ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 33 Выпуск 5

ISSN Online 2220-6426
Volume 33 Issue 5

Институт системного
программирования
им. В.П. Иванникова РАН

Москва, 2021

ИСП РАН

ISSN 2079-8156 (Print), ISSN 2220-6426 (Online)

Труды Института системного программирования РАН
Proceedings of the Institute for System Programming of the RAS

Труды ИСП РАН – это издание с двойной анонимной системой рецензирования, публикующее научные статьи, относящиеся ко всем областях системного программирования, технологий программирования и вычислительной техники. Целью издания является формирование научно-информационной среды в этих областях путем публикации высококачественных статей в открытом доступе.

Издание предназначено для исследователей, студентов и аспирантов, а также практиков. Оно охватывает широкий спектр тем, включая, в частности, следующие:

- операционные системы;
- компиляторные технологии;
- базы данных и информационные системы;
- параллельные и распределенные системы;
- автоматизированная разработка программ;
- верификация, валидация и тестирование;
- статический и динамический анализ;
- защита и обеспечение безопасности ПО;
- компьютерные алгоритмы;
- искусственный интеллект.

Журнал издается по одному тому в год, шесть выпусков в каждом томе.

Поддерживается открытый доступ к содержанию издания, обеспечивая доступность результатов исследований для общественности и поддерживая глобальный обмен знаниями.

Труды ИСП РАН реферируются и/или индексируются в:

Proceedings of ISP RAS are a double-blind peer-reviewed journal publishing scientific articles in the areas of system programming, software engineering, and computer science. The journal's goal is to develop a respected network of knowledge in the mentioned above areas by publishing high quality articles on open access.

The journal is intended for researchers, students, and practitioners. It covers a wide variety of topics including (but not limited to):

- Operating Systems.
- Compiler Technology.
- Databases and Information Systems.
- Parallel and Distributed Systems.
- Software Engineering.
- Software Modeling and Design Tools.
- Verification, Validation, and Testing.
- Static and Dynamic Analysis.
- Software Safety and Security.
- Computer Algorithms.
- Artificial Intelligence.

The journal is published one volume per year, six issues in each volume.

Open access to the journal content allows to provide public access to the research results and to support global exchange of knowledge. **Proceedings of ISP RAS** is abstracted and/or indexed in:

УДК004.45

Редколлегия

Главный редактор - Аветисян Арутюн Ишханович, академик РАН, доктор физико-математических наук, профессор, ИСП РАН (Москва, Российская Федерация)

Заместитель главного редактора - Кузнецов Сергей Дмитриевич, д.т.н., профессор, ИСП РАН (Москва, Российская Федерация)

Члены редколлегии

Воронков Андрей Анатольевич, доктор физико-математических наук, профессор, Университет Манчестера (Манчестер, Великобритания)
Вирбицкайте Ирина Бонавентуровна, профессор, доктор физико-математических наук, Институт систем информатики им. академика А.П. Ершова СО РАН (Новосибирск, Россия)
Коннов Игорь Владимирович, кандидат физико-математических наук, Технический университет Вены (Вена, Австрия)
Ластовецкий Алексей Леонидович, доктор физико-математических наук, профессор, Университет Дублина (Дублин, Ирландия)
Ломазова Ирина Александровна, доктор физико-математических наук, профессор, Национальный исследовательский университет «Высшая школа экономики» (Москва, Российская Федерация)
Новиков Борис Асенович, доктор физико-математических наук, профессор, Санкт-Петербургский государственный университет (Санкт-Петербург, Россия)
Петренко Александр Федорович, доктор наук, Исследовательский институт Монреаля (Монреаль, Канада)
Черных Андрей, доктор физико-математических наук, профессор, Научно-исследовательский центр CICESE (Энсенада, Баха Калифорния, Мексика)
Шустер Ассаф, доктор физико-математических наук, профессор, Технион — Израильский технологический институтTechnion (Хайфа, Израиль)

Editorial Board

Editor-in-Chief - Arutyun I. Avetisyan, Academician of RAS, Dr. Sci. (Phys.–Math.), Professor, Ivannikov Institute for System Programming of the RAS (Moscow, Russian Federation)

Deputy Editor-in-Chief - Sergey D. Kuznetsov, Dr. Sci. (Eng.), Professor, Ivannikov Institute for System Programming of the RAS (Moscow, Russian Federation)

Editorial Members

Igor Konnov, PhD (Phys.–Math.), Vienna University of Technology (Vienna, Austria)
Alexey Lastovetsky, Dr. Sci. (Phys.–Math.), Professor, UCD School of Computer Science and Informatics (Dublin, Ireland)
Irina A. Lomazova, Dr. Sci. (Phys.–Math.), Professor, National Research University Higher School of Economics (Moscow, Russian Federation)
Boris A. Novikov, Dr. Sci. (Phys.–Math.), Professor, St. Petersburg University (St. Petersburg, Russian Federation)
Alexandre F. Petrenko, PhD, Computer Research Institute of Montreal (Montreal, Canada)
Assaf Schuster, Ph.D., Professor, Technion - Israel Institute of Technology (Haifa, Israel)
Andrei Tchernykh, Dr. Sci., Professor, CICESE Research Centre (Ensenada, Baja California, Mexico).
Irina B. Virbitskaite, Dr. Sci. (Phys.–Math.), The A.P. Ershov Institute of Informatics Systems, Siberian Branch of the RAS (Novosibirsk, Russian Federation)
Andrey Voronkov, Dr. Sci. (Phys.–Math.), Professor, University of Manchester (Manchester, United Kingdom)

Адрес: 109004, г. Москва, ул. А. Солженицына, дом 25.
Телефон: +7(495) 912-44-25
E-mail: proceedings@ispras.ru
Сайт: https://ispranproceedings.elpub.ru/

Address: 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.
Tel: +7(495) 912-44-25
E-mail: proceedings@ispras.ru
Web: https://ispranproceedings.elpub.ru/

© Институт Системного Программирования им. В.П. Иванникова РАН, 2021

Содержание

Формальные правила продукции объектной нотации для данных, определяемых EXPRESS схемой
Семенов В.А., Аришин С.В., Семенов Г.В. 7

Формирование методологии разработки безопасного системного программного обеспечения на примере операционных систем
Девянин П.Н., Тележников В.Ю., Хорошилов А.В. 25

Онтологическое обеспечение управления рисками информационной безопасности
Бубакар И., Будько М.Б., Будько М.Ю., Гирик А.В. 41

Методы маркирования текстовых документов при печати посредством вертикального сдвига и изменения яркости фрагментов слов
Обыденков Д.О., Фролов А.Е., Маркин Ю.В., Фомин С.А., Кондратьев Б.В. 65

Методика сбора обучающего набора данных для модели обнаружения компьютерных атак
Гетьман А.И., Горюнов М.Н., Мацкевич А.Г., Рыболовлев Д.А. 83

Оптимизация программ на ProVerif для криптографических протоколов выработки общего ключа
Винарский Е.М., Демаков А.В. 105

Формальный язык первичных спецификаций криптографических протоколов
Прокопьев С.Е. 117

Разработка компилятора для стековой процессорной архитектуры TF16 на основе LLVM
Скворцов Л.В., Баев Р.В., Долгорукова К.Ю., Шарыгин Е.Ю. 137

Динамический анализ IoT-систем на основе полносистемной эмуляции в QEMU
Коваленко Р.Д., Макаров А.Н. 155

Использование библиотеки функционального программирования для решения численных задач на графических ускорителях с технологией CUDA
Краснов М.М., Феодоритова О.Б. 167

Автоматизация разработки на Vulkan: предметно-ориентированный подход
Фролов В.А., Санжаров В.В., Галактионов В.А., Щербаков А.С. 181

Управление требованиями при обучении профессиональных руководителей проектов
Варгас-Перес В.А., Варгас-Перес Л.С., Гутьеррес-Торнес А.Ф., Сото-Эрнандес А.М., Фелипе-Риверон Э.М. *205*

Гибкая разработка программного обеспечения для реконструкции жилья: землетрясение в Мексике в 2017 г.
Ортис-Эрнандес Х., Руис-Мартинес В.Х., Эрнандес-Перес М.Я., Миярес-Кастро Р. *219*

Математические модели и методы расчета процессов, сопровождающих обледенение летательного аппарата
Амелюшкин И.А., Кудров М.А., Морозов А.О., Щеглов А.С. *237*

Особенности реализации эффективного алгоритма параллельных вычислений для моделирования обледенения стреловидного крыла с профилем GLC-305
Кошелев К.Б., Осипов А.В., Стрижак С.В. *249*

Численное исследование микроклимата помещения с использованием СПО
Цынаева А.А., Цынаева Е.А. *259*

Тепловая нагрузка биметаллической оребренной трубки
Байметова Е.С., Гиззатуллина А.Ф., Королева М.Р., Мищенкова О.В., Пушкарев Ф.Н., Чернова А.А. *271*

Table of Contents

Formal Rules to Produce Object Notation for EXPRESS Schema-Driven Data
Semenov V.A., Arishin S.V., Semenov G.V. *7*

Building a methodology for secure system software development on the example of operating systems
Devyanin P.N., Telezhnikov V.I., Khoroshilov A.V. *25*

Ontological support of information security risk management
Boubacar I., Budko M.B., Budko M.Y., Guirik A.V. *41*

Printed text documents watermarking based on vertical word shift and word fragments brightness changing
Obydenkov D.O., Frolov A.E., Markin Y.V., Fomin S.A., Kondrat'ev B.V. *65*

Methodology for Collecting a Training Dataset for an Intrusion Detection Model
Getman A.I., Goryunov M.N., Matskevich A.G., Rybolovlev D.A. *83*

Optimization of ProVerif programs for AKE-protocols
Vinarskii E.M., Demakov A.V. *105*

A formal language for primary specifications of the cryptographic protocols
Prokopev S.E. *117*

Developing an LLVM-based compiler for stack based TF16 processor architecture
Skvortsov L.V., Baev R.V., Dolgorukova K.Y., Sharygin E.Y. *137*

Dynamic analysis of IoT systems based on full-system emulation in QEMU
Kovalenko R.D., Makarov A.N. *155*

Using the functional programming library for solving numerical problems on graphics accelerators with CUDA technology
Krasnov M.M., Feodoritova O.B. *167*

Development in Vulkan: a domain-specific approach
Frolov V.A., Sanzharov V.V., Galaktionov V.A., Shcherbakov A.S. *181*

A Requirements Solution While Training Professional Project Leaders
Vargas-Pérez V.A., Vargas-Pérez L.S., Gutiérrez-Tornés A.F., Soto-Hernández A.M., Felipe-Riverón E.M. *205*

Agile Software Development for Housing Reconstruction: The 2017 Earthquake case in Mexico
Ortiz-Hernández J., Ruiz-Martínez V.J., Hernández-Pérez M.Y., Mijarez-Castro R. *219*

Mathematical models and methods of numerical investigation of processes which accompany aircraft icing
Amelyushkin I.A., Kudrov M.A., Morozov A.O., Shcheglov A.S. *237*

Features of the implementation of an efficient parallel computation algorithm for modeling the icing of a swept wing with a GLC-305 airfoil
Koshelev K.B., Osipov A.V., Strijhak S.V. *249*

Numerical simulation of indoor microclimate using free software
Tsynaeva A.A., Tsynaeva E.A. *259*

Heat load of bimetallic ribbed tube
Baimetova E.S., Gizzatullina A.F., Koroleva M.R., Mishchenkova O.M., Pushkarev F.N., Chernova A.A. *271*

DOI: 10.15514/ISPRAS-2021-33(5)-1

CC BY

Формальные правила продукции объектной нотации для данных, определяемых EXPRESS схемой

[1,2,3] *В.А. Семенов, ORCID: 0000-0002-8766-8454 <sem@ispras.ru>*
[1] *С.В. Аришин, ORCID: 0000-0001-6128-7082 <arishin@ispras.ru>*
[4] *Г.В. Семенов, ORCID: 0000-0003-4725-7666 <georgii.v.semenov@gmail.com>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский физико-технический институт, 141701, Россия, Московская область, г. Долгопрудный, Институтский пер., 9*
[3] *Национальный исследовательский университет «Высшая школа экономики», 101978, Россия, г. Москва, ул. Мясницкая, д. 20*
[4] *Национальный исследовательский университет «ИТМО», 197101, Россия, г. Санкт-Петербург, Кронверкский проспект, д. 49, литер А*

Аннотация. В последнее время системы управления данными об изделии (PDM) широко используются в сложных междисциплинарных проектах в различных промышленных областях. PDM-системы позволяют группам проектировщиков, инженеров и менеджеров удаленно общаться в сети, обмениваться и совместно пользоваться общей информацией об изделии. Для интеграции приложений CAD/CAM/CAE с PDM-системами и обеспечения их совместимости было разработано и используется специальное семейство стандартов STEP (ISO 10303). Частью этого семейства является объектно-ориентированный язык моделирования EXPRESS, предназначенный для формального описания схем данных, а также форматы файлов для хранения и передачи данных об изделии, управляемых этими схемами. Это формат кодирования данных открытым текстом SPF и STEP-XML. В настоящее время, с развитием и широким внедрением веб-технологий, формат JSON становится все более популярным благодаря тому, что он подходит для задач обмена и хранения объектно-ориентированных данных, а также благодаря его простому и удобному для анализа синтаксису. В статье исследуется возможность применения формата JSON для представления, хранения и однозначной интерпретации данных об изделии. В предположении, что данные определяются информационными схемами на языке EXPRESS, в статье предложены и описаны формальные правила продукции объектной нотации JSON. Приводятся примеры, иллюстрирующие предложенные правила. Также обсуждаются результаты проведенных вычислительных экспериментов, которые подтверждают преимущества использования формата JSON по сравнению с SPF и STEP-XML и служат основанием для его широкого применения при интеграции программных приложений.

Ключевые слова: объектно-ориентированное моделирование; системная интеграция; интероперабельность; STEP; EXPRESS; JSON

Для цитирования: Семенов В.А., Аришин С.В., Семенов Г.В. Формальные правила продукции объектной нотации для данных, определяемых EXPRESS схемой. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 7-24. DOI: 10.15514/ISPRAS–2021–33(5)–1

Formal Rules to Produce Object Notation for EXPRESS Schema-Driven Data

[1,2,3] *V.A. Semenov, ORCID: 0000-0002-8766-8454 <sem@ispras.ru>*
[1] *S.V. Arishin, ORCID: 0000-0001-6128-7082 <arishin@ispras.ru>*
[4] *G.V. Semenov, ORCID: 0000-0003-4725-7666 <georgii.v.semenov@gmail.com>*
[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *Moscow Institute of Physics and Technology, 9, Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia*
[3] *HSE University, 20, Myasnitskaya, Moscow, 101978, Russia*
[4] *ITMO University, Kronverksky pr. 49, bldg. A, St. Petersburg, 197101, Russia*

Abstract. Recently, product data management systems (PDM) are widely used to conduct complex multidisciplinary projects in various industrial domains. The PDM systems enable teams of designers, engineers, and managers to remotely communicate on a network, exchange and share common product information. To integrate CAD/CAM/CAE applications with the PDM systems and ensure their interoperability, a dedicated family of standards STEP (ISO 10303) has been developed and employed. The STEP defines an object-oriented language EXPRESS to formally specify information schemas as well as file formats to store and transfer product data driven by these schemas. These are clear text encoding format SPF and STEP-XML. Nowadays, with the development and widespread adoption of Web technologies, the JSON language is getting increasingly popular due to it being apropos for the tasks of object-oriented data exchange and storage, as well as its simple, easy to parse syntax. The paper explores the topic of the suitability of the JSON language for the unambiguous representation, storage and interpretation of product data. Under the assumption that the product data can be described by arbitrary information schemas in EXPRESS, formal rules for the producing JSON notation are proposed and presented. Explanatory examples are provided to illustrate the proposed rules. The results of computational experiments conducted confirm the advantages of the JSON format compared to SPF and STEP-XML, and motivate its widespread adoption when integrating software applications.

Keywords: object-oriented modeling; system integration; interoperability; STEP; EXPRESS; JSON

For citation: Semenov V.A., Arishin S.V., Semenov G.V. Formal rules to produce object notation for EXPRESS schema-driven data. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 7-24 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–1

1. Введение

В последние десятилетия системы управления данными об изделии (Product data management, PDM) широко используются для реализации сложных междисциплинарных проектов в таких отраслях, как аэрокосмическая, оборонная, автомобильная промышленности, судостроение, электроника и строительство. Данные системы позволяют группам проектировщиков, инженеров и менеджеров удаленно взаимодействовать в сети, обмениваться и совместно пользоваться общей информацией об изделии, полученной с помощью приложений автоматизированного проектирования, производства и инженерного анализа (CAD/CAM/CAE). Для интеграции программных приложений и обеспечения их интероперабельности был разработан специальный международный стандарт представления и обмена данными об изделии (STEP) [1]. Его цель состоит в том, чтобы обеспечить общий для отдельных отраслей способ описания данных об изделии на протяжении всего жизненного цикла от проектирования до анализа, производства, контроля качества, инспекции, поддержки и эксплуатации. Благодаря своему нейтральному характеру такое описание подходит не только для организации файлового обмена данными, но и многопользовательской работы с использованием архивов и баз данных. Примечательно, что

многие другие отраслевые стандарты, такие как P-LIB (ISO 13584), CIS/2 (CIMSteel Integration Format), POSC/CAESAR (ISO 15926), PSL (ISO 18629) и IFC (ISO 16739), разработанные для аналогичных целей, заимствуют методологию и фундаментальные части STEP.

Стандарт STEP организован в виде нескольких частей, каждая из которых опубликована отдельно. Наибольший интерес в контексте обсуждаемой задачи представляют три из них: части 11, 21 и 28, поэтому в настоящей работе подробное рассмотрение стандарта ограничивается именно данными частями. Объектно-ориентированный язык моделирования данных EXPRESS (11 часть стандарта) определяет базовый метод описания [2,3]. Часть 21 определяет формат для записи данных в файл открытым текстом, также известный как STEP-файл или SPF [4]. Часть 28 использует язык разметки XML как для представления EXPRESS схем, так и непосредственно данных, описываемых схемами [5]. Данный формат известен как STEP-XML.

В настоящее время с развитием веб-технологий язык объектной нотации JavaScript (JSON) становится все более популярным, поскольку он подходит для задач обмена и хранения объектно-ориентированных данных [6]. JSON – это облегченный формат обмена данными в стиле "ключ-значение", который не зависит от языка программирования, при этом просто анализируется и генерируется приложениями. В то же время он легко читаем для человека. Применительно к данным об изделии эти свойства приближают его к формату SPF. В то же время JSON широко используется в качестве формата обмена данными между различными веб-приложениями, в частности, AJAX [7]. Примечательно, что JSON используется в веб-приложениях наряду с XML. Однако при использовании XML наблюдается значительное снижение производительности веб-сервисов из-за низкой эффективности чтения и анализа данных. На основе измерения показателей производительности, таких как количество отправленных объектов, среднее время передачи одного объекта, загрузка процессора и памяти, было показано, что обработка данных в JSON формате происходит значительно быстрее и обладает преимуществом над XML [8].

Несмотря на предпочтительность использования JSON по сравнению с XML для разработки веб-приложений, а также его удобочитаемости, близком к SPF, до сих пор не проводилось системных исследований, каким образом данные об изделии могут быть представлены, сохранены и однозначно проинтерпретированы с помощью объектной нотации. Целью данной статьи было восполнить данный пробел. В разд. 2 проведен краткий обзор языка EXPRESS, в котором основной акцент делается на категоризацию базовых типов данных. Пример схемы данных, описанной на EXPRESS, представлен в разд. 3. Здесь же приводится набор данных, использующий указанную схему, и представленный альтернативными способами в форматах SPF и JSON. Формальные правила для продукции объектной нотации JSON для данных, определяемых произвольной EXPRESS схемой, обобщены, систематизированы и проиллюстрированы в разд. 4. Разд. 5 посвящен вычислительным экспериментам и сравнительному анализу предложенного формата, основанного на JSON, со стандартными SPF и STEP-XML. В выводах определены перспективы стандартизации и широкого применения формата при интеграции индустриальных программных приложений, прежде всего, в области информационного моделирования зданий и сооружений (Building Information Modeling).

