimated Error Probability Using Attentive LSTM Language Model and Its Application in Programming Education // Applied Sciences. 2020. Vol. 10, issue 8. Article number: 2973. https://doi.org/10.3390/app10082973

Поступила 27.04.2024; одобрена после рецензирования 29.06.2024; принята к публикации 11.08.2024.

Об авторе:

Доренская Елизавета Александровна, инженер-программист, ФГБУ «Национальный исследовательский центр «Курчатовский институт»» (123182, Российская Федерация, **г**. Москва, пл. Академика Курчатова, д. 1), **ORCID: https://orcid.org/0000-0002-4249-5131,** dorenskaya@itep.ru

Автор прочитал и одобрил окончательный вариант рукописи.

References

[1] Dorenskaya E.A., Kulikovskaya A.A., Semenov Y.A. Exploring Possibilities of Language for Describing the Problem. *Modern Information Technologies and IT-Education.* 2020;16(3):653-663. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.16.202003.653-663

[2] Dorenskaya E.A., Semenov Y.A. About the programming techniques, oriented to minimize errors. *Modern Information Technologies and IT-Education.* 2017;13(2):50-56. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.2017.2.226

[3] Dorenskaya E.A., Semenov Y.A. New Methods of Minimizing the Errors in the Software. *CEUR Workshop Proceedings.* 2018;2267:150-154. Available at: https://ceur-ws.org/Vol-2267/150-154-paper-27.pdf (accessed 27.04.2024).

[4] Semenov Y.A., Ovsyannikov A.P., Ovsyannikova T.V. Development Bank of algorithms and basics of the language problem descriptions to minimize the number of software errors. *SRISA Proceedings.* 2016;6(2):96-100. (In Russ., abstract in Eng.) EDN: ZCCBKT

[5] Kulikovskaya A.A., Dorenskaya E.A., Semenov Yu.A. Development of an Algorithm's Bank and Method for Searching Programs in Accordance with User Requirements. *Modern Information Technologies and IT-Education*. 2020;16(1):81-89. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.16.202001.81-89

[6] Chen M. Trust, understanding, and machine translation: the task of translation and the responsibility of the translator. *AI & SOCIETY*. 2024;39:2307-2319. https://doi.org/10.1007/s00146-023-01681-6

[7] Jose M.A., Cozman F.G. A multilingual translator to SQL with database schema pruning to improve self-attention. *International Journal of Information Technology*. 2023;15(6):3015-3023. https://doi.org/10.1007/s41870-023-01342-3

[8] Xie B., et al. On-Demand Triggered Memory Management Unit in Dynamic Binary Translator. In: Li C., Li Z., Shen L., Wu F., Gong X. (Eds.) Advanced Parallel Processing Technologies. APPT 2023. *Lecture Notes in Computer Science.* Vol. 14103. Singapore: Springer; 2024. p. 297-309. https://doi.org/10.1007/978-981-99-7872-4_17

[9] Guizzo G., Zhang J.M., Sarro F., et al. Mutation analysis for evaluating code translation. *Empirical Software Engineering.* 2024;29(1):19. https://doi.org/10.1007/s10664-023-10385-w

[10] Even-Mendoza K., Cadar C., Donaldson A.F. CsmithEdge: more effective compiler testing by handling undefined behaviour less conservatively. *Empirical Software Engineering*. 2024;27(6):129. https://doi.org/10.1007/s10664-022-10146-1

[11] Chatterjee T., Das A., Mohtashim S.I., et al. Qurzon: A Prototype for a Divide and Conquer-Based Quantum Compiler for Distributed Quantum Systems. *SN Computer Science*. 2022;3(4):323. https://doi.org/10.1007/s42979-022-01207-9

[12] Geng H.N., Lyu F., Zhong M., et al. Automatic Target Description File Generation. *Journal of Computer Science and Technology*. 2023;38(6):1339-1355. https://doi.org/10.1007/s11390-022-1919-x

[13] Baev R.V., Skvortsov L.V., Kudryashov E.A., et al. Preventing Vulnerabilities Caused by Optimization of Code with Undefined Behavior. *Programming and Computer Software*. 2022;48(7):445-454. https://doi.org/10.1134/S0361768822070027

[14] Yu X., Tang W., Xiong T., et al. Enhancing embedded systems development with TS. *Automated Software Engineering.* 2024;31(1):6. https://doi.org/10.1007/s10515-023-00404-x

[15] Kulikovskaya A.A., Dorenskaya E.A., Semenov Yu.A. Quantitative Security Characteristics of Perl Programs. *Modern Information Technologies and IT-Education*. 2022;18(4):855-860. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.18.202204.855-860

[16] Dorenskaya E.A., Semenov Y.A. The determination method for contextual meanings of words and documents. *Modern Information Technologies and IT-Education*. 2018;14(4):896-902. (In Russ., abstract in Eng.) https://doi.org/10.25559/SITITO.14.201804.896-902

[17] Dorenskaya E.A., Semenov Y.A. The Improved Algorithm for Calculation of the Contextual Words Meaning in the Text. *Modern Information Technologies and IT-Education*. 2019;15(4):954-960. https://doi.org/10.25559/SITITO.15.201904.954-960

[18] Padaryan V.A., Getman A.I., Solovyev M.A., et al. Methods and software tools to support combined binary code analysis. *Programming and Computer Software*. 2014;40:276-287. https://doi.org/10.1134/S0361768814050077

[19] Vasilenko N.V., Makarov V.A. Software reliability assessment models. *Vestnik Novgorodskogo gosudarstvennogo universiteta* = Vestnik of Novgorod State University. 2004;(28):126-132. (In Russ., abstract in Eng.) EDN: PHYITB

[20] Lebedeva T.N., Nosova L.S. Formalization of data in the programming language 1C. *Vestnik of Astrakhan State Technical University. Series: Management, Computer Sciences and Informatics.* 2015;(3):113-121. (In Russ., abstract in Eng.) EDN: UBKNKP
[21] Kazakova A.E. Peculiarities of semantics of programming languages. Moscow University Bulletin. Series 7. Philosophy. 2007;(6):69-75. (In Russ., abstract in Eng.) EDN: JWPOYP
[22] Naumov R.V. *Aktual'nye jazyki programmirovanija* [Current programming languages]. *ACADEMY.* 2016;(1):49-50. (In Russ., abstract in Eng.) EDN: VIQSND
[23] Ferrara P., Cortesi A., Spoto F., From CIL to Java bytecode: Semantics-based translation for static analysis leveraging. *Science of Computer Programming*. 2020;191:102392. https://doi.org/10.1016/j.scico.2020.102392
[24] Zaytsev V. Modelling of language syntax and semantics: The case of the assembler compiler. *Journal of Object Technology.* 2020;19(2):1-22. https://doi.org/10.5381/jot.2020.19.2.a5
[25] Rahman M.M., Watanobe Y., Nakamura K. Source Code Assessment and Classification Based on Estimated Error Probability Using Attentive LSTM Language Model and Its Application in Programming Education. *Applied Sciences.* 2020;10(8):2973. https://doi.org/10.3390/app10082973

Submitted 27.04.2024; approved after reviewing 29.06.2024; accepted for publication 11.08.2024.

About the author:

Elizaveta A. Dorenskaya, Software Engineer, National Research Centre «Kurchatov Institute» (1 Academician Kurchatov Square, Moscow 123182, Russian Federation), **ORCID: https://orcid.org/0000-0002-4249-5131,** dorenskaya@itep.ru

The author has read and approved the final manuscript.