2. Краткий обзор языка моделирования данных EXPRESS

EXPRESS – это декларативный язык информационного моделирования, предусматривающий формальные правила описания объектов реального мира, их характеристик и ограничений. Многие конструктивные принципы языка EXPRESS были заимствованы из Ada, Algol, C, C++, Eiffel, Euler, Icon, Modula-2, Pascal, PL/I, SQL, ER и UML. Некоторые оригинальные возможности были специально привнесены для того, чтобы сделать EXPRESS более

подходящим для задач информационного моделирования. Также было разработано и стандартизировано графическое представление для подмножества EXPRESS-конструкций, называемое EXPRESS-G.

Основной конструкцией языка EXPRESS является схема, поэтому все определения появляются в контексте объявления схемы. Схема включает в себя определения типов данных и ограничений, заданных для их экземпляров. Типы данных подразделяются на следующие категории: простые типы, агрегированные типы (различные виды коллекций), перечисляемые типы, выборки, а также переопределенные типы данных, в том числе объектные типы (или сущности). Эти категории показаны на рис. 1. Предопределенные типы данных, предусмотренные EXPRESS, выделены жирным курсивом. Конструкции для пользовательских типов показаны обычным шрифтом. Стрелки указывают отношения обобщения/специализации между типами данных. В последней редакции стандарта EXPRESS для объектных типов может быть определено динамическое поведение как результат реакции на соответствующие события. Данные возможности языка не существенны для обсуждаемого круга задач и поэтому дальнейшее рассмотрение стандарта будет ограничено его первоначальной версией [3].

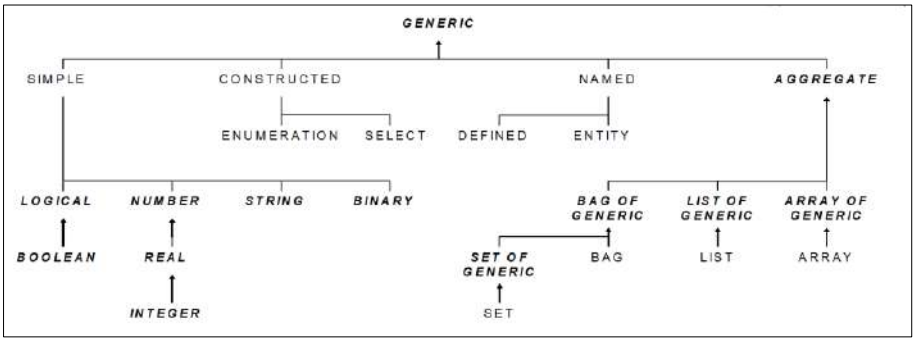


Рис. 1. Классификация типов данных языка EXPRESS
Fig. 1. Datatype categories available in EXPRESS language

Простые типы определяют атомарные единицы данных в EXPRESS. Простыми типами являются: вещественный (REAL), целочисленный (INTEGER), числовой (NUMBER), строковый (STRING), логический (LOGICAL), булев (BOOLEAN) и двоичный (BINARY).

Перечисляемые типы и выборки объявляются с использованием ключевых слов ENUMERATION и SELECT соответственно. Множеством значений ENUMERATION является множество имен, которые называются элементами перечисления. Два разных типа данных ENUMERATION могут содержать один и тот же элемент перечисления. В этом случае ссылки на элементы перечисления должны снабжаться идентификатором типа данных, чтобы однозначно интерпретироваться. Множеством значений типа данных SELECT является объединение множеств значений всех именованных типов данных, указанных в его определении. Тем самым, тип данных SELECT является обобщением каждого из указанных именованных типов данных.

Агрегации в EXPRESS описывают различные виды коллекций. Они задаются путем указания базового типа их элементов. Количество элементов может варьироваться и быть ограничено спецификацией. В EXPRESS представлены четыре основные типа агрегаций, определяемых с помощью следующих ключевых слов.

- BAG определяет неупорядоченную коллекцию, в которой разрешены повторяющиеся элементы.
- SET определяет неупорядоченную коллекцию уникальных элементов.
- LIST определяет упорядоченную коллекцию элементов, к которым можно получить доступ по их положению в последовательности. Объявление LIST может быть дополнено

с помощью ограничения UNIQUE, что подразумевает последовательность уникальных элементов.

- ARRAY определяет упорядоченную коллекцию фиксированного размера с индексированным доступом к элементам. Количество элементов в коллекции ARRAY может фиксироваться как конкретным значением, так и переменной.

Агрегации языка EXPRESS одномерны. Структуры данных, которые обычно имеют несколько измерений, могут быть представлены агрегацией, базовым типом которой является другая агрегация. Такой подход позволяет представить любое количество измерений. Например, матрица линейной алгебры может быть определена как массив массивов вещественных чисел.

Переопределенные типы данных, объявленные с помощью ключевого слова TYPE, включают типы данных с именем и с ограничениями. Множеством значений ограниченного переопределенного типа данных является подмножество значений базового типа, удовлетворяющих логическое ограничение WHERE.

Обобщенные типы данных обычно используются при определении параметров функций и процедур EXPRESS схемы. Тип данных GENERIC является обобщением всех типов данных. Тип данных AGGREGATE является обобщением всех агрегированных типов данных BAG, SET, LIST и ARRAY и их специализаций.

Объектные типы данных объявляются с помощью ключевого слова ENTITY. Объектный тип определяется в терминах атрибутов, каждый из которых характеризуется именем и множеством значений. Атрибуты могут принимать значения из указанного множества с учетом ограничений, определяемых индивидуально для отдельного атрибута или для комбинации атрибутов. Обычно значения атрибутов явно предоставляются реализацией при конструировании объектных экземпляров. Исключение делается для атрибутов, объявленных с ключевым словом OPTIONAL. Если атрибут не имеет значения, оно считается неопределенным (?). Ключевое слово DERIVED указывает, что значение атрибута должно быть вычислено предписанным способом. Ключевое слово INVERSE указывает, что значением атрибута является множество объектов, ассоциированных с данным объектом в определенной атрибутивной роли.

EXPRESS позволяет специфицировать объектные типы с помощью отношений подтипа/супертипа. В этом случае подтип наследует свойства (т. е. атрибуты, поведение, ограничения) своего супертипа. Атрибут, объявленный в супертипе, может быть повторно объявлен в подтипе с уточнением множества его значений. Например, атрибут типа данных NUMBER в супертипе может быть изменен на REAL или на тип данных INTEGER в подтипе, необязательный атрибут – на обязательный атрибут, явный атрибут – на вычисляемый атрибут, атрибут типа неупорядоченной коллекции – на атрибут упорядоченной коллекции.

Сложное наследование реализуется с помощью ограничений ONEOF, ANDOR и AND, которые устанавливают отношения между прямыми подтипами. Ограничение ONEOF устанавливает, что перечисленные в его списке подтипы являются взаимоисключающими. Каждый конструируемый объект должен иметь строго один из перечисленных подтипов и не может быть принадлежать любому другому подтипу (например, в результате конструирования составных экземпляров). Если подтипы не являются взаимоисключающими и объекты могут быть экземплярами более чем одного из перечисленных подтипов, то отношения между подтипами устанавливаются с помощью ограничения ANDOR. Если объекты могут быть подразделены на несколько групп взаимоисключающих подтипов (т. е. на несколько групп ONEOF), причем объекты каждой такой группы конструируются как составные экземпляры нескольких подтипов, то отношения между ними устанавливаются с помощью ограничения AND.

Наконец, при определении объектных типов обычно уточняется множество допустимых значений. С этой целью используются специальные ограничения, каждое из которых устанавливает одно из следующих свойств объекта:

- допустимое количество элементов для атрибута агрегатного типа;
- зависимость между значениями атрибутов или ограничение значений атрибута для данного объекта, называемые правилами домена WHERE;
- условие уникальности значений атрибутов на экстенте объектного типа (для всех объектов данного типа), называемое правилом уникальности UNIQUE;
- ограничение количества объектов, участвующих в обратном атрибуте данного объекта INVERSE;
- зависимость между экстентами нескольких объектных типов, которая определяется как глобальное правило RULE.

Рассмотренных выше принципов и конструкций языка EXPRESS достаточно для дальнейшего изложения. Более подробную информацию об языке можно почерпнуть непосредственно из стандартов [2,3].

3. Пример EXPRESS схемы и набора данных

В качестве примера рассмотрим схему ActorResource, фрагмент спецификации которой представлен в листинге 1. Диаграмма EXPRESS-G для данной схемы показана на рис. 2. Схема определяет общую предметную область для ключевых объектных сущностей Person, Organization, Address, PostalAddress, TelecomAddress, OrganizationRelationship и некоторых вспомогательных типов данных, таких как выборка ActorSelect, перечисление AddressTypeEnum, строковый тип Label и переопределенный строковый тип ActorRole.

Address объявлен как абстрактный супертип для объектных типов PostalAddress и TelecomAddress. Это означает, что Address конструируется только через его подтипы. PostalAddress и TelecomAddress наследуют как атрибуты, так и ограничения их общего супертипа Address. Это явные атрибуты Purpose, UserDefinedPupose, обратные атрибуты OfPerson, OfOrganization и правило WR1. Согласно правилу, значение атрибута UserDefinedPupose не может быть неопределенным, если атрибут Purpose принимает элемент перечисления USERDEFINED. Кроме того, PostalAddress определяет атрибут AddressLines, а TelecomAddress – атрибуты TelephoneNumbers, FacsimileNumbers, ElectronicMailAddresses, WWWUrls и собственное правило WR1.

```
TYPE ActorSelect = SELECT (Organization, Person);
  END_TYPE;
  TYPE AddressTypeEnum = ENUMERATION OF (OFFICE, HOME, USERDEFINED);
  END_TYPE;
  TYPE Label = STRING(255);
  END_TYPE;
  TYPE ActorRole = Label;
  END_TYPE;

  ENTITY Address
    ABSTRACT SUPERTYPE OF (ONEOF(PostalAddress, TelecomAddress));
      Purpose            : AddressTypeEnum;
      UserDefinedPurpose : OPTIONAL STRING;
    INVERSE
      OfPerson           : SET OF Person FOR Addresses;
      OfOrganization     : SET OF Organization FOR Addresses;
    WHERE
      WR1 : (Purpose <> AddressTypeEnum.USERDEFINED) OR
            ((Purpose = AddressTypeEnum.USERDEFINED) AND
```

```
              EXISTS(UserDefinedPurpose));
  END_ENTITY;

  ENTITY PostalAddress
    SUBTYPE OF(Address);
      AddressLines     : LIST [1:?] OF Label;
  END_ENTITY;

  ENTITY TelecomAddress
    SUBTYPE OF(Address);
      TelephoneNumbers        : OPTIONAL LIST [1:?] OF Label;
      FacsimileNumbers        : OPTIONAL LIST [1:?] OF Label;
      ElectronicMailAddresses : OPTIONAL LIST [1:?] OF Label;
      WWWUrls                 : OPTIONAL LIST [1:?] OF Label;
    WHERE
      WR1 : EXISTS (TelephoneNumbers) OR EXISTS (FacsimileNumbers) OR
            EXISTS (ElectronicMailAddresses) OR EXISTS (WWWUrls);
  END_ENTITY;

  ENTITY Organization;
      Id          : INTEGER;
      Name        : Label;
      Description : OPTIONAL STRING;
      Roles       : LIST [1:?] OF UNIQUE ActorRole;
      Addresses   : LIST [1:?] OF UNIQUE Address;
    INVERSE
      IsRelatedBy : SET OF OrganizationRelationship FOR
RelatedOrganizations;
      Relates     : SET OF OrganizationRelationship FOR
RelatingOrganization;
      Engages     : SET OF Person FOR EngagedIn;
    UNIQUE
      UR1 : Id;
    WHERE
      WR1 : SIZEOF( QUERY( temp <* Engages |
            'Director' in temp.Roles ) ) = 1;
  END_ENTITY;

  ENTITY OrganizationRelationship;
      Name                 : Label;
      Description          : OPTIONAL STRING;
      RelatingOrganization: Organization;
      RelatedOrganizations: SET [1:?] OF Organization;
    WHERE
      WR1 : SIZEOF( [RelatingOrganization] * RelatedOrganizations ) = 0;
  END_ENTITY;

  ENTITY Person;
      Id           : INTEGER;
       FamilyName   : OPTIONAL Label;
       GivenName    : OPTIONAL Label;
       MiddleNames  : OPTIONAL LIST [1:?] OF Label;
       PrefixTitles : OPTIONAL LIST [1:?] OF Label;
       SuffixTitles : OPTIONAL LIST [1:?] OF Label;
       Roles        : LIST [0:?] OF UNIQUE ActorRole;
       Addresses    : LIST [1:?] OF UNIQUE Address;
       EngagedIn    : SET OF Organization;
     UNIQUE
```

```
        UR1 : Id;
    WHERE
        WR1 : EXISTS(FamilyName) OR EXISTS(GivenName);
  END_ENTITY;

END_SCHEMA;
```

Листинг 1. Спецификация схемы ActorResource на языке EXPRESS
Listing 1. Specification of ActorResource schema in EXPRESS language

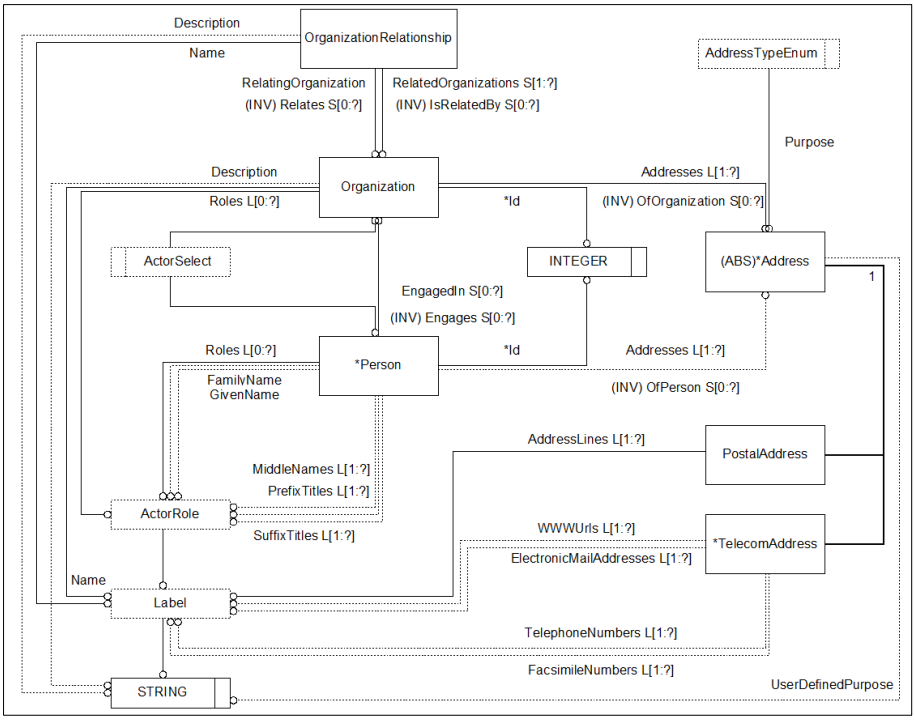


Рис. 2. Графическое представление схемы ActorResource на языке EXPRESS-G
Fig. 2. Graphical representation of ActorResource schema in EXPRESS-G language

Объектные сущности Person и Organization определяют собственные наборы явных и обратных атрибутов, а также задают ограничения на них в виде соответствующих правил. Объектный тип OrganizationRelationship определен в схеме для того, чтобы устанавливать квалифицированные атрибутивные отношения между объектами Organization. Для задания отношений «один ко многим» используются атрибуты RelatingOrganization и RelatedOrganization, первый из которых имеет тип Organization, а второй определен как непустое множество объектов типа Organization. Кроме того, используя обратные атрибуты IsRelatedBy, Relates, Engages объекта типа Organization, появляется возможность определить, с какими именно объектными экземплярами и в каких отношениях он участвует.

Для краткости опустим дальнейшие пояснения, поскольку оставшаяся часть спецификации схемы использует аналогичные конструкции.

В качестве примера рассмотрим набор данных, определяемых описанной выше EXPRESS схемой и представленных в файловом формате ASCII SPF (листинг 2). Файл организован как набор строк, каждая из которых представляет один из экземпляров объектного типа и передается в составе структуры обмена. Объектный тип экземпляра должен быть определен в EXPRESS схеме, указанной в заголовке файла. В нашем примере указанная схема – это ActorResource, а экземпляры – объекты соответствующих типов Organization, Person, PostalAddress, TelecomAddress и OrganizationRelationship, определяемых данной схемой. Каждый экземпляр представлен в структуре обмена точно один раз и имеет уникальный

идентификатор. Порядок экземпляров в структуре обмена не имеет значения. На экземпляры можно ссылаться с помощью идентификаторов из любого места структуры обмена.

```
#11 = Organization(1203, 'Automobile Inc.', 'In cars we trust.',
     ('Supply Chain Manager', 'Executive Manager', 'Sells Manager'),
     (#31, #34));
#12 = Organization(1204, 'Wheels Inc.', 'We do wheels.',
     ('Executive Manager', 'Sells Manager'), (#32, #35));
#13 = Organization(1205, 'Motor Inc.', 'Motors are essential.',
     ('Executive Manager', 'Sells Manager'), (#33, #36));

#31 = PostalAddress(.OFFICE., $, ('9292 Automobile Dr.', 'Mc Lean',
     'VA 22101'));
#32 = PostalAddress(.USERDEFINED., 'Sells Department',
     ('1172 Wheel Avenue', 'Fresno', 'CA 93711'));
#33 = PostalAddress(.USERDEFINED., 'Sells Department',
     ('1119 Motor Road', 'Reno', 'NV 89501'));
#34 = TelecomAddress(.OFFICE., $, ('678-762-2354', '678-762-2355'),
     $, ('automobile@cars.com'), ('http://www.cars.com/automobile'));
#35 = TelecomAddress(.USERDEFINED., 'Product Information',
     ('775-201-8669', '775-761-2384'), $, ('wheels@cars.com'),
     ('http://www.cars.com/wheels'));
#36 = TelecomAddress(.USERDEFINED., 'Product Information',
     ('609-639-9256', '201-213-0598'), $, ('motor@cars.com'),
     ('http://www.cars.com/motor'));

#51 = OrganizationRelationship('Consumer', $, #11, (#12, #13));
#52 = OrganizationRelationship('Supplier', $, #12, (#11));
#53 = OrganizationRelationship('Supplier', $, #13, (#11));

#61 = Person(901, 'Pringle', 'Andrew', $, ('Mr'), $,
     ('Supply Chain Manager', 'Executive Manager'), (#34), (#11));
#62 = Person(902, 'Martinez', 'Bill', $, ('Mr'), $,
     ('Executive Manager', 'Sells Manager'), (#35), (#12));
#63 = Person(903, 'Ackley', 'Chris', $, ('Mr'), $,
     ('Executive Manager', 'Sells Manager'), (#36), (#13));
```

Листинг 2. Пример набора данных, определенных схемой ActorResource и представленных в формате SPF

Listing 2. Sample dataset driven by ActorResource schema and represented in SPF format

Порядок параметров в строке экземпляра такой же, как порядок определения соответствующих атрибутов в объявлении объектной сущности. При этом атрибуты, определенные в супертипах, предшествуют атрибутам подтипов. Значение каждого параметра должно соответствовать типу данных атрибута.

В представленном наборе данных экземпляр PostalAddress с уникальным идентификатором #31 имеет следующие параметры: первый параметр «.OFFICE.» – значение атрибута Purpose, объявленного в супертипе Address, второй параметр «$» – неопределенное значение атрибута UserDefinedPurpose, также объявленного в супертипе Address, третий параметр «('9292 Automobile Dr.', 'Mc Lean', 'VA 22101 ')» – это значение явного атрибута AddressLines, объявленного в объекте PostalAddress как список строк. Здесь стоит указать, что в структуре обмена присутствуют только хранимые значения явных атрибутов, а производные и обратные атрибуты, объявленные с помощью ключевых слов DERIVED и INVERSE, игнорируются. Предполагается, что они всегда могут быть вычислены приложением по заданным явным атрибутам и поэтому нет необходимости включать их в структуру обмена.

4. Правила продукции объектной нотации JSON

JSON – это текстовый формат для представления и сериализации объектно-ориентированных данных, основанный на языке программирования ECMAScript, который в свою очередь является расширением JavaScript [6]. Данные в этом формате описываются с использованием четырех примитивных типов (строки, числа, логические значения и null) и двух структурированных типов (объекты и массивы). Объект – это неупорядоченная коллекция из нуля или более пар (членов) ключ / значение, в которых ключи представляются строками, а значения – любыми предусмотренными типами данных. Массив – это упорядоченная последовательность из нуля или более значений. Строка – это последовательность из нуля или более символов Unicode.

Данный раздел посвящен обобщению и систематизации формальных правил, посредством которых типы данных, описанные на языке EXPRESS, могут быть сопоставлены с типами данных JSON. Такое отображение должно определять непротиворечивый способ представления, хранения и интерпретации данных об изделии в формате JSON наряду с форматами SPF и STEP-XML, определяемыми действующими международными информационными стандартами.

Первые два правила носят общий характер и применяются независимо от пользовательских типов данных EXPRESS. Первое правило регулирует представление опциональных атрибутов, объявленных в схеме с использованием ключевого слова OPTIONAL. Такие атрибуты не обязаны иметь значений в объектных экземплярах. Незаданные значения кодируются в JSON значением "null", которое соответствует знаку доллара «$» в формате SPF. Заданные значения должны быть представлены в соответствии с правилами, представленными в следующих подразделах.

Второе правило касается уточнения типов данных EXPRESS. Уточнение необходимо в случаях, когда значение атрибута не может быть однозначно интерпретировано без явного указания типа данных. Обычно такие случаи возникают для атрибутов выборочного типа SELECT. Уточнение типа данных также необходимо в составных экземплярах объектных типов, объявленных с ограничениями подтипа/супертипа ONEOF, ANDOR, AND. Чтобы избежать неправильной интерпретации, простые экземпляры объектных типов должны быть снабжены именем типа данных. Для определенности и читаемости все имена типов данных указываются с использованием только строчных букв.

```
SCHEMA IllustrationResource;

TYPE Url = STRING; END_TYPE;
TYPE Png = BINARY; END_TYPE;
TYPE Pixels = LIST[1:?] OF INTEGER; END_TYPE;
TYPE Image = SELECT (Url, Png, Pixels); END_TYPE;

ENTITY Title;
  text : STRING;
END_ENTITY;

ENTITY Picture;
  figure : Image;
END_ENTITY;

ENTITY Illustration SUBTYPE OF (Title AND Picture);
END_ENTITY;

END_SCHEMA;


[
```

```
        {
           "_oid": "#1",
           "type": "Picture",
           "figure": {
              "type": "Pixels",
              "value": [0,255,255,128,128]
           }
        },
        {
           "_oid": "#2",
           "type": "Illustration",
           "_prototype":
           [
              {
              "type": "Title",
              "text": "Book"
              },
              {
              "type": "Picture",
              "figure": {
                  "type": "Url",
                  "value": "http://www.cars.com/automobile/picture.jpg"
              }
           ]
        }
    ]
```

Листинг 3. Пример уточнения типа данных EXPRESS в формате JSON
Listing 3. An example of EXPRESS datatype qualification in JSON format

Проиллюстрируем данное правило, используя схему IllustrationResource и набор данных, представленные в листинге 3. Схема определяет основные объектные сущности Title, Picture и Illustration, а также несколько вспомогательных типов данных. Тип данных Image является выборкой и объявлен как обобщение типов данных Url, Png, Pixels, каждый из которых соответствует одному альтернативному способу представления изображений. Поэтому, когда конструируется экземпляр Picture и инициируется его атрибут «figure», присвоенное значение должно дополняться именем типа Url, Png или Pixels. В приведенном наборе данных экземпляр типа Picture с идентификатором «# 1» определен как имеющий атрибут «figure», инициированный заданным списком целых чисел и квалифицированный как «Pixels». Как видно из примера, в таких случаях значения атрибутов должны быть представлены как вложенные объекты JSON с собственными членами «type», «value».

Схема IllustrationResource определяет также сложный объектный тип «Illustration», являющийся подтипом «Title» и «Picture». Это означает, что экземпляры типа Illustration состоят из простых экземпляров соответствующих супертипов. В приведенном примере экземпляр типа Illustration с идентификатором «# 2» определен как состоящий из экземпляров типов Title и Picture. Экземпляры супертипов должны быть представлены как объекты JSON и квалифицированы с помощью члена «type». Объекты включаются во внешний массив JSON, который определяется как член «_prototype» сложного экземпляра. Простым экземплярам в данном случае идентификаторы не присваиваются.

Стоит отметить, что порядок следования атрибутов в любом объекте JSON не имеет значения, однако рекомендуется первыми устанавливать атрибуты «_oid», «type» и «_prototype», а для следующих использовать порядок, в котором определены атрибуты сущности в исходной схеме. В подобном предположении удается оптимизировать обработку данных об изделии, представленных в формате JSON в соответствии с предложенными правилами.

4.1 Правила представления простых, конструируемых и агрегатных типов EXPRESS

Для простых типов данных EXPRESS определим следующие правила их представления в нотации JSON.

- Типу данных NUMBER соответствует числовой тип данных JSON. Представление числа аналогично тому, которое используется в большинстве языков программирования. Число представляется по десятичному основанию с использованием соответствующих цифр. Оно содержит целочисленный компонент, которому может предшествовать необязательный знак минус “-“, за которым может следовать дробная часть и / или экспоненциальная часть.
- Тип данных INTEGER рассматривается как целочисленное подмножество для числового типа данных JSON, а его экземпляры представляются как последовательности из одной или нескольких цифр без начальных нулей, которой может предшествовать знак минус “-“.
- Типу данных BOOLEAN соответствует логический тип данных JSON.
- Типу данных LOGICAL соответствует предопределенное перечисление строк "true", "false" и "unknown".
- Типу данных STRING соответствует строковый тип JSON, который определяется соглашениями, принятыми для семейства языков программирования C. Строка начинается и заканчивается кавычками. Все символы Unicode могут быть помещены в кавычки, за исключением символов, которые должны быть экранированы: кавычки, обратная косая черта и управляющие символы (от U + 0000 до U + 001F);
- Типу данных BINARY соответствует строковый тип JSON, значения которого представлены в закодированном виде в соответствии со стандартом Base64 [9].

Перечислимый тип данных ENUMERATION, объявленный в EXPRESS схеме, должен быть представлен как строка JSON, соответствующая одному из имен элементов перечисления. Тип данных SELECT отображается в типы данных JSON таким же образом, как и типы данных, перечисленные в его выражении. Значения этих типов данных, включая вложенные выборки, кодируются в соответствии с указанными выше правилами и должны быть снабжены полем «type» для уточнения типов выбранных значений.

Агрегатные типы данных BAG, SET, LIST, ARRAY представляются как массивы JSON. Допустимо использование неоднородных элементов разных типов данных. В массиве JSON каждый элемент должен быть закодирован в соответствии с его типом данных, объявленным в схеме EXPRESS. Порядок элементов в кодировке должен поддерживаться для упорядоченных агрегаций LIST и ARRAY. Если агрегация пуста, то она должна быть представлена как пустой массив JSON, а не как неопределенное значение, обозначенное ключевым словом "null".

Наконец, значение атрибута объектного типа ENTITY, являющееся ассоциацией на соответствующий объект данного типа, должно быть представлено в виде строки JSON, содержащей идентификатор объектного экземпляра. Напомним, что каждый объектный экземпляр представлен в структуре обмена ровно один раз.

4.2 Правила представления объектных типов EXPRESS

Структура обмена данными об изделии в формате JSON представляется как массив объектов, каждый из которых соответствует экземпляру объектного типа EXPRESS. Каждый объект представляется в виде набора пар ключ/значение.

Представление объекта должно содержать следующие элементы.

- Строковый идентификатор объекта с ключом «_oid». Его значение должно быть

уникальным в пределах структуры обмена. В зависимости от целевого приложения можно требовать и глобальную уникальность значений идентификаторов. Данный атрибут является обязательным для всех экземпляров объектных типов EXPRESS, не являющихся простыми экземплярами сложных объектных типов. Предшествующий символ подчеркивания, который используется в ключевом слове, должен предотвращать конфликты имен с возможными определениями базовой схемы. Для обеспечения согласованности с форматом SPF в качестве идентификаторов могут использоваться целочисленные значения, которым предшествует символ «#»;

- Тип объекта должен быть представлен как строковое значение JSON с ключом «type». Значение должно соответствовать имени объектного типа, определенного в базовой схеме. Данный атрибут является обязательным для всех объектов;
- Прототип объекта с ключом «_prototype» представляется как массив вложенных объектов JSON. Данный атрибут определяется только для составных экземпляров сложных объектных типов ENTITY, объявленных с использованием ограничений подтипа/супертипа ONEOF, ANDOR, AND. В таких случаях каждый простой экземпляр супертипа должен быть представлен как объект массива прототипа. Экземплярам супертипов объектные идентификаторы не назначаются. Дополнительные сведения о конструировании экземпляров сложных сущностей можно найти в стандарте языка EXPRESS [3];
- Каждый атрибут объекта соответствуют одному из явных атрибутов экземпляра объектного типа EXPRESS. В качестве ключа используется имя атрибута как оно определено в базовой схеме. Значение атрибута должно быть представлено в соответствии с правилами, описанными в предыдущем подразделе. Уточнение типа данных должно применяться во всех случаях, когда значение не может быть однозначно интерпретировано. Никаких ограничений на порядок задания атрибутов не накладывается. Обязательные атрибуты, не объявленные в определении объектного типа как OPTIONAL, обязаны быть заданы в структуре JSON объекта. Отсутствующие необязательные атрибуты считаются неустановленными. Атрибуты, объявленные в схеме как DERIVED и INVERSE, игнорируются.

4.3 Пример продукции объектной нотации на основе правил

Чтобы проиллюстрировать предложенные формальные правила, воспользуемся схемой ActorResource и набором данных из разд. 3.

```
[
   {
      "_oid":"#11",
      "type":"Organization",
      "id":1203,
      "name":"Automobile Inc.",
      "description":"In cars we trust.",
      "roles":["Supply Chain Manager", "Executive Manager",
               "Sells Manager"],
      "addresses":["#31", "#34"]
   },
{
      "_oid":"#12",
      "type":"Organization",
      "id":1203,
      "name":"Automobile Inc.",
      "description":"In cars we trust.",
      "roles":["Supply Chain Manager", "Executive Manager",
               "Sells Manager"],
      "addresses":["#31", "#34"]
```

```
      },
   {
         "_oid":"#13",
         "type":"Organization",
         "id":1203,
         "name":"Automobile Inc.",
         "description":"In cars we trust.",
         "roles":["Supply Chain Manager", "Executive Manager",
                  "Sells Manager"],
         "addresses":["#31", "#34"]
      },
      {
         "_oid":"#61",
         "type":"Person",
         "id":901,
         "familyName":"Pringle",
         "givenName":"Andrew",
         "middleNames":null,
         "prefixTitles":["Mr"],
         "suffixTitles":null,
         "roles": ["Supply Chain Manager", "Executive Manager"],
         "addresses":["#34"],
         "engagedIn":["#11"]
      },
      {
         "_oid":"#51",
         "type":"OrganizationRelationship",
         "name":"Consumer",
         "description":null,
         "relatingOrganization":"#11",
         "relatedOrganizations":["#12","#13"]
      },
      {
         "_oid":"#31",
         "type":"PostalAddress",
         "purpose":"OFFICE",
         "userDefinedPurpose":null,
         "addressLines":["9292 Automobile Dr.", "Mc Lean", "VA 22101"]
      },
      {
         "_oid":"#34",
         "type":"TelecomAddress",
         "purpose":"OFFICE",
         "userDefinedPurpose":null,
         "telephoneNumbers":["678-762-2354", "678-762-2355"],
         "facsimileNumbers":null,
         "electronicMailAddresses": ["automobile@cars.com"],
         "WWWUrls":["http://www.cars.com/automobile"]
      }
]
```


Листинг 4. Набор данных, определяемый схемой ActorResource и представленный в формате JSON
Listing 4. Sample dataset driven by ActorResource schema and encoded in JSON format

В листинге 4 приведен JSON-документ с тем же набором данных, который был представлен на рис. 2 в формате SPF. В структуре обмена определены те же объекты типов Organization, OrganizationRelationship, Person, PostalAddress и TelecomAddress с теми же значениями атрибутов. Для сопоставления наборов данных идентификаторы объектов были специально назначены такими же как в SPF-файле, хотя правила допускают более общие способы.

Важно отметить, что схема ActorResource была использована для наглядности и компактности представления данных. Она является частью спецификации на языке EXPRESS международного индустриального стандарта Industry Foundation Classes (IFC) [10], который определяет около тысячи объектных типов данных и несколько тысяч вспомогательных типов для представления данных об архитектурно-строительных проектах. Очевидно, что без обобщения и систематизации формальных правил отображения типов данных из EXPRESS в JSON и уточнения способов представления данных об изделии, их обработка и интерпретация разными приложениями не представлялись бы возможными.

5. Валидация формальных правил

Предложенные правила представления данных об изделии были апробированы в ходе разработки программного приложения, предназначенного для их сериализации в формат JSON и трехмерной визуализации. С помощью приложения была выполнена серия вычислительных экспериментов и проведен сравнительный анализ с альтернативными способами представления данных об изделии в стандартных форматах SPF и STEP-XML.

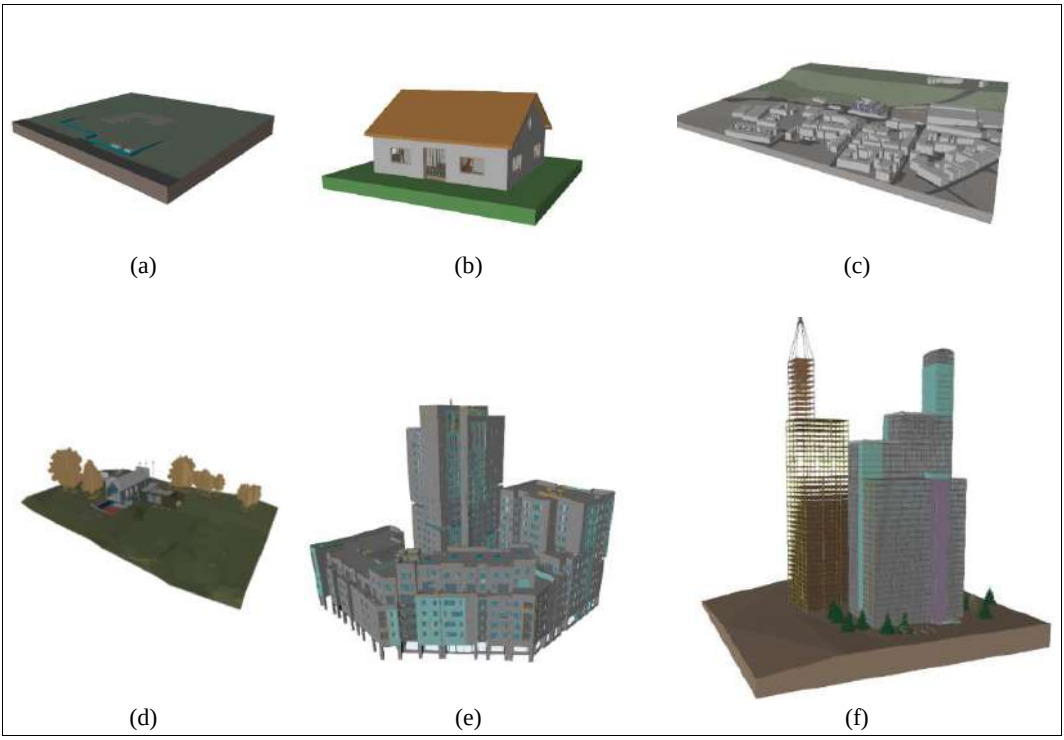


Рис. 3. Визуализация IFC-данных, использованных в вычислительных экспериментах
Fig. 3. Visualization of IFC data used in computational experiments

В качестве тестовых наборов данных для вычислительных экспериментов были использованы IFC данные разной сложности (рис. 3). Данные имелись в открытом доступе и были получены путем экспорта из Revit 2019 в формат SPF (применительно к IFC схеме часто называемом IFC форматом). Эксперименты проводились на компьютере с процессором Intel Core i7-7700 CPU, тактовая частота 3.60 ГГц, объем ОЗУ 32 Гб.

В первом эксперименте сравнивался размер файлов в указанных форматах (табл. 1) и в их архивированных вариантах (табл. 2). Видно, что размер файлов в формате JSON в среднем в два раза меньше, чем в формате STEP-XML, и приблизительно в 2 раза больше, чем в формате SPF. Архивирование нивелирует разницу между размерами файлов, однако общая тенденция не меняется: SPF файлы компактнее, чем JSON, а JSON файлы компактнее, чем XML.

Табл. 1. Размер исходных IFC данных, представленных в SPF, JSON и STEP-XML форматах
Table. 1. The size of original IFC data represented in SPF, JSON и STEP-XML formats

Набор IFC данных	SPF Kb	JSON Kb (% от размера SPF)	STEP-XML Kb (% от размера SPF)
a	952	2195 (231%)	4375 (460%)
b	2516	6054 (241%)	11556 (459%)
c	26274	56613 (215%)	113199 (431%)
d	40816	89794 (220%)	175689 (430%)
e	93160	237564 (255%)	379806 (408%)
f	173307	430514 (248%)	746400 (431%)

Табл. 2. Размер IFC данных после zip-архивирования
Table. 2. The size of IFC data after zip-archiving

Набор IFC данных	SPF zip Kb (% сжатия)	JSON zip Kb (% сжатия)	STEP-XML zip Kb (% сжатия)
a	175 (82%)	202 (91%)	237 (95%)
b	429 (83%)	545 (91%)	646 (94%)
c	4230 (84%)	4635 (92%)	5643 (95%)
d	6903 (83%)	7587 (92%)	8906 (95%)
e	16095 (83%)	19920 (92%)	23225 (94%)
f	30084 (83%)	35252 (92%)	39860 (95%)

Во втором эксперименте сравнивалось процессорное время, требуемое для считывания IFC данных в разных форматах в оперативную память компьютера. Для этого использовались парсеры JSON и XML файлов в составе среды Qt версии 5.15.1. Результаты в табл. 3 свидетельствуют о существенном ускорении работы при использовании формата JSON по сравнению с STEP-XML.

Табл. 3. Процессорное время считывания данных об изделии в форматах JSON и STEP-XML
Table. 3. CPU time for parsing product data in JSON and STEP-XML formats

Набор IFC данных	JSON CPU time (ms)	STEP-XML CPU time (ms)	Ускорение (JSON по отношению к STEP-XML)
a	38	146	3.84
b	129	564	4.37
c	824	2772	3.36
d	1281	4205	3.28
e	3124	9025	2.89
f	5626	31864	5.66

Таким образом, проведенные вычислительные эксперименты подтверждают преимущества использования формата JSON, порождаемого предложенными формальными правилами представления данных об изделии, по сравнению со стандартным форматом STEP-XML. Однако следует отметить, что в случаях, когда считывание данных в JSON сочетается с семантическим контролем, например, при получении данных PDM-системой и необходимой проверкой на соответствие EXPRESS схеме, выигрыш может быть нивелирован относительно высокими затратами на семантический контроль. В других случаях, когда такой контроль не требуется, например, при получении достоверных данных клиентом PDM-системы, скорость считывания оказывается важным фактором. С учетом существенного уменьшения объема пересылаемых данных при распределенной конфигурации PDM-системы преимущества представления данных об изделии в формате объектной нотации JSON становятся очевидными. Наконец, заметим, что стандартный SPF формат обеспечивает приблизительно такой же размер архивированных файлов, однако имеет другие

существенные недостатки, связанные со сложностью обработки и интерпретации данных web приложениями.

6. Заключение

Таким образом, в статье рассмотрены возможности применения формата JSON для представления данных об изделии, определяемых информационной схемой на языке EXPRESS. В статье представлены формальные правила продукции однозначной, не избыточной и независимой от программных реализаций объектной нотации JSON для данных об изделии. Приведены содержательные примеры, иллюстрирующие применение предложенных правил. Правила прошли апробацию в ходе разработки программного приложения, предназначенного для сериализации и трехмерной визуализации данных об изделии. Проведенные с его помощью вычислительные эксперименты подтвердили преимущества использования формата JSON по сравнению со стандартными форматами SPF и STEP-XML, что служит основанием для его более широкого применения при интеграции индустриальных программных приложений. Ожидается, что усилия по стандартизации предложенных правил и порождаемого ими файлового формата, прежде всего, в области информационного моделирования зданий и сооружений (Building Information Modeling), будут способствовать этому.

Отметим, что предложенные правила определяют способ представления данных об изделии, определяемых EXPRESS схемой, но не самой схемы. В некоторых работах [11] предпринимаются попытки представления IFC схемы в виде JSON документа, однако их следует признать довольно частными и ограниченными с учетом разнообразия конструкций языка EXPRESS. В частности, наличие алгебраических спецификаций для атрибутов DERIVED и разного рода правил WHERE, UNIQUE и GLOBAL не позволяет полноценно представить информационную схему в JSON и применить ее, например, для валидации данных [12]. Поэтому данные возможности не являлись предметом представленных исследований.

Список литературы / References

[1] ISO 10303. Industrial automation systems and integration — Product data representation and exchange.
[2] ISO 10303-11:2004. Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.
[3] ISO 10303-11:1994. Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.
[4] ISO 10303-21:2016. Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure.
[5] ISO 10303-28:2007. Industrial automation systems and integration — Product data representation and exchange — Part 28: Implementation methods: XML representations of EXPRESS schema and data.
[6] T. Bray. The JavaScript Object Notation (JSON) data interchange format. Internet Engineering Task Force (IETF), Request for Comments: 8259, 2014.
[7] D. Peng, L. Cao, W. Xu. Using JSON for data exchanging in web service applications. Journal of Computational Information Systems, 2011, vol. 7, no. 16, pp. 7552-7569.
[8] N. Nurseitov, M. Paulson et al. Comparison of JSON and XML Data Interchange Formats.
[9] Base64. https://en.wikipedia.org/wiki/Base64
[10] ISO 16739-1:2018. Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries.
[11] K. Afsari, Charles M. Eastman, Daniel Castro-Lacouture. JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange. Automation in Construction, 2017, vol. 77, pp. 24-51.
[12] V. Semenov, D. Ilyin, S. Morozov, O. Tarlapan. Effective consistency management for large-scale product data. // Journal of Industrial Information Integration, 2019, vol. 13, pp. 13-21.

Информация об авторах / Information about authors

Виталий Адольфович СЕМЕНОВ – доктор физико-математических наук, профессор, заведующий отделом системной интеграции и прикладных программных комплексов Института системного программирования им. В.П. Иванникова РАН с 2015 года. Сфера научных интересов: модельно-ориентированные методологии и инструменты программной инженерии для создания цифровых платформ и мультидисциплинарных программных комплексов, визуализация и компьютерная графика, технологии информационного моделирования в архитектуре и строительстве, проектное управление и календарно-сетевое планирование.

Vitaly Adolfovich SEMENOV – Doctor of Physical-Mathematical Sciences, Professor, Head of the Department of System Integration and Multi-disciplinary Applied Systems of the Ivannikov Institute for System Programming of the RAS since 2015. Research interests: model-driven methodologies and CASE toolkits for creating digital platforms and advanced applied systems, visualization and computer graphics, building information modeling, project management and scheduling.

Семен Васильевич АРИШИН – аспирант ИСП РАН. Научные интересы: системная интеграция, интероперабельность программного обеспечения и технологии информационного моделирования в архитектуре и строительстве.

Semen Vasilyevich ARISHIN – a postgraduate student of ISP RAS. Research interests: system integration, software interoperability and building information modeling.

Георгий Витальевич СЕМЕНОВ является студентом факультета информационных технологий и программирования Национального исследовательского университета «ИТМО». Его научные интересы включают технологии оптимистической репликации данных, математические методы сопоставления и слияния семантически сложных данных.

Georgii Vitalyevich SEMENOV is a student of the Information Technologies and Programming Faculty of the ITMO University. His research interests include technologies of optimistic replication of data, mathematical methods for differing and merging semantically rich datasets.

DOI: 10.15514/ISPRAS-2021-33(5)-2

CC BY

Формирование методологии разработки безопасного системного программного обеспечения на примере операционных систем

[1] *П.Н. Девянин, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>*
[1] *В.Ю. Тележников, ORCID: 0000-0002-6192-2856 <vtelezhnikov@astralinux.ru>*
[2,3,4,5] *А.В. Хорошилов, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*

[1] *ООО «РусБИТех-Астра»,*
117105, г. Москва, Варшавское ш., д. 26, стр.11
[2] *Институт системного программирования имени В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
[3] *Московский государственный университет имени М.В. Ломоносова,*
119991, Россия, Москва, Ленинские горы, д. 1
[4] *НИУ Высшая школа экономики,*
101978, Россия, г. Москва, ул. Мясницкая, д. 20
[5] *Московский физико-технический институт,*
141701, Россия, Московская область, г. Долгопрудный, Институтский пер., 9

Аннотация. Разработка безопасного системного программного обеспечения (ПО), на основе которого строятся сертифицированные средства защиты информации, достижение доверия к нему согласно требованиями нормативных документов отечественных регуляторов – крупная научно-техническая проблема. Возможным путем ее решения является формирование соответствующей методологии, которая должна включить передовые научные результаты в области информационной безопасности и системного программирования и отразить лучшие практики разработки такого ПО. В статье рассматриваются текущие результаты формирования этой методологии, которое осуществляется по следующим направлениям. Во-первых, это деятельность по развитию нормативной базы в области разработки и обеспечения доверия к безопасному системному ПО, включая создание профильных национальных стандартов. Во-вторых, разработка и верификация формальных моделей управления доступом, как основы механизмов защиты, входящих в состав системного ПО и составляющих его поверхность атаки. В-третьих, методы и технологии статического и динамического анализа программного кода системного ПО с учётом его специфики. В-четвертых, способы сбора и аналитической обработки данных, получаемых в ходе анализа программного кода системного ПО. Все направления формирования методологии иллюстрируются примерами ее практического применения и апробации при разработке ОС семейства Linux, в особенности сертифицированной по высшим классам защиты и уровням доверия операционной системы специального назначения Astra Linux Special Edition.

Ключевые слова: системное программное обеспечение; формальная модель управления доступом; верификация; статический и динамический анализ кода; операционная система; Astra Linux

Для цитирования: Девянин П.Н., Тележников В.Ю., Хорошилов А.В. Формирование методологии разработки безопасного системного программного обеспечения на примере операционных систем. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 25-40. DOI: 10.15514/ISPRAS–2021–33(5)–2.

Building a methodology for secure system software development on the example of operating systems

[1] *P.N. Devyanin, ORCID: 0000-0003-2561-794X <pdevyanin@astralinux.ru>*
[1] *V.Y. Telezhnikov, ORCID: 0000-0002-6192-2856 <vtelezhnikov@astralinux.ru>*
[2,3,4,5] *A.V. Khoroshilov, ORCID: 0000-0002-6512-4632 <khoroshilov@ispras.ru>*

[1] *RusBITech-Astra*
26, Varshavskoe, Moscow, 117105, Russia
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[3] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[4] *National Research University, Higher School of Economics*
20, Myasnitskaya Ulitsa, Moscow, 101978, Russia
[5] *Moscow Institute of Physics and Technology (State University),*
9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russian Federation

Abstract. System software is a cornerstone of any software system, so building secure system software in accordance with requirements of certification authorities and state-of-the-art practices is an important scientific and technical problem. One of possible approaches to cope with the problem is to build a methodology for secure system software development including advanced scientific technologies and industry best practices. The paper presents current results achieved in building such methodology in the following directions. The first one is regulatory framework improvement including development of GOST R specifications defining requirements to formal models of access control policies and their formal verification. The second direction is design and verification of formal models of corresponding security functional requirements. The third direction is application of new and well established technologies of static and run-time analysis of systems software. The considered technologies include static analysis, fuzzing, functional and unit testing as well as testing the system software against formal models of its functional security requirements. The forth direction is development of methods for acquisition of results of all kinds of the analysis and for its analytical processing. All the directions are illustrated by practical examples of application of the methodology to development of Astra Linux operating system distribution that is certified according to the highest evaluation assurance levels.

Keywords: security development lifecycle; access control; formal methods; verification; static analysis; fuzzing; operating systems; Astra Linux

For citation: Devyanin P.N., Telezhnikov V.Y., Khoroshilov V.V. Building a methodology for secure system software development on the example of operating systems**.** Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 25-40 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–2

1. Введение

Разработка безопасного системного программного обеспечения (ПО), достижение доверия к нему является самостоятельной крупной научно-технической проблемой. Это объясняется, во-первых, большими объемами кода, сложностью архитектуры и интерфейсов взаимодействия компонент такого ПО как внутри самой информационной системы, так и со средой ее функционирования. Во-вторых, системное ПО, например, операционные системы (ОС), по сути являются фундаментом, на котором строится безопасность и надежность работы всей системы в целом, в том числе ее прикладного ПО. В-третьих, на основе системного ПО создаются многие отечественные сертифицированные средства защиты информации (СЗИ), которые внедряются в рамках импортозамещения во многих информационных системах органов государственной власти, критической информационной инфраструктуры и коммерческих компаний Российской Федерации. Все это в свою очередь повышает ответственность всех участников процесса разработки системного ПО за

обеспечение его безопасности и доверия к нему, выбор используемых при этом технических решений, научных подходов и методов.

Непрерывно совершенствуемые с участием самих разработчиков ПО, представителей научного сообщества нормативные документы отечественных регуляторов в области разработки безопасного ПО и обеспечения доверия к нему также оказывают существенное влияние на процессы разработки системного ПО. Большое значение здесь имеют нормативные документы ФСТЭК России, в первую очередь базирующиеся на ГОСТ Р 56939-2016 [1] утвержденные Приказом № 76 от 02.06.2020 во второй редакции «Требования по безопасности информации, устанавливающие уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий» (далее – Требования доверия) [2] совместно с детализирующей их утверждённой ФСТЭК России 25.12.2020 второй редакцией «Методики выявления уязвимостей и недекларированных возможностей в программном обеспечении» (далее – Методика выявления уязвимостей) [3].

Эти нормативные документы, уже прошедшие несколько лет апробации при проведении сертификации СЗИ, включают широкий спектр требований, основанных на применении передовых результатов науки и реализующих их технологий. При этом востребованы научные теории, позволяющие с учётом сложности системного ПО моделировать и анализировать различные аспекты его безопасности.

В этом ПО большую роль играет управление доступом к контролируемым им ресурсам, например, для ОС это, как минимум, управление доступом субъектов доступа (процессов) к объектам доступа (файлам, каталогам и т. д.). Поэтому механизм управления доступом часто является основным из входящим в состав системного ПО механизмом защиты, формирующим поверхность атаки [3], и, следовательно, от того, как он реализован, во многом зависит безопасность всего системного ПО. В этой связи теория управления доступом и информационными потоками (скрытыми каналами) по памяти и по времени [4, 5] уже много лет является «локомотивом» разработки безопасного системного ПО. В рамках этой теории на математическом или формализованном (машиночитаемом) языках формируются и верифицируются вплоть до реализации в программном коде системного ПО формальные модели управления доступом [6], что полностью отвечает требованиям действующих нормативных документов.

Несомненно, важнейшей составляющей разработки любого безопасного ПО является обеспечение качества и доверия к его программному коду, которые достигаются в результате применения широкого спектра технологий его анализа и автоматизирующих их инструментальных средств, включая использование лучших международных практик по разработке безопасного ПО, таких как Secure Software Development Lifecycle (SSDL) [7]. Вместе с тем важно рассматривать эти практики с учётом специфики создания и применения системного ПО в основе сертифицированных СЗИ, часто включения в это ПО значительных объемов программного кода, заимствованного из свободного ПО (Open source). В частности, необходимо провести приоритизацию компонентов, исходя из их роли в реализации функциональных требований безопасности и поверхности атаки СЗИ [3], и адаптировать уровень проводимого анализа в зависимости от установленных приоритетов. Наиболее значимыми компонентами для системного ПО, как правило, являются те, что отвечают за реализацию механизма управления доступом. Поэтому анализ программного кода в наиболее тщательном виде должен быть сконцентрирован именно на них. В рамках этого необходимо применение статического анализа (в том числе чувствительного к контексту и путям выполнения) и динамического анализа (включая генерационное и мутационное фаззинг-тестирование, анализ трасс, выявление утечки чувствительных данных) программного кода [1], вобравшего в себя передовые результаты теорий системного программирования, методов верификации, символьных вычислений и даже искусственного интеллекта.

При этом также важно использовать возможности внедрения в таком системном ПО, в применяемых для обеспечения доверия к нему методах и инструментальных средствах результатов научных исследований отечественных специалистов, возможности отразить все перечисленное в новых или актуализируемых нормативных документах.

Причем для системного ПО важен комплексный подход в применении рассматриваемых технологий и методов обеспечения его безопасности, так как фрагментарность здесь (с учётом специфики и сложности такого ПО) приведет к наличию в нем уязвимостей и тем самым сведет на нет все прилагаемые усилия по достижению к нему доверия. Кроме того, значительные объемы кода и сложность системного ПО, как правило, приводят к большим объемам результатов его анализа, обработка которых за приемлемое время и приведение их к виду, позволяющему использовать эти результаты для оперативного исправления выявляемых ошибок, также является проблемой, для решения которой создаются соответствующие технологии и методы. Таким образом, уместно говорить о формировании у нас в стране методологии разработки безопасного системного ПО, направленной на комплексное решение практических проблем в этой предметной области в контексте выполнения требований соответствующих нормативных документов регуляторов.

Участие в этом процессе, с одной стороны, осуществляют представители научного сообщества, ведущих научных центров компетенций в области системного программирования и разработки безопасного ПО. С другой стороны, успех формирования методологии невозможен без использования опыта, технологий и компетенций отечественных разработчиков системного ПО, в том числе ОС и СУБД. При этом данный опыт будет тем ценнее, чем большему составу требований доверия удовлетворяет создаваемое системное ПО, и чем больше его разработчик самостоятельно внедрил и развивает необходим~~ые~~ для этого научно обоснованные технические решения, методы и реализующие их инструментальные средства.

В связи с этим в настоящей статье анализируются результаты совместных усилий по формированию и апробации рассматриваемой методологии специалистами Института системного программирования Российской академии наук им. В.П. Иванникова (ИСП РАН) и ООО «РусБИТех-Астра» (ГК Astra Linux) – разработчика сертифицированной по высшему 1 классу защиты (1 уровню доверия) в системе сертификации ФСТЭК России (а также сертифицированной по высоким классам защиты в системах сертификации ФСБ, Минобороны России и Республики Беларусь) операционной системы специального назначения (ОССН) Astra Linux Special Edition [8, 9].

Статья организована в соответствии с основными направлениями формирования методологии, непосредственное участие в реализации которых принимают авторы, и в первую очередь применительно к разработке сертифицированных ОС. В следующем разделе описывается являющаяся важной частью методологии деятельность по развитию нормативной базы в области разработки и обеспечения доверия к безопасному системному ПО, в том числе по созданию соответствующих национальных стандартов (ГОСТ). В разд. 3 приводится описание основных приемов и методов по разработке и верификации формальных моделей управления доступом, как основы механизмов защиты, входящих в состав системного ПО и составляющих его поверхность атаки. В разд. 4 анализируются особенности применения методов и технологий статического и динамического анализа программного кода системного ПО с учётом его специфики. В разд. 5 рассматривается ряд способов сбора и аналитической обработки данных, получаемых в ходе анализа программного кода системного ПО. Все разделы включают примеры практического применения и апробации формируемой методологии при разработке ОС семейства Linux, в особенности ОССН Astra Linux Special Edition. Заключение завершает статью, в нем приводятся возможные направления дальнейшего совершенствования методологии.

2. Развитие нормативной базы

Основополагающим национальным стандартом в области разработки безопасного ПО, в том числе системного, является ГОСТ Р 56939-2016 [1], в котором определены применяемые в этой области базовые технологии и методы. Для содержательного их наполнения, создания условий для эффективного использования на практике либо уже утверждены, либо готовится еще ряд национальных стандартов и нормативных документов (в том числе руководства по реализации мер по разработке безопасного ПО, по проведению статического или динамического анализа ПО и др.). Эта деятельность организуется техническим комитетом по стандартизации «Защита информации» (ТК 362), в состав которого наряду с другими лицензиатами ФСТЭК России, испытательными лабораториями входят ИСП РАН и ГК Astra Linux.

Это позволяет включать в новые нормативные документы положения, основанные на передовых научных исследованиях и при этом апробированные в результате разработки промышленного системного ПО. Однако здесь имеются трудности, которые требуют устранения в рамках формируемой методологии. Они заключаются в необходимости находить оптимальное сбалансированное сочетание таких положений. Во-первых, в нормативные документы необходимо включать требования, отражающие политику регуляторов в области разработки безопасного ПО, стимулирующие развитие отрасли. Во-вторых, целесообразно, чтобы эти положения основывались на лучших практиках в данной области и актуальных научных разработках. В-третьих, важно, чтобы положения нормативных документов проходили всестороннюю проверку на возможность их применения в полном объеме и с учётом специфики системного ПО, имеющего широкое внедрение в реальных информационных системах, включая СЗИ, а не только на макетах такого ПО или только в части стандартизируемых положений.

В этой связи для примера рассмотрим положения ГОСТ 59453.1-2021 «Защита информации. Формальная модель управления доступом. Часть 1. Общие положения» [10] и ГОСТ 59453.2-2021 «Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификации формальной модели управления доступом» [11], непосредственное участие в разработке которых принимали авторы.

Подготовка этих национальных стандартов стала необходимой после включения в 2017 г. компоненты доверия ADV_SPM.1 «Формальная модель политики безопасности» в утвержденные ФСТЭК России профили защиты ОС [12], начиная с 2 класса защиты, и в дальнейшем в 2018 г. включения требований о разработке формальной модели управления доступом и ее верификации с применением инструментальных средств, соответственно, в 4 и 3 уровни доверия первой редакции Требований доверия [2].

На первом шаге разработки этих национальных стандартов были проанализированы лучшие практики использования формальных моделей при реализации механизмов управления доступом в реальных СЗИ, в том числе ОС [9, 14]. Также были рассмотрены актуальные технологии верификации формальных моделей управления доступом с применением инструментальных средств [6, 15]. В результате было принято решение, что в ГОСТ 59453.1-2021 должны быть стандартизованы не конкретная модель или требования, которым должны удовлетворять все формальные модели, а с учётом текущего уровня развития технологий и опыта создания СЗИ, реализующих политики управления доступом, стандартизованы только критерии, которым должны соответствовать описания формальных моделей управления доступом, на основе которых разрабатываются эти СЗИ. При этом в ГОСТ 59453.2-2021 должны быть стандартизованы рекомендации по верификации с применением инструментальных средств формальных моделей управления доступом.

Далее были проанализированы классические и современные формальные модели управления доступом (например, изложенные в [4, 5]), подходы к их описанию. Это позволило определить научные основы разрабатываемых стандартов. В том числе, в ГОСТ 59453.1-2021 указано, что содержание описания формальной модели управления доступом должно быть дано на

математическом или формализованном (машиночитаемом) языках и включать описание состояний и правил перехода между состояниями абстрактного автомата, соответствующего политикам управления доступом, реализуемым моделируемым СЗИ. Это описание должно быть дано как минимум с использованием терминов: объект доступа (объект, контейнер, сущность), учётная запись пользователя, субъект доступа, доступ, право доступа, информационный поток. Также должны быть описаны условия безопасности, выполнение которых в абстрактном автомате указывает на реализацию заданных политик управления доступом. Уверенность в корректности формальной модели управления доступом должна быть достигнута математическим (формальным) доказательством того, что в ней не содержится противоречий, т. е. в абстрактном автомате выполняются условия безопасности.

Поскольку в существующих нормативных документах, технической и научной литературе часто приводятся несогласованные определения терминов, используемых при моделировании управления доступом (например, объект доступа, субъект доступа, учётная запись пользователя и др.), существенных для корректного изложения положений ГОСТ 59453.1-2021, то в этом стандарте потребовалось дать определения большинства таких терминов. При чем определения ряда терминов, например, политика мандатного контроля целостности или объект доступа, ассоциированный функционально с субъектом доступа, были стандартизованы впервые.

Для того чтобы, с одной стороны, дать критерии, которым должны соответствовать описания большинства формальных моделей управления доступом, а, с другой стороны, привести более детальные критерии для описания моделей для СЗИ, реализующих наиболее востребованные на практике виды политик управления доступом, структура ГОСТ 59453.1-2021 была сформирована следующим образом. Каждый раздел стандарта (посвященный, соответственно, описанию в рамках формальной модели управления доступом состояний абстрактного автомата, описанию правил его перехода из состояний в состояния и доказательству выполнения условий безопасности во всех его состояниях и при всех переходах из состояний в состояния) был разбит на пять частей. Первая часть такого раздела предназначалась для изложения критериев, которым должны удовлетворять описания всех формальных моделей управления доступом, а последующие части – для критериев описания моделей для СЗИ, реализующих, соответственно, дискреционное, ролевое, мандатное управление доступом и мандатный контроль целостности.

В свою очередь при изложении в ГОСТ 59453.2-2021 рекомендаций по верификации с применением инструментальных средств формальных моделей управления доступом наибольшее внимание было уделено описанию выполняемых при этом действий, в том числе:

- разработка и применение критериев выбора инструментальных средств верификации;
- перевод (при необходимости) описания модели из математического в формализованное (машиночитаемое) описание;
- автоматическое доказательство непротиворечивости формальной модели и выполнения заданных в ее рамках условий безопасности (условий верификации).

Для придания положениям ГОСТ 59453.1-2021 и ГОСТ 59453.2-2021 большей ясности и обеспечения большего удобства их использования при разработке и верификации формальных моделей управления доступом в текст первого стандарта почти к каждому пункту был добавлен пример, а во втором стандарте примеры собраны в приложении. В этих примерах на классическом математическом языке и на формализованном языке, поддерживаемом формальным методом Event-B [15], поясняются положения обоих стандартов.

Оба стандарта прошли апробацию на примере описания и верификации мандатной сущностно-ролевой ДП-модели управления доступом и информационными потоками в ОС семейства Linux (МРОСЛ ДП-модели) [5, 9], являющейся научной основой механизма управления доступом ОССН Astra Linux Special Edition.

Таким образом, при разработке ГОСТ 59453.1-2021 и ГОСТ 59453.2-2021, по мнению авторов, удалось, руководствуясь политикой регуляторов, учесть лучшие практики при описании и верификации формальных моделей управления доступом, а также современные научные теории и решения в этой области. Выработанные при этом апробированные подходы являются частью формируемой методологии разработки безопасного системного ПО.

3. Разработка и верификация формальных моделей управления доступом

Как уже было отмечено, управление доступом к контролируемым ресурсам является одной из ключевых функций системного ПО. Поэтому реализация механизма управления доступом с использованием соответствующей формальной модели согласно требованиям нормативных документов регуляторов становится неотъемлемой частью разработки безопасного системного ПО. При этом, как правило, интерфейсы этого механизма составляют поверхность атаки, а, значит, при его реализации следует применять весь спектр средств анализа его программного кода. Это в совокупности предоставляет возможность выработки и апробации комплексных научно обоснованных технологий и методов обеспечения доверия как к самому механизму управления доступом, так и распространения этого доверия на все системное ПО в целом.

Очевидно, что первым шагом при разработке формальной модели управления доступом должен быть анализ известных моделей, их теоретических основ, опыта реализации в системном ПО, в том числе СЗИ, с целью обоснования возможности использования какой-либо существующей модели, комбинации таких моделей, либо необходимости формирования новой модели.

В соответствии с ГОСТ 59453.1-2021 следующим шагом, когда разрабатывается новая формальная модель, является определение видов реализуемых СЗИ политик управления доступом. Это, с одной стороны, задает все дальнейшее содержание модели, с другой стороны, поскольку в большинстве реальных СЗИ используются несколько политик (например, часто дискреционное или ролевое управление доступом сочетается с мандатным управлением доступом), то важно при объединении в рамках модели нескольких политик сделать это не «механически», т.е. в способах такого объединения должны учитываться свойства каждой из политик, чтобы при реализации модели в СЗИ одни политики не противоречили другим. Примером проявления такого противоречия являются запрещенные политикой мандатного управления доступом информационные потоки (скрытые каналы) по времени от сущностей с большим уровнем конфиденциальности к сущностям с меньшим уровнем конфиденциальности, возникающие за счет использования параметров дискреционного или ролевого управления доступом [5].

Дальнейшим шагом является собственно разработка с использованием ГОСТ 59453.1-2021 самой формальной модели управления доступом и ее верификация с применением инструментальных средств согласно ГОСТ 59453.2-2021. Схема этого процесса и рекомендации по его выполнению изложены в указанных стандартах, а также в научной и учебной литературе (например, в [4-6]), вобравшей в себя уже почти полувековой опыт создания и верификации формальных моделей вплоть до их реализации в программном коде СЗИ. Самым трудным здесь является найти баланс между выразительностью формальной модели, обеспечением ее адекватности реальному СЗИ и чрезмерным усложнением модели, что сделает технически сложными ее верификацию и доказательство в ее рамках условий безопасности. Кроме того, при этом следует учитывать, что, как и моделируемое СЗИ, которое в большинстве случаев будет дорабатывается, так и формальная модель должна будет развиваться, поэтому потребуют многократного повторения и доказательства, и верификация.

Для примера рассмотрим опыт формирования МРОСЛ ДП-модели, на основе которой в подсистеме безопасности PARSEC [9] реализуется механизм управления доступом ОССН Astra

Linux Special Edition (в ОССН не применяются аналогичные заимствованные иностранные механизмы защиты, в том числе SELinux и AppArmor). В начале разработки этой модели в 2012 г. анализ существовавших на тот момент формальных моделей управления доступом показал отсутствие готовой модели, которую можно было использовать без существенной переработки. Поэтому было принято решение на основе семейства ДП-моделей и ряда классических моделей создать новую ДП-модель и объединить в ней мандатное управление доступом, мандатный контроль целостности и ролевое управление доступом. Целесообразность первой политики следовала из области применения ОССН как СЗИ, которое может обрабатывать информацию различных уровней конфиденциальности. Необходимость мандатного контроля целостности обосновывалась его эффективностью при защите целостности программной среды ОС, что подтверждалось практикой его внедрения с 2007 г. в ОС семейства Microsoft Windows. Ролевое управление доступом рассматривалось как мощная, гибкая и перспективная политика, с помощью которой легко выражается традиционное для ОС семейства Linux дискреционное управление доступом. При этом все три политики сочетались в МРОСЛ ДП-модели с учётом их свойств, не противореча друг другу, в том числе, не создавая условий для возникновения запрещенных информационных потоков по памяти или по времени.

Практически сразу МРОСЛ ДП-модель формировалась в двух нотациях: математической и формализованной на языке метода Event-B [6, 15]. Модель в первой нотации кроме использования опыта разработки классических моделей управления доступом, предоставляет возможность объективного анализа её корректности любым специалистом в области информационной безопасности, знакомым с языком математики. Например, может быть проверена корректность доказательств безопасности моделируемой системы при выполнении заданных в рамках модели условий безопасности. В формализованной нотации МРОСЛ ДП-модель верифицируется с применением инструментальных средств дедуктивно (с помощью Rodin) и по методу проверки моделей (с помощью ProB) [16], что также обеспечивает уверенность в её корректности.

МРОСЛ ДП-модель в обеих нотациях имеет иерархическое представление, состоящее из восьми уровней – четырех уровней для моделирования управления доступом непосредственно в ОССН Astra Linux Special Edition (первый – ролевого управления доступом, второй – мандатного контроля целостности, третий – мандатного управления доступом с информационными потоками по памяти, четвертый – мандатного управления доступом с информационными потоками по времени) и четырех уровней для решения аналогичной задачи в штатной для ОССН СУБД PostgreSQL, что обеспечивает согласованность механизмов управления доступом в ОССН и СУБД, а также возможность анализа безопасности информационных потоков (скрытых каналов) по памяти и по времени между сущностями ОССН и СУБД.

Переход к иерархическому представлению (первоначально модель формировалась монолитно без разделения её описания по видам политик управления доступом) был важным шагом в разработке модели, так как создавал предпосылки, во-первых, для работы с большим объемом описания модели (сейчас в математической нотации оно составляет более 500 страниц текста). Во-вторых, для удобства её представления и верификации в формализованной нотации, где для этого применялись техники пошагового уточнения (refinement) Event-B, давшие возможность разделить модель объемом более 32 тысяч строк кода на последовательность из восьми связанных между собой спецификаций, соответствующих уровням модели в математической нотации. В-третьих, помимо дедуктивной верификации поступательно по уровням спецификаций использовать метод проверки моделей, который из-за проблемы «комбинаторного взрыва» чрезвычайно чувствителен к объему верифицируемой модели. В-четвертых, разделение модели на уровни соответствует подходу по её реализации непосредственно в программном коде подсистемы безопасности PARSEC ОССН Astra Linux Special Edition. В итоге, метод основанный на применении иерархического представления и использовании для этого одновременно математической и формализованной нотаций не только

оправдал себя на примере ОССН, его целесообразно рекомендовать для разработки других сложных формальных моделей управления доступом.

Также важно отметить, что с использованием описания МРОСЛ ДП-модели в формализованной нотации для демонстрации корректности её реализации непосредственно в программном коде ОССН Astra Linux Special Edition на языке ACSL разрабатываются спецификации функций подсистемы безопасности PARSEC, после чего происходит их дедуктивная верификация с применением инструментального средства Frama-C [17]. В результате корректность наиболее существенной для безопасности ОССН части её программного кода доказывается математически, а уже далее, как это будет показано в последующих разделах настоящей статьи, этот код обрабатывается средствами статического и динамического анализа. Таким образом, делается переход от разработки формальной модели с применением в основном математических методов к использованию методов системного программирования для анализа и обеспечения доверия к программному коду, реализующему в ОССН поверхность атаки, что является важной частью формируемой методологии разработки безопасного системного ПО.

4. Статический и динамический анализ программного кода

Согласно ГОСТ Р 56939-2016 [1] статический и динамический анализ программного кода определены как основные меры, используемые при разработке безопасного ПО. Конкретное описание того, как эти меры должны применяться для достижения доверия к такому ПО, изложено в Методике выявления уязвимостей [3]. В том числе, это автоматизированные технологии и методы статического анализа, чувствительного к контексту и путям выполнения и реализующего метод статического символьного выполнения, а для динамического анализа – это в первую очередь технологии и методы мутационного (включая реализацию генетических алгоритмов) и генерационного (включая динамическое символьное выполнение) фаззинг-тестирования, а также анализ трасс выполнения тестируемого ПО и выявление уязвимостей, связанных с утечкой чувствительных данных (например, паролей).

Для системного ПО, такого как ОС или СУБД, ввиду сложности его архитектуры, интерфейсов взаимодействия его компонент, объема кода, часто исчисляемого десятками миллионов строк, адаптация технологий и методов анализа программного кода системного ПО, организация соответствующего технологического процесса имеют существенное значение.

В Методике выявления уязвимостей при осуществлении статического и динамического анализа требуется акцентировать внимание на программном коде, реализующем поверхность атаки. В случае системного ПО определение адекватной поверхности атаки чрезвычайно важно. С одной стороны, включение в нее максимального состава компонент системного ПО и реализуемых ими интерфейсов может сделать задачу полного анализа их программного кода не выполнимой на практике. С другой стороны, необоснованное уменьшение поверхности атаки наверняка приведет к пропуску существенных уязвимостей в таком ПО. В связи с этим для определения поверхности атаки системного ПО следует исходить из всестороннего исследования архитектуры системного ПО, его механизмов защиты и формирования соответствующей модели нарушителя. Здесь обоснованным научно и подтвержденным практикой можно считать подход, предполагающий определение механизма управления доступом, других связанных с ним механизмов защиты (например, направленных на создание изолированной программной среды) и их интерфейсов в качестве основы поверхности атаки системного ПО. Это подтверждается тем, что прежде чем нарушитель сможет проэксплуатировать уязвимость в некоторой компоненте реализующего развитые механизмы защиты системного ПО, ему в большинстве случаев потребуется преодолеть их, т. е. «пробить» поверхность атаки. Аналогично, если даже нарушителю удастся воспользоваться уязвимостью функционирующего в среде системного ПО прикладного ПО, то для дальнейшего распространения атаки на всю систему, захвата контроля над ней, иных атакующих действий,

например, реализации утечки конфиденциальных данных, ему, как правило, потребуется преодолеть механизмы защиты системного ПО.

Рассмотрим опыт применения статического и динамического анализа программного кода при разработке безопасного системного ПО на примере ОССН Astra Linux Special Edition. Для автоматизации её процессов сборки, тестирования и верификации используется практика непрерывной интеграции (continuous integration) в инструментальной среде GitLab. В этой среде развёрнут специальный компьютерный полигон, названный «Стендом доверия», являющийся комплексом современных инструментальных средств применения научно обоснованных технологий безопасной разработки и обеспечения доверия к ОССН, а также сбора и аналитической обработки выявленных ошибок в программном коде и результатов их устранения.

В качестве основных инструментальных средств статического анализа кода ОССН на «Стенде доверия» применяются cppcheck [18], Clang Static Analyzer [19] и разработанное ИСП РАН инструментальное средство Svace [20].

С учётом специфики реализации основного механизма управления доступом PARSEC в пространстве ядра ОССН для средств статического анализа специалистами ГК Astra Linux уточняются существующие и создаются новые модули тестирования (checkers). Так, например, принимаются во внимание соглашения о вызовах и передаче параметров в функции как непосредственно в ядре ОССН, так и при обращении к модулям ядра, в том числе PARSEC, что значительно сокращает количество «условно ложных» предупреждений об ошибках вида «разыменование нулевого указателя» и им подобных. При этом делаются проверки отсутствия нарушающих данные соглашения путей выполнения, приводящих к вызову потенциально небезопасной функции в условиях отсутствия внутри нее проверок корректности переданных аргументов. Для обеспечения качества проводимого анализа системного ПО, сопоставимого с результатами анализа прикладного ПО, для соответствующих модулей тестирования разрабатываются спецификации функций работы с памятью в пространстве ядра ОССН. Например, в набор спецификаций инструментального средства Svace добавлены описания специализированных функций класса kvmalloc (kvfree), что позволяет отслеживать использование динамически выделяемой памяти и выявлять ошибки при работе с ней, в том числе приводящие к ее «утечке».

При выборе средств динамического анализа ОССН Astra Linux Special Edition для достижения наибольшей эффективности фаззинг-тестирования учитываются условия функционирования каждого из её тестируемых компонентов, в первую очередь подсистемы безопасности PARSEC. Поэтому в качестве основного инструментального средства на «Стенде доверия» используется Syzkaller [21], с применением которого открытым сообществом выявлено множество уязвимостей в различных версиях ядер ОС семейства Linux.

Syzkaller обладает достаточно широкими возможностями и реализует эффективные методы (алгоритмы) фаззинг-тестирования: генерационные алгоритмы для составления начальных целевых тестовых данных на основе предоставленных ему описаний системных вызовов на специализированном языке; различные алгоритмы мутации сгенерированных тестовых значений с целью достижения большего покрытия; генетические алгоритмы для отбора наиболее подходящих («интересных») экземпляров набора сгенерированных значений (корпусов входных данных), например тех, которые либо расширяют покрытие, в том числе в заданном направлении, либо чаще приводят к «падениям». Помимо этого, Syzkaller также обеспечивает возможность минимизации корпуса с целью сокращения содержащихся в нем данных, что значительно повышает эффективность фаззинг-тестирования и позволяет сосредоточиться на участках кода (модулях), непосредственно составляющих поверхность атаки. Благодаря этим возможностям апробированы подходы по уточнению существующих и описанию дополнительных прототипов для генерации на их основе начальных данных фаззинг-тестирования, нацеленных непосредственно на подсистему безопасности PARSEC, что позволило добиться существенного увеличения покрытия (до 70% по базовым блокам)

наиболее «чувствительного» кода с одновременным кратным увеличением сгенерированных для этого наборов тестовых данных.

Для реализации метода регрессионного фаззинг-тестирования авторами развивается инструмент Syzcrawler, который обеспечивает автоматизированное тестирования обнаруженных ранее ошибок (падений) на стендах фаззинга с новыми (доработанными) ядрами ОССН Astra Linux Special Edition с целью подтверждения их успешного устранения. Помимо тестирования ошибок, найденных с помощью Syzkaller, для которых фаззером были построены файлы для воспроизведения с помощью встроенной в него утилиты syz-repro, Syzcrawler поддерживает воспроизведение ошибок по файлам журнала, которые Syzkaller составляет для каждого обнаруженного падения. Также в рамках работы над повышением эффективности регрессионного тестирования и его автоматизацией проходит апробацию метод обнаружения ошибок по шаблонам, составленным на основе уже выявленных в результате фаззинга падений, что позволяет находить типовые ошибки, не дожидаясь результатов динамического анализа.

С целью фаззинг-тестирования отдельных драйверов или приложений пользовательского пространства, в том числе входящих в подсистему безопасности PARSEC, на «Стенде доверия» применяются положительно зарекомендовавшие себя средство Amrican Fuzzy Lop [22] (далее – AFL) и комплекс динамического анализа программ Crusher [23], разрабатываемый ИСП РАН. AFL – фаззер с открытым исходным кодом, в основе которого лежат инструментация кода на этапе компиляции программы и применение генетических и мутационных алгоритмов для поиска новых тестовых случаев и расширения покрытия анализируемого кода. Crusher – это инструментальное средство, реализующее фаззинг-тестирование с использованием генерационных и генетических алгоритмов и динамического символьного выполнения.

Для фаззинг-тестирования приложений пользовательского пространства подсистемы безопасности PARSEC для AFL и Crusher были написаны дополнительные «программы-обертки» (далее – обертки), которые в свою очередь передают полученные от фаззера данные в целевую программу. При этом важно, чтобы обертка принимала любые искаженные значения и имела узкую цель, по этой причине при тестировании было принято решение не использовать одну обертку для целого набора приложений, разделяя между ними входной буфер, а создавать для каждого тестируемого приложения свою. Данный подход позволил значительно повысить эффективность фаззинга и скорость обнаружения новых путей исполнения, что в сочетании с применением составленных для каждого тестируемого компонента словарей входных данных обеспечило высокие результаты покрытия за короткое время работы программного комплекса тестирования на «Стенде доверия». В результате апробации данных инструментальных средств покрытие приложений (библиотек) пользовательского пространства, входящих в PARSEC, достигает 60-80% (в отдельных случаях и выше), а параллельное применение AFL и Crusher обеспечивают повышенную эффективность тестирования и большую уверенность в качестве и достоверности полученных результатов. Следует отметить, что в соответствии с Методикой выявления уязвимостей достижение такого покрытия может характеризовать процедуру фаззинг-тестирования как эффективную.

Для требуемого согласно Методике выявления уязвимостей анализа помеченных данных (taint-анализа) применяется созданное ИСП РАН инструментальное средство Блесна. Оно, например, эффективно позволяет осуществлять поиск путей несанкционированной передачи между программными модулями ОССН пароля пользователя (вместо его затирания в памяти).

Таким образом, применение на «Стенде доверия» совокупности рассмотренных методов, технологий и реализующих их инструментальных средств статического и динамического анализа кода ОССН Astra Linux Special Edition к составляющим поверхность атаки модулям её подсистемы безопасности PARSEC, позволяют обеспечивать высокий уровень доверия к ней на протяжении всего жизненного цикла её разработки, устраняя большинство программных ошибок на его ранних этапах. Это создает условия для распространения доверия далее на весь программный код ОССН.

5. Аналитическая обработка результатов анализа

С учетом специфики системного ПО при проведении анализа его программного кода одной из основных проблем является аналитическая обработка и интерпретация результатов такого анализа. На объёмах программного кода, типичных для системного ПО, а это миллионы строк, инструментальные средства выдают тысячи, а иногда, десятки тысяч предупреждений о возможных ошибках, большая часть из которых, как правило, оказываются ложными или не учитывающими особенности анализируемого кода, что наиболее актуально для модулей или драйверов ядра ОС. Возложение обязанностей по анализу этих предупреждений об ошибках непосредственно на программистов-разработчиков системного ПО резко снизит эффективность их работы. В связи с этим необходимы технологии и методы, реализующие их инструментальные средства, автоматизирующие обработку этих предупреждений, чтобы программистам-разработчикам информация предоставлялась с учётом степени критичности ошибки в виде, максимально точно указывающем на место ее возникновения.

Поскольку большинство инструментальных средств анализа программного кода имеют индивидуальные часто несовместимые форматы представления результатов своей работы, то, конечно, для дальнейшей аналитической обработки этих результатов требуется реализация средств их приведения к единому формату в некоторой общей базе данных. При этом для повышения эффективности аналитической обработки желательно оснащение этой базы данных интерфейсами, позволяющими визуализировать полученные результаты анализа. Например, проводить разметку результатов анализа, сопоставление таких результатов, полученных либо в разное время, либо от разных инструментальных средств.

При организации аналитической обработки следует иметь ввиду, что для системного ПО (особенно ОС) проведение динамического анализа программного кода и главное интерпретация его результатов является более трудоёмкой и технически сложной задачей, чем для статического анализа кода. С одной стороны, общее число сообщений об ошибках (сбоях в работе системного ПО), выдаваемых инструментальными средствами (например, при фаззинг-тестировании) за приемлемое время тестирования, существенно меньше, чем при проведении статического анализа кода. С другой стороны, практически весь дальнейший анализ этих ошибок с большим трудом автоматизируется и в большинстве случаев предполагает непосредственную их обработку («ручной» анализ) аналитиком. Именно поэтому средства динамического анализа чаще всего применяются непосредственно после выхода системного ПО на стадию тестирования, когда ошибки, обнаруженные статическими анализаторами устранены.

Кроме того, в случае, когда большие объемы программного кода системного ПО являются заимствованными из свободного ПО (например, когда оно разрабатывается на основе ОС семейства Linux), имеет большое значение применение технологий и методов статистической обработки результатов анализа. Это позволит выявлять аномальные изменения в таком заимствованном коде, что может указывать на необходимость более детального анализа выявленных в нем ошибок.

Применительно к ОССН Astra Linux Special Edition в рамках рассмотренного в предыдущем разделе специального компьютерного полигона – «Стенда доверия» развернута база данных, названная «Базой данных доверия». В ней собираются данные от всех инструментальных средств анализа программного кода, осуществляется их аналитическая обработка, для чего специалистами ГК Astra Linux разрабатываются соответствующие средства автоматизации, в том числе включающие интерфейсы визуализации результатов такой обработки.

Для обеспечения возможности реализации единого интерфейса работы с предупреждениями, полученными от различных инструментальных средств статического, динамического и других видов анализа, с привязкой к исходному коду тестируемых компонентов, осуществляется автоматизированное сопоставление полученных результатов по значительному числу параметров, включая тип и шаблон описания ошибки, уровень достоверности, место сработки,

последовательность вызовов, приводящих к ошибке, а так же другие значимые характеристики, перечень которых постоянно расширяется. Помимо этого, в функциональные возможности «Базы данных доверия» входит применение алгоритмов машинного обучения для определения уровня достоверности получаемых результатов анализа. Обучение проводится на достаточно объемных наборах данных, предварительно обработанных аналитиками на предмет выявления ложных срабатываний и корректировки, при необходимости, степени критичности предупреждений, действительно способных привести к возникновению ошибок. Такой подход обеспечивает возможность применения накопленных результатов при анализе как еще не затронутых компонентов прикладного и системного ПО, так и новых версий ранее протестированных пакетов, в исходных кодах которых возможны существенные изменения, как например при переходе на новую версию ядра ОССН.

В ходе аналитической обработки при определении уровня критичности выявленных инструментальными средствами ошибок в программном коде (и, как следствие, приоритетности их устранения) учитывается наличие собственных реализованных в ОССН средств защиты в её подсистеме безопасности PARSEC, формирующей поверхность атаки. Т.е. такие ошибки в PARSEC проходят через самый тщательный анализ. Вместе с тем в «Базе данных доверия» осуществляется сопоставление ошибок, одновременно выявленных средствами статического и динамического анализа, что упрощает процесс оценивания их достоверности и критичности. Для некоторых типов ошибок апробируются подходы, позволяющие на основе результатов, полученных статическими анализаторами, осуществлять автоматизированную генерацию наборов входных данных для средств динамического анализа, выполняющих дополнительную проверку воспроизводимости найденных потенциальных ошибок.

При этом разрабатываются подходы по обработке результатов статического анализа программного кода подсистемы безопасности PARSEC в сочетании с методами и технологиями дедуктивной верификации этого кода с применением инструментального средства Frama-C. В особенности той части кода, которая непосредственно реализует функции управление доступом согласно МРОСЛ ДП-модели. Задание спецификаций таких функций на ACSL и дальнейшая их верификация взаимно дополняют и развивают технологии и методы, используемые при обработке результатов статического анализа.

В итоге программисты-разработчики ОССН Astra Linux Special Edition получают только достоверную, верифицированную аналитиками информацию об ошибках с разъяснением причин возникновения, и, возможно, с предложением способов их устранения. Таким образом, по сути требование отсутствия (минимизации до определённой степени критичности) выявляемых инструментальными средствами статического и динамического анализа ошибок в программном коде становится для разработчика ОССН стандартом качества.

6. Заключение и направления развития

В настоящей статье проанализированы основные направления формирования методологии разработки безопасного системного ПО, в том числе деятельность по развитию соответствующей нормативной базы, включая профильные национальные стандарты, создание и верификация формальных моделей управления доступом, методы и технологии статического и динамического анализа программного кода системного ПО, аналитической обработки данных, получаемых в ходе анализа программного кода системного ПО. Этот процесс формирования методологии еще далек от своего завершения, требуют совершенствования многие составляющие ее технологии и методы, в особенности направленные на определение глубины поверхности атаки и на анализ программного кода системного ПО с учётом его объемов и сложности. Кроме того, требует расширения спектр поддерживаемых инструментальными средствами анализа кода языков программирования и аппаратных

платформ, в первую очередь базирующихся на отечественных процессорах таких, как «Эльбрус» и «Байкал».

В то же время уже имеющийся опыт успешной апробации методологии при создании СЗИ на основе ОС семейства Linux, включая ОССН Astra Linux Special Edition, подтверждает правильность выбранных направлений её формирования и востребованность в дальнейшем при разработке безопасного системного ПО и обеспечении доверия к нему.

Список литературы / References

[1] ГОСТ Р 56939-2016. Защита информации. Разработка безопасного программного обеспечения. Общие требования. / GOST R 56939-2016. Information protection. Secure Software Development. General requirements. Federal Agency for Technical Regulation and Metrology, 2016 (in Russian).

[2] Выписка из Требований по безопасности информации, утвержденных приказом ФСТЭК России от 2 июня 2020 г. N 76 / Excerpts from Requirements for information security approved by FSTEK Russia order #76 of 2nd June 2020. Available at: https://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty-po-sertifikatsii/120-normativnye-dokumenty/2126-vypiska-iz-trebovanij-po-bezopasnosti-informatsii-utverzhdennykh-prikazom-fstek-rossii-ot-2-iyunya-2020-g-n-76, accessed 14.11.2021 (in Russian).

[3] Информационное сообщение ФСТЭК России от 10.02.2021 № 240/24/647 / Informational message of FSTEK Russia of 10th February 2021 #240/24/647. Available at: https://fstec.ru/normotvorcheskaya/informatsionnye-i-analiticheskie-materialy/2171-informatsionnoe-soobshchenie-fstek-rossii-ot-10-fevralya-2021-g-n-240-24-647, accessed 14.11.2021 (in Russian).

[4] Bishop M. Computer Security: art and science. Addison-Wesley Professional, 2002. 1084 p.

[5] Девянин П.Н. Модели безопасности компьютерных систем. Управление доступом и информационными потоками. Учебное пособие для вузов. М., Горячая линия – Телеком, 2020 г.. 352 стр. / P.N. Devyanin. Security models of computer systems. Control for access and information flows. Hotline-Telecom, 2013, 338 p. (in Russian).

[6] Девянин П.Н., Ефремов Д.В. и др. Моделирование и верификация политик безопасности управления доступом в операционных системах. М., Горячая линия – Телеком, 2019 г., 214 стр. / P.N. Devyanin, D.V. Efremov et al. Modeling and verification of access control access policies in operating systems. Hotline-Telecom, 2019, 214 p. (in Russian).

[7] Microsoft Security Development Lifecycle. Available at: https://www.microsoft.com/en-us/securityengineering/sdl, accessed 14.11.2021.

[8] Операционная система специального назначения Astra Linux Special Edition / Astra Linux Special Edition operating system. Available at: https://astralinux.ru/products/astra-linux-special-edition, accessed 14.11.2021.

[9] П.В. Буренин, П.Н. Девянин и др. Безопасность операционной системы специального назначения Astra Linux Special Edition. Учебное пособие для вузов. М., Горячая линия – Телеком, 2019 г., 404 стр. / P.V. Burenin, P.N. Devyanin et al. Information security with Astra Linux Special Edition. Hotline-Telecom, 2019, 404 p. (in Russian).

[10] ГОСТ Р 59453.1-2021. Защита информации. Формальная модель управления доступом. Часть 1. Общие положения. / GOST R-59453.1-2021. Information protection. Formal access control model. Part 1. General principles (in Russian).

[11] ГОСТ Р 59453.2-2021. Защита информации. Формальная модель управления доступом. Часть 2. Рекомендации по верификация формальной модели управления доступом / GOST R-59453.2-2021 «Information protection. Formal access control model. Part 2. Recommendations on verification of formal access control model» (in Russian).

[12] Документы по сертификации средств защиты информации и аттестации объектов информатизации по требованиям безопасности информации / Documents for certification of information security software and attestation of information systems according to information security requirements. Available at: http://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty-po-sertifikatsii/120-normativnye-dokumenty, accessed 14.11.2021 (in Russian).

[13] Информационное сообщение ФСТЭК России от 29.03.2019 № 240/24/1525 / Informational message of FSTEK Russia 29th March 2021 #240/24/1525. Available at: https://fstec.ru/component/attachments/download/2286, accessed 14.11.2021 (in Russian).

[14] Буренков В.С., Кулагин Д.А. Модель мандатного контроля целостности в операционной системе KasperskyOS. Труды ИСП РАН, том 32, вып. 1, 2020 г, стр. 27-56 / Burenkov V.S., Kulagin D.A. A Mandatory Integrity Control Model for the KasperskyOS Operating System. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 1, 2020. pp. 27-56 (in Russian). DOI: 10.15514/ISPRAS-2020-32(1)-2.

[15] Abrial J.-R., Butler M. et al. Rodin: An Open Toolset for Modelling and Reasoning in Event-B. International Journal on Software Tools for Technology Transfer, vol. 12, no. 6. 2010, pp. 447-466.

[16] Девянин П.Н., Леонова М.А. Приёмы по доработке описания модели управления доступом ОССН Astra Linux Special Edition на формализованном языке метода Event-B для обеспечения её автоматизированной верификации с применением инструментов Rodin и ProB. Прикладная дискретная математика, no. 52, 2021 г., стр. 83-96 / Devyanin P.N., Leonova M.A. The techniques of formalization of OS Astra Linux Special Edition access control model using Event-B formal method for verification using Rodin and ProB. Applied discrete mathematics, no. 52, 2021, pp. 83–96 (In Russian).

[17] Kirchner F., Kosmatov N. et al. Frama-C: a software analysis perspective. Formal Aspects of Computing, vol. 27, issue 3, 2015, pp. 573-609.

[18] Cppcheck. A tool for static C/C++ code analysis. Available at: http://cppcheck.sourceforge.net, accessed 14.11.2021.

[19] Clang Static Analyzer. Available at: https://clang-analyzer.llvm.org/, accessed 14.11.2021.

[20] Статический анализатор Svace. URL: http://www.ispras.ru/technologies/svace / SVACE static analyzer. Available at: https://www.ispras.ru/en/technologies/svace/, accessed 14.11.2021.

[21] Syzkaller project. Available at: https://github.com/google/syzkaller, accessed 14.11.2021.

[22] American Fuzzy Loop. Available at: https://github.com/google/AFL, accessed 14.11.2021.

[23] Комплекс динамического анализа программ Crusher. URL: https://www.ispras.ru/technologies/crusher / ISP Crusher: a dynamic analysis toolset. Available at: https://www.ispras.ru/en/technologies/crusher/, accessed 14.11.2021.

Информация об авторах / Information about authors

Петр Николаевич ДЕВЯНИН – член-корреспондент Академии криптографии России, доктор технических наук, профессор, научный руководитель ООО "РусБИТех-Астра" (ГК Astra Linux). Область интересов: теория информационной безопасности, формальные модели безопасности компьютерных систем.

Petr Nikolaevich DEVYANIN – Doctor of Technical Sciences, corresponding member of Russian Academy of Cryptography, professor, scientific director in RusBITech-Astra (Astra Linux). Field of Interest: information security theory, formal security models of computer systems.

Владимир Юрьевич ТЕЛЕЖНИКОВ – кандидат технических наук, начальник отдела научных исследований ООО "РусБИТех-Астра" (ГК Astra Linux). Область интересов: формальные модели безопасности компьютерных систем, методы статического и динамического анализа программного кода, операционная система Linux.

Vladimir Iurevich TELEZHNIKOV, Ph.D in Technical Science, Head of Research Department of the RusBITech-Astra (Astra Linux). Field of Interest: formal security models of computer systems, methods of static and dynamic analysis of program code, Linux operating system.

Алексей Владимирович ХОРОШИЛОВ, ведущий научный сотрудник, кандидат физико-математических наук, директор Центра верификации ОС Linux в ИСП РАН, доцент кафедр системного программирования МГУ, ВШЭ и МФТИ. Основные научные интересы: методы проектирования и разработки ответственных систем, формальные методы программной инженерии, методы верификации и валидации, тестирование на основе моделей, методы анализа требований, операционная система Linux.

Alexey Vladimirovich KHOROSHILOV, Leading Researcher, Ph.D. in Physics and Mathematics, Director of the Linux OS Verification Center at ISP RAS, Associate Professor of System Programming Departments at Moscow State University, the Higher School of Economics, and Moscow Institute of Physics and Technology. Main research interests: design and development

methods for critical systems, formal methods of software engineering, verification and validation methods, model-based testing, requirements analysis methods, Linux operating system.

DOI: 10.15514/ISPRAS-2021-33(5)-3

Онтологическое обеспечение управления рисками информационной безопасности

И. Бубакар, ORCID: 0000-0003-1094-8666 <Ibbatoure10@gmail.com>
М.Б. Будько, ORCID: 0000-0001-7054-5709 <mbbudko@itmo.ru>
М.Ю. Будько, ORCID: 0000-0002-1444-277X <mbudko@itmo.ru>
А.В. Гирик, ORCID: 0000-0002-4021-7605 <avg@itmo.ru>
Университет ИТМО,
197101, Россия, Санкт-Петербург, Кронверкский пр., д. 49, лит. А

Аннотация. В итоге выполнения работы, ориентированной на повышение эффективности системы информационной безопасности за счет разработки онтологической модели и подхода на ее основе к обеспечению управления рисками информационной безопасности (ИБ), был получен гибкий результат, который призван обеспечить повышение эффективности системы защиты информации за счет снижения временных затрат на принятие управленческих решений. В конце работы проведен сравнительный анализ существующих подходов и методик к управлению рисками ИБ и описываемый подход. На основе разработанной онтологии и подхода на её основе могут быть созданы высокоинтеллектуальные системы управления рисками ИБ и системой защиты информации в целом.

Ключевые слова: информационный риск; управление рисками; защита информации; информационная безопасность; принятие управленческих решений; поддержка принятия решений; динамическая экспертная система; онтология

Для цитирования: Бубакар И., Будько М.Б., Будько М.Ю., Гирик А.В. Онтологическое обеспечение управления рисками информационной безопасности. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 41-64. DOI: 10.15514/ISPRAS–2021–33(5)–3.

Ontological support of information security risk management

I. Boubacar, ORCID: 0000-0003-1094-8666 <Ibbatoure10@gmail.com>
M.B. Budko, ORCID: 0000-0001-7054-5709 <mbbudko@itmo.ru>
M.Yu. Budko, ORCID: 0000-0002-1444-277X <mbudko@itmo.ru>
A.V. Guirik, ORCID: 0000-0002-4021-7605 <avg@itmo.ru>
ITMO University,
49, bldg. A, Kronverksky Pr., St. Petersburg, Russia, 197101

Abstract. As a result of the work focused on improving the efficiency of the information security system through the development of an ontological model and an approach based on it to ensure information security (IS) risk management, a flexible result was obtained, which is designed to ensure an increase in the efficiency of the information security system by reducing the time spent on managerial decision-making. At the end of the work, a comparative analysis of existing approaches and techniques to information security risk management and the described approach was carried out. Based on the developed ontology and approach, highly intelligent information security risk management systems and the information security system can be created on its basis.

Keywords: information risk; risk management; information security; information safety; management decision-making; decision support; dynamic expert system; ontology

For citation: Boubacar I., Budko M.B., Budko M.Yu., Guirik A.V. Ontological support of information security risk management. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 41-64 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–3

1. Введение

Информационная безопасность (ИБ) является одним из важнейших процессов любой серьёзной современной организации. Поэтому строжайший контроль над ИБ и управление ею в режиме реального времени являются экзистенциональной необходимостью для современной организации. Сегодня ядром ИБ являются анализ и управление информационными рисками (ИР). Поэтому изучение, создание и внедрение средств, автоматизирующих процесс анализа и управления ИР, являются весьма актуальными задачами для любого специалиста в области ИБ. Ведь это позволяет здорово сократить время на принятие управленческих решений (ПУР).

Одним из актуальнейших и многообещающих направлений при решении проблемы автоматизации процессов анализа и управления, как ИР, так и ИБ в целом является применение экспертных систем (ЭС) поддержки принятия решений (ППР), позволяющих выполнить львиную долю функций и рутинных операций, выполняемых обычно персоналом, что сильно сократит время на принятие правильных необходимых управленческих решений. В условиях, когда высокая: динамика изменения требований по ИБ, динамика изменения методологических подходов к ИБ, динамика изменения мнений экспертов по ИБ, динамика изменения факторов, влияющих на информацию, логичным и оправданным является применение динамических экспертных систем (ДЭС). Но имплементация систем управления ИБ на базе ДЭС ППР затруднена нехваткой научно обоснованного методического и методологического аппарата, учитывающего не только требования и специфику управления ИБ, в том числе мнение экспертов компании, но и важную специфику реализации инфраструктуры информационных технологий (ИТ).

ДЭС это, прежде всего, системы, основанные на экспертных знаниях в предметной области (ПрО). В настоящее время, для формализации ПрО используются онтологии. А динамика изменения требований, методологических подходов, мнений экспертов, влияющих на информацию факторов и т.д., можно просто рассматривать как обучение онтологии. Обучить онтологию означает просто корректировать в ней знания или дополнить её новыми знаниями. Сегодня мы знаем, что процесс обучения онтологии может быть полностью автоматизирован различными техниками (правилами вывода, машинным обучением, прецедентным подходом и т.д. и т.п.). Следовательно, можем рассматривать ДЭС, основанные на онтологию, т.е. онтологические ДЭС (ОДЭС). Выше сказанное и определило нашу тему и гипотезу.

Предполагается, что создание онтологии ПрО «Управление рисками ИБ» является очень важным шагом к построению эффективной интеллектуальной системы анализа и управления рисками ИБ. С помощью онтологии можно до мельчайших подробностей объяснить машине смысл понятия и указать его правильное использование. Онтология может содержать все знания ПрО, в том числе аксиомы и правила логического вывода. Как только мы сумели объяснить, по-настоящему онтологическим подходом, машине что такое «ресурс», «ценность ресурса», «угроза», «уязвимость», «вероятность успешной реализации угрозы», «коэффициент разрушительности», «негативное событие», «частота возникновения неблагоприятного события за фиксированный промежуток времени», «риск», «управление рисками» и т.д., то система на основе онтологии будет способна без колебаний обнаруживать негативные события, уязвимости, угрозы, определить риски, дать рекомендации и т.д. Конечно, все зависит от степени разработанности и качества этой самой онтологии. Тогда распознавания проблем ИБ будет основываться на знаниях и правилах логического вывода. Правила позволят извлечь даже сложные имплицитные (не явные) знания, что в свою очередь позволит не включить в онтологию выводимые знания, т.е. сокращать объём работы и сильно оптимизировать как саму онтологию, так и процедуру распознавания уязвимостей, угроз и

т.д. на её основе. Отсюда наша гипотеза о том, что полная и непротиворечивая онтология ПрО «Управление рисками ИБ» с соответствующими правилами логического вывода, в том числе с учетом прецедентов является необходимым и достаточным фундаментом эффективной интеллектуальной системы управления рисками. Отсюда цели и задачи данной работы.

Объектом данной работы является система управления рисками ИБ. **Предметом** работы является онтологическое обеспечение системы управления рисками ИБ. **Целью** является повышение качества и обоснованности принимаемых решений при управлении рисками ИБ за счет организации интеллектуальной ППР на основе онтологии ПрО «Управление рисками ИБ», включающей экспертные знания (стандарты, правовые нормы, концепты, процессы, отношения, требования и ограничения, решающие правила, формулы, алгоритмы, прецеденты и т.д.) и рассуждений по ним.

Для достижения указанной цели требуется решение следующих задач.

1) Составление онтологической модели ПрО «Управление рисками ИБ».
2) Составление онтологической модели процесса управлении рисками ИБ.
3) Определение места и роли управления рисками ИБ на основе ОДЭС ППР.
4) Управление рисками ИБ на основе ОДЭС ППР.
5) Структура подхода на основе ОДЭС ППР.
6) Оценка эффективности СЗИ на основе ОДЭС ППР.

Данная работа является важной составной частью большой исследовательской работы, целью которой является разработка, реализация и исследование современной интеллектуальной системы поддержки ПУР по ИБ с применением подхода онтологического инжиниринга, которая бы учитывала требования и ограничения нормативно правовой документации по ИБ, мнения экспертов по ИБ и связи процессов ИБ, и которая бы обеспечила повышение эффективности системы защиты информации (СЗИ) за счет сокращения временных затрат на ПУР.

2. Онтологическое моделирование предметной области «Управление рисками ИБ»

2.1 Онтология ПрО «Управление рисками ИБ»

2.1.1 Основные концепты ПрО

Из анализа нормативно-правовой базы [1]-[29] была выявлена совокупность базовых концептов ПрО. Ниже перечислены некоторые концепты этой совокупности.

Актив – любая ценность организации. Есть следующие виды активов: материальные активы (например, оборудование), финансовые активы, информационные активы (ИА), человеческие активы (люди и их квалификация, навыки и опыт), процессы, нематериальные активы (например, репутация, имидж). **Информационные активы (ИА)** – информационные ресурсы, в том числе различные виды информации, циркулирующие в информационной системе на всех этапах жизненного цикла (генерация, хранение, обработка, передача, уничтожение), или средства обработки информации.

Атака – попытка реализации угрозы (уничтожения, раскрытия, изменения, блокирования, кражи, получения несанкционированного доступа к активу или его несанкционированного использования. **Угроза ИБ** – совокупность условий и факторов, создающих потенциальную или реально существующую опасность нарушения ИБ. **Угроза** – потенциальная возможность использования уязвимости. **Интенсивность угрозы** – потенциальный ущерб, который может быть нанесен организации в случае реализации данной угрозы. **Уязвимость** – недостаток в

активе или в СЗИ, используя который, можно намеренно нарушить её целостность и вызвать неправильную работу. **Негативное событие (НС)** – событие, которое может породить угрозы ресурсам (активам), задействованным в основных процессах организации.

Основной процесс (ОП) – это совокупность взаимосвязанных мероприятий или задач, направленных на создание определённого продукта или услуги для потребителей. Процесс – агрегированный актив, который оперирует всеми другими активами для достижения бизнес целей организации. **Подпроцесс (ПП)** – процессы, работы, входящие в состав основного процесса и представляющие собой также процессы.

Риск ИБ – возможность того, что данная угроза сможет воспользоваться уязвимостью актива или группы активов и тем самым нанесет ущерб организации. **Информационный риск** – это опасность возникновения убытков или ущерба в результате применения компанией информационных технологий. **Приемлемый риск** – такой риск, с которым лицо, принимающее решения (ЛПР), в имеющейся ситуации может смириться. **Толерантный риск** – предельный уровень риска, который организация может выдержать без значительного ущерба для своей финансовой и конкурентной позиции.

Стратегия защиты от угроз - это общая, рассчитанная на перспективу руководящая установка при организации и обеспечении ИБ, направленная на то, достигались цели ИБ при наиболее рациональном расходовании имеющихся ресурсов.

Эффективность средства – оценка, которая отражает, с какой вероятностью применение данного средства локализует соответствующую ему базовую угрозу.

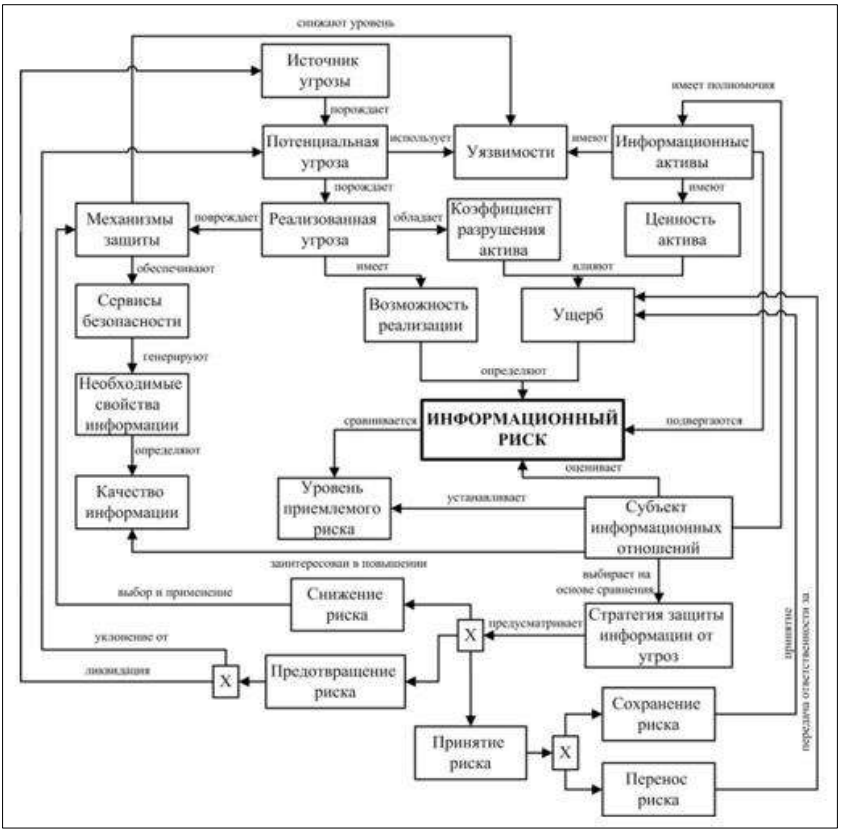


Рис. 1. Онтологическая модель ПрО «Управление рисками ИБ»
Fig. 1. Ontological model of subject area «IS Risk Management»

2.1.2 Онтологическая модель ПрО

На рис. 1 отображена онтологическая модель ПрО «Управление рисками ИБ».

Данная онтологическая модель отражает взаимосвязь основных концептов и компонент управления рисками комплексной системы ИБ.

2.1.3 Характеристики некоторых концептов онтологии ПрО

На рис. 2 и 3 представлены в виде ER-диаграмм свойства, ограничения и взаимоотношения некоторых концептов онтологии ПрО.

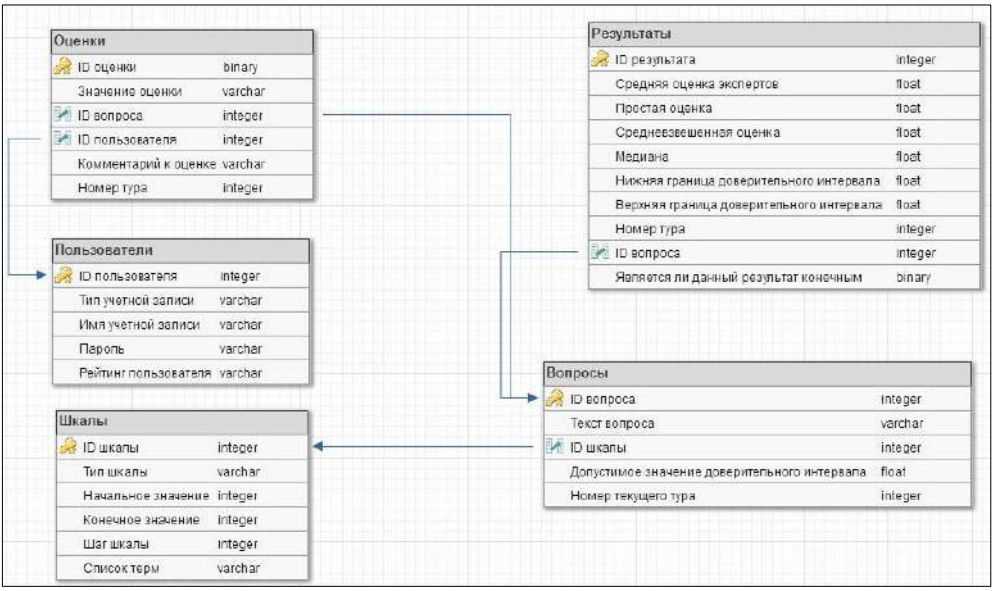


Рис. 2. Характеристики понятия: Вопрос, Шкала, Оценка, Пользователь, Результат
Fig. 2. Characteristics of the concept: Question, Scale, Assessment, User, Result

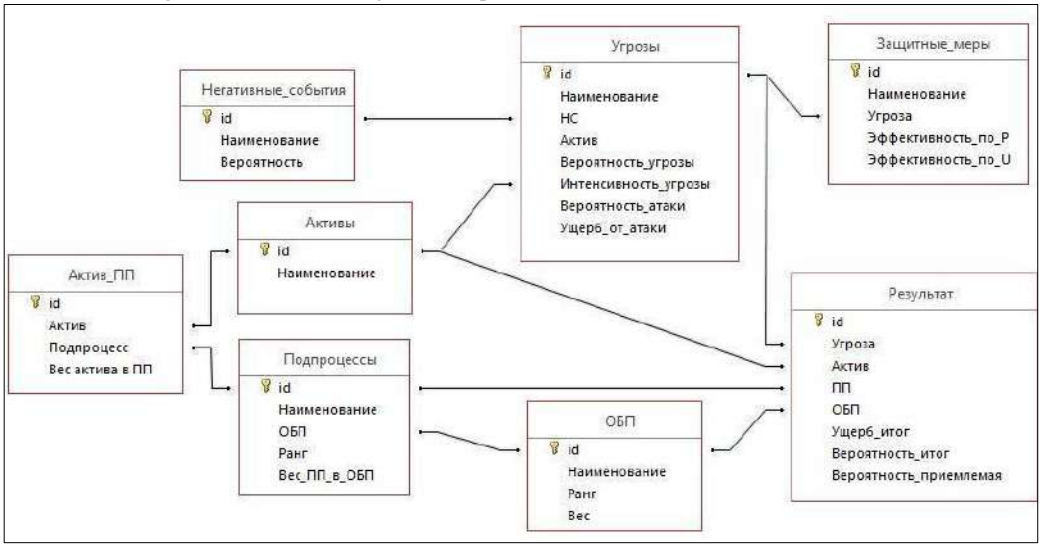


Рис. 3. Характеристики понятия: Актив, Угроза, Негативное событие, Защитная мера, Основной базовый процесс, Подпроцесс, Результат
Fig. 3. Characteristics of the concept: Asset, Threat, Negative event, Protective measure, Main basic process, Sub-process, Result

Характеристики концептов на диаграммах (рис. 2 и рис. 3) представлены в таком виде только для удобства чтения и понятности. Они на самом деле описывают экземпляры классов соответствующих концептов онтологии и являются частью ее.

2.1.4 Возможные вопросы (запросы) к онтологии ПрО

Запросы к онтологии осуществляются для извлечения явных и неявных знаний из неё. В данной работе онтология представлена спецификациями IDEF. В будущем для извлечения больше выгоды из нее, она будет реализована спецификациями и средствами семантического веба. При реализации онтологии спецификациями (RDF/RDFS, OWL, SKOS, FOAF, DC, vCard и т.д.) и средствами (protègè и его плагинами, графами знаний и т.д.) семантического веба, то запросы для извлечения явных знаний могут быть описаны с помощью языка запросов SPARQL, а для извлечения неявных (скрытых) знаний, с помощью правил логического вывода, можно применить языки SHACL, PIN, SWRL и подобные.

Ниже дана вербальная формулировка примеров возможных вопросов (запросов) к онтологии ПрО.

- Является ли данное событие негативным событием?
- Каковы негативные последствия данного негативного события?
- Какие угрозы вызывает данное событие?
- Какие уязвимости может эксплуатировать данная угроза?
- Какие уязвимости присутствуют в системе?
- Какие угрозы присутствуют в системе?
- Каково происхождение угрозы?
- Какие совместимые меры и средства защиты оптимально выбрать для данной угрозы?
- Какие совместимые меры и средства защиты оптимально выбрать для данного набора угроз?
- Совместимы ли данные мера и средство защиты?
- Решают ли данную угрозу данные мера и средство защиты?
- По каким параметрам оптимально выбирать совместимые мера и средство защиты для данной угрозы или их набора?
- Отвечают ли данные мера и средство защиты требованиям данного российского или международного стандарта?
- Отвечают ли данные мера и средство защиты требованиям данной НПД?
- Какова приемлемость данного риска?

2.2 Онтологическая модель процесса управления рисками ИБ

Онтология данного процесса базируется на двух главных понятиях: просчет рисков и обработка рисков. С этими понятиями связаны следующие важные понятия: угрозы ИБ; ИА (объекты воздействия); источники угроз; воздействия на объекты защиты.

При просчете рисков определяются: опасность угрозы; вероятность реализации угрозы; уязвимости реализации угрозы; способы реализации угрозы; возможные разрушительные воздействия; риски и их характеристики.

При обработке рисков должны быть определены корректирующие и предупреждающие воздействия.

3. Управление рисками на основе ОДЭС ППР

3.1 Место и роль управления рисками ИБ на основе ОДЭС ППР

В состав функций данного метода входят: онтологическая формализация требований и систематизация процессов обеспечения ИБ; онтологическая автоматизация совместимого выбора мер защиты и технических средств защиты; логический вывод (мониторинг) действия принятых мер и средств защиты; аналитическое обеспечение деятельности персонала по ИБ.

В результате реализации данного метода мы получим: онтологическую систематизацию требований по управлению ИБ, исходя из состава процессов обеспечения ИБ; связанное применение организационных мер и технических СЗИ; формированную аналитическую базу проведения исследований и анализа состояния ИБ; рационально обоснованное стратегическое, тактическое и оперативное управление процессами обеспечения ИБ; возможности своевременного применения корректирующих и предупреждающих воздействий на основе комплексного анализа состояния ИБ в целом, и объектов защиты в частности; возможности обеспечения планирования развития и поддержания СЗИ; онтологическую формализацию основных и второстепенных процессных составляющих управления и обеспечения ИБ; повышение общего уровня ИБ за счет формализации и систематизации ее основных процессов; снижение операционных рисков за счет уменьшения вероятности реализации угроз ИБ, ввиду повышения ее общего уровня; повышение прозрачности процессов ИБ в рамках СЗИ; повышение оперативности при решении задач обеспечения ИБ; снижение трудоемкости операций по обеспечению ИБ; повышение уровня компетенции персонала организации и специалистов по ЗИ в вопросах ИБ; повышение оперативности реагирования на инциденты ИБ; минимизацию затрат на эксплуатацию СЗИ.

3.2 Управление рисками ИБ на основе ОДЭС ППР

3.2.1 Трудность ПУР при управлении ИБ

Пусть у нас имеется организация, в которой выполняются $i = 1, \dots, n$ равноценных процессов управления ИБ и пусть r_i некоторое количество ресурсов, затрачиваемое на реализацию i-го процесса.

Пусть процесс i характеризуется параметрами: $t_i(r_i)$ – время на получение информации в ходе выполнение процесса с использованием ресурса r_i; (x_i) – количество получаемой информации в ходе выполнения процесса.

Пусть на организацию провели атаку, угрожающую ее непрерывную работу и руководству нужно срочно ПУР для нейтрализации атаки за время t_{max}, при неизменном общем количестве ресурсов R. Тогда: $R = \sum_{i=1}^{n} r_i = const$, и руководству необходимо получить такое количество информации f от всех процессов управления ИБ для перевода СЗИ в безопасное от атаки состояние на основе доступных ресурсов и за не более чем максимальное доступное время на принятие решения. Формально математически это значит:

$$\begin{cases} \sum_{i=1}^{n} f_i(x_i) \rightarrow \max\limits_{x=(x_1,\cdots,x_n)\in Z_+^n} \\ \sum_{i=1}^{n} t_i(r_i)x_i \le t_{max} \end{cases} \quad (1)$$

По теории управления, решение, которое необходимо принять, будет описываться параметрами: F – совокупный объем информации, необходимый для принятия решения; T – время, необходимое для принятия решения и обработки информации; R – состав ресурсов на

выполнение задачи; аналитические алгоритмы для обработки информации и принятия решения.

Тогда возможны следующие ситуации:

1) нехватка времени для получения необходимого количества информации, т.е. $\max \sum_{i=1}^{n} f_i(x_i) < F$;
2) возможная хватка времени для получения необходимого количества информации, т.е. $\max \sum_{i=1}^{n} f_i(x_i) \leq F$;
3) достаточно времени для получения необходимого количества информации, т.е. $\max \sum_{i=1}^{n} f_i(x_i) = F$;
4) более чем достаточно времени для получения необходимого количества информации, т.е. $\max \sum_{i=1}^{n} f_i(x_i) \geq F$.

Из условий задачи и алгоритма решения можно заключить, что для 1-го случая, СЗИ и механизмы управления ею будут недостаточны, т.е. набор применяемых процессов неэффективен и не может решить задачу полностью.

Для негативного сценария (нехватка времени) 2-го случая все как в 1-ом случае, но для его положительного сценария (минимальное время достаточно), полное решение можно получить. Кроме неэффективности реализации и управления СЗИ, также возможен вопрос о недостаточности ресурсов или о нехватке времени из-за применения неэффективных алгоритмов. В таком случае можно искать решение по алгоритму на рис. 4.

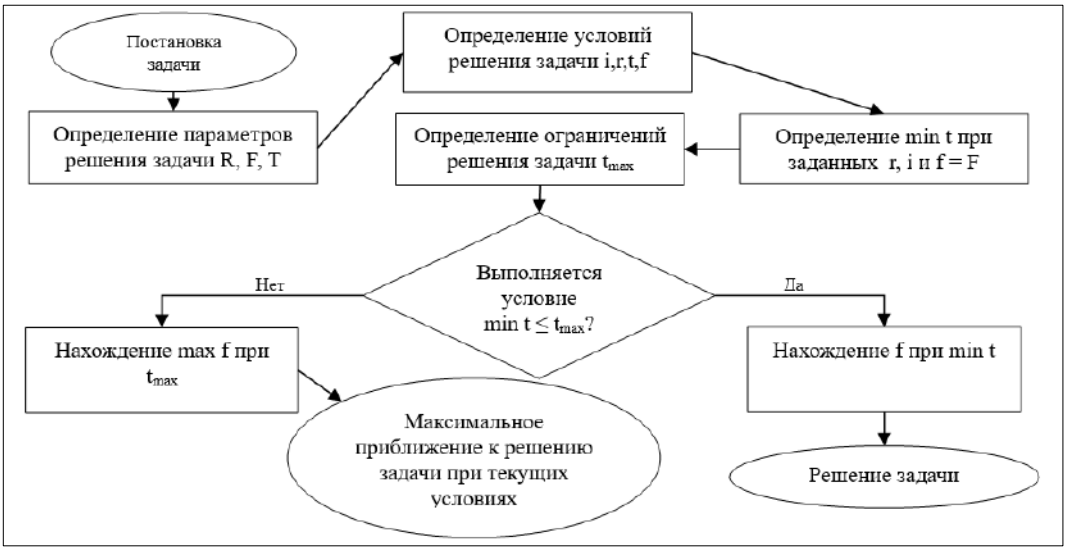


Рис. 4. Алгоритм решения задачи
Fig. 4. Algorithm for solving the problem

Как можно видеть из рис. 4, возможна ситуация нехватки времени для полного решения задачи. При этом также возможно, что нельзя оказать влияние на временные рамки, что больше затрудняет ситуацию.

Предполагается, что разрешение этой сложности возможно за счет введения в практику методику управления ИБ на базе обобщенного набора правил логического вывода ОДЭС ППР, позволяющих сильно сократить временные затраты на информационные процессы и процессы принятия должного и адекватного управленческого решения.

3.2.2 Процесс управления рисками ИБ на основе ОДЭС ППР

Общее описание процесса управления рисками

Управление рисками является одним из ключевых процессов в обеспечении ИБ. В данный момент есть множество методик управления рисками ИБ. Ключевым понятием всех методик является понятие «угроза ИБ».

Главное предназначение процесса управления рисками ИБ состоит в идентификации, анализе и контроле мер, учитывающих вероятности имплементации угроз ИБ, а по типу и составу защищаемого ИА, математически просчитать потенциальную величину потерь от этой имплементации. Эти потери зависят от стоимости актива (важности актива) и опасности имплементации угрозы. «Стоимость актива» или «Важность актива» может принимать следующие значения: «Высокая», «Средняя», «Низкая». Угроза ИБ, вместе с теми на рис. 3, описывается следующими основными параметрами: объект воздействия угрозы (актив\тип актива); источник возникновения угрозы; уязвимости, используемые в процессе имплементации угрозы; способ имплементации угрозы, посредствам использования присущей ей уязвимостей; деструктивное воздействие, возникающее в процессе имплементации угрозы; вероятность имплементации угрозы; опасность имплементации угрозы. Данные параметры взаимосвязаны (рис. 5.). С учетом параметров актива для оценки рисков ИБ получим рис. 6.

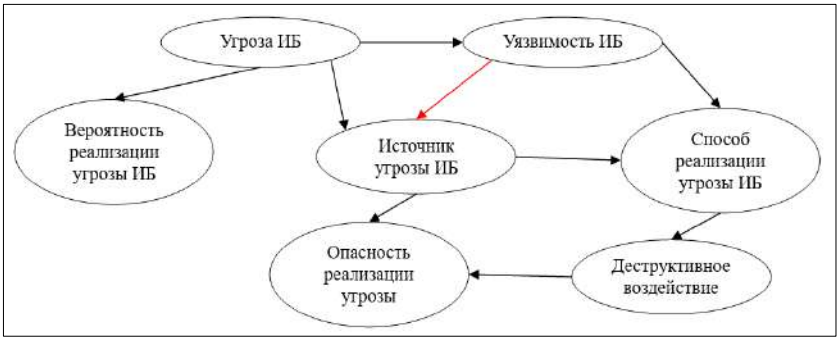


Рис. 5. Взаимосвязь параметров угроз ИБ
Fig. 5. Interrelation of IS threat parameters

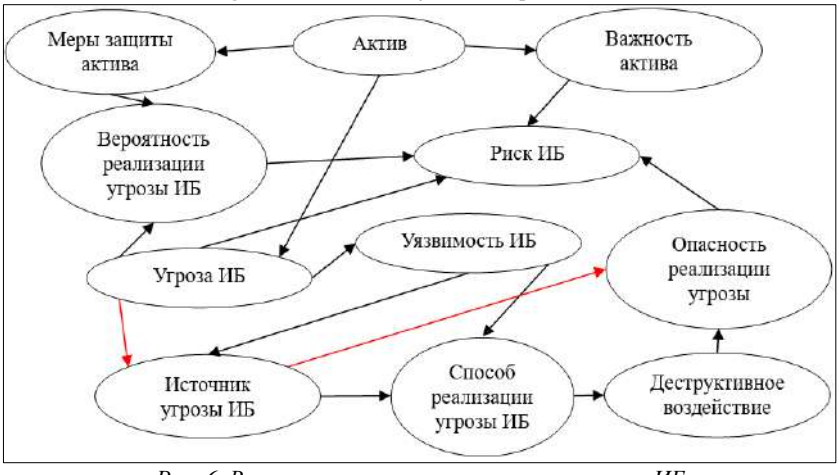


Рис. 6. Взаимосвязь параметров актива и угроз ИБ
Fig. 6. Relationship between asset parameters and IS threats

Исходя из рис. 6 видно, что для каждого актива можно получить дерево вида, представленного на рис. 7, позволяющего легко анализировать данный актив.

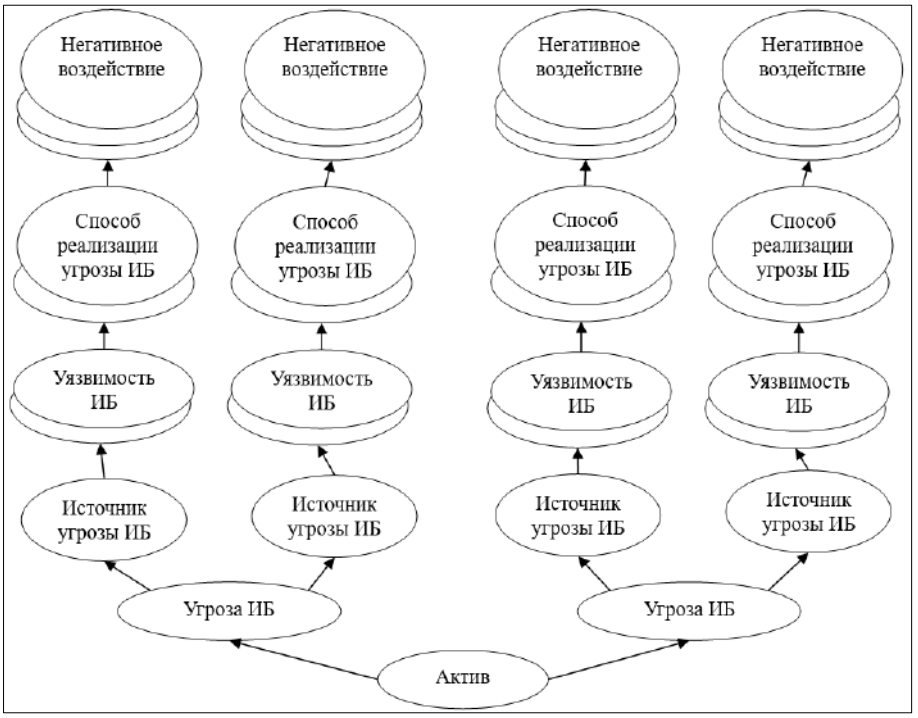


Рис. 7. Дерево угроз актива
Fig. 7. Asset threat tree

При управлении рисками ИБ важной задачей является контроль и учет контрмер (мероприятий по защите актива). По контрмерам и их параметрам можно получить вероятность имплементации угрозы, являющуюся ключевым при создании модели угроз и при расчете рисков. Расчет рисков выполняется на основе мнений всех экспертов, полученных методом экспертной оценки. Такой подход обладает следующими недостатками: субъективная оценка эксперта; разрозненность параметров, используемых экспертами; рутинность протекания процесса; длительность периода проведения оценки, вызванная необходимостью постоянной актуализации и переоценки информации, большим объемом анализируемой информации, большим количеством связанных параметров, используемых при оценке; сомнительность полученных результатов, с учетом человеческого фактора; трудоемкость процесса; отсутствие возможности дальнейшего использования и актуализации собранной информации.

Эти проблемы могут быть решены применением средств автоматизации на основе метода ОДЭС ППР. В данном методе используется онтология, полученная в результате онтологического инжиниринга (конструирования онтологической базы знаний) ПрО «Управление рисками ИБ».

Априорная вероятность реализации угроз ИБ по методу ОДЭС ППР

Для априорного определения вероятности реализации угроз (АОВРУ) ИБ, естественно применять в ОДЭС метод экспертной оценки (МЭО). МЭО позволяет учитывать специфические особенности конкретного объекта и динамику изменения мнений экспертов по ИБ. Это помогает повысить точность оценки рисков ИБ учетом всех свойств ИС. Также применяется принцип усреднения коэффициентов (ПУК) для уменьшения погрешностей МЭО. ПУК позволяет существенно снизить среднюю погрешность МЭО. Метод управления ИБ на основе ОДЭС ППР рассматривает каждую угрозу как абстрактный дискретный объект со своим набором взаимосвязанных параметров.

АОВРУ состоит в: определении проблемной области (вероятности возникновения угроз ИБ), для исключения комбинаторного взрыва; формировании группы экспертов; инженерии знаний экспертов – онтологический инжиниринг; формировании рабочей базы данных – заполнить онтологию; формализации принятия решений при получении результатов – рассуждении и логическом выводе.

С применением МЭО и ПУК нужно выполнить ещё и следующие пункты: Определение возможности осуществления способа реализации МЭО; Усреднение коэффициентов, полученных от экспертов.

Для начального наполнения онтологии знаниями, группа экспертов по ИБ предлагается для заполнения матрица из табл. 1.

Для применения АОВРУ необходимо сначала провести анализ всех угроз ИБ, найденных при просчете актуальности угроз с целью определения всех связанных с ними способов имплементации.

Табл. 1. Матрица оценки связи источника угроз и способа реализации угроз ИБ
Table 1. Matrix for assessing the connection between the source of threats and the method of implementation of IS threats

	Способ реализации	Способ реализации 1	Способ реализации 2	…	Способ реализации *M*
Источник угрозы					
Источник угрозы 1					
Источник угрозы 2					
…					
Источник угрозы *N*					

Ячейка матрицы заполняется экспертом следующим образом: невозможна (Н\В) – если, по мнению эксперта, способ реализации не используется источником угрозы; низкая (Н) – если, по мнению эксперта, способ реализации слабо используется источником угрозы; средняя (С) – если, по мнению эксперта, способ реализации нормально используется источником угрозы; высокая (В) – если, по мнению эксперта, способ реализации часто используется источником угрозы.

Данные качественные значения имеют следующие количественные оценки: Н\В = 0; Н = 0,2; С = 0,5; В = 1. Конечная матрица k_r возможности реализации угроз (ВРУ) после заполнения всеми экспертами имеет вид:

$$k_r = \sum_{i=1}^{n} \begin{bmatrix} 11 & \cdots & 1M \\ \vdots & \ddots & \vdots \\ N1 & \cdots & NM \end{bmatrix}_{k_i},$$

где i – число экспертов, N число источников угроз, а M – число способов реализации. При этом возможность реализации V_i угрозы ИБ i описывается набором всех способов реализации данной угрозы $E_i \in M$. Показатель ВРУ V_i нарушителем J рассчитывается как среднее арифметическое полной суммы показателей строки J матрицы k_r для всех E_i.

В зависимости от используемой методологии расчета актуальности угроз и рисков ИБ, коэффициенты подлежат дополнительной трактовке, однако исходя из общей логики метода разброс возможных значений итоговых показателей следует трактовать так: $V_i \in [0,\ 0{,}2)$ – нереализуема; $V_i \in [0{,}2\ ,\ 0{,}5)$ – мало вероятно реализуема; $V_i \in [0{,}5\ ,\ 0{,}8)$ – вероятно реализуема; $V_i \in [0{,}8\ ,\ 1)$ – крайне вероятно реализуема.

Степень опасности реализации угроз ИБ по методу ОДЭС ППР

Оценка степени опасности реализации угроз (СОРУ) ИБ основывается на принципах ОДЭС ППР. При этом принимаются во внимание величину и степень влияния возможных негативных последствий при реализации угрозы (ВНПРУ). В основном СОРУ формализуется в виде денежной суммы (см. ниже), однако это не применяется, если последствия реализации угроз ИБ неоценимы в денежном эквиваленте, или, если эта оценка неприемлема по степени важности самих активов. Тогда построение модели угроз имеет цель не минимизация финансовой потери, а определение наиболее уязвимых мест в СЗИ и формирование перечня контрмер.

Для оценки СОРУ ИБ будут применяться в ОДЭС МЭО и ПУК. Как и выше метод управления ИБ на основе ОДЭС ППР рассматривает каждую угрозу как абстрактный дискретный объект со своим набором взаимосвязанных параметров. При этом СОРУ будет меняться относительно важности актива.

Оценка СОРУ состоит в определении проблемной области (степень опасности угроз ИБ), для исключения комбинаторного взрыва; формировании группы экспертов; инженерии знаний экспертов – онтологический инжиниринг; формировании рабочей базы данных – заполнить онтологию; формализации принятия решений при получении результатов – рассуждении и логическом выводе.

С применением МЭО и ПУК нужно выполнить ещё и следующие пункты: определение СОРУ деструктивного воздействия на основе МЭО; усреднение коэффициентов, полученных от экспертов; определение важности защищаемых активов.

Для начального наполнения онтологии знаниями группа экспертов по ИБ предлагается для заполнения матрица из табл. 2.

Для оценки СОРУ необходимо сначала провести анализ всех угроз ИБ, найденных при просчете актуальности угроз с целью определения всех связанных с ними ВНПРУ.

Табл. 2. Матрица оценки связи источника угроз и ВНПРУ ИБ
Table 2. Matrix for assessing the connection between the source of threats and the impact of possible negative consequences in the implementation of the IS threat

	ВНПРУ	ВНПРУ 1	ВНПРУ 2	…	ВНПРУ М
Источник угрозы					
Источник угрозы 1					
Источник угрозы 2					
…					
Источник угрозы N					

Ячейка матрицы заполняется экспертом следующим образом: Низкая (Н) – если, по мнению эксперта, ВНПРУ имеет низкую степень влияния на защищаемые от источника угрозы активы; Средняя (С) – если, по мнению эксперта, ВНПРУ имеет среднюю степень влияния на защищаемые от источника угрозы активы; Высокая (В) – если, по мнению эксперта, ВНПРУ имеет высокую степень влияния на защищаемые от источника угрозы активы.

Данные качественные значения имеют следующие количественные оценки: Н = 0; С = 0,5; В=1. Конечная матрица k_r СОРУ после заполнения всеми экспертами имеет вид:

$$k_r = \sum_{i=1}^{n} \begin{bmatrix} 11 & \cdots & 1M \\ \vdots & \ddots & \vdots \\ N1 & \cdots & NM \end{bmatrix}_{k_i},$$

где i – число экспертов, N число источников угроз, а M – число ВНПРУ. При этом степень опасности D_i угрозы ИБ i описывается набором всех ВНПРУ $Q_i \in M$.

Показатель СОРУ D_i нарушителем J рассчитывается как среднее арифметическое полной суммы показателей строки J матрицы k_r для всех Q_i.

В зависимости от используемой методологии расчета актуальности угроз и рисков ИБ, коэффициенты подлежат дополнительной трактовке, однако исходя из общей логики метода разброс возможных значений итоговых показателей следует трактовать так: $D_i \in [0\ ,\ 0{,}4)$ – низкая степень опасности; $D_i \in [0{,}4\ ,\ 0{,}8)$ – средняя степень опасности; $D_i \in [0{,}8\ ,\ 1)$ – высокая степень опасности.

Формализация СОРУ в виде денежной суммы производится следующим образом:

Для каждого актива определим денежную стоимость с точки зрения его материальной (например, физической инфраструктуры) и нематериальной (например, репутации организации и цифровой информации) ценности для организации. Оценивая общую стоимость влияния для каждого актива, используем следующие категории: стоимость замены, затраты на обслуживание и поддержание работоспособности, затраты на обеспечение избыточности и доступности, репутация организации (репутация на рынке), эффективность работы организации, годовой доход, конкурентное преимущество, внутренняя эффективность эксплуатации, правовая и регулятивная ответственность.

Еще один подход к определению стоимости активов основан на сотрудничестве с группой управления финансовыми рисками, у которой должны быть страховые оценки и информация о страховом покрытии для соответствующих активов.

В результате получится перечень активов с указанием их приоритетов и приблизительной оценки их денежной стоимости для организации.

Теперь можно определить степень ущерба в процентах, который может быть причинен активу. Для этого используем уровень подверженности воздействию, определенный в ходе анализа и обсуждения собранных данных. Полученное значение называется фактором подверженности воздействию.

Следующий шаг состоит в получении количественной оценки влияния путем умножения стоимости актива C_j на фактор подверженности воздействию f_j. Тогда СОРУ D_i в деньгах определяется суммированием всех количественных оценок влияния на активы для данной угрозы, т.е.

$$D_i = \sum_{j=1}^{L} C_j * f_j$$

где D_i степень опасности угрозы ИБ i в деньгах, L - число активов, C_j – стоимость j-ого актива, f_j - фактор подверженности воздействию j-ого актива.

Постаприорная оценка возможности реализации угроз ИБ

На основе определении пост априорной оценки возможности реализации угроз (ПАОВРУ) ИБ положены структура ОДЭС ППР и МЭО. Базой для принятия решений при оценке ВРУ ИБ являются знания в статической и динамической частях онтологии. Динамическая часть онтологии содержит знания по ИС, включая сведения об ИА, средствах и механизмах ЗИ, а также свойства и метазнания по каждому ИА, по каждой СЗИ.

Статическая же часть онтологии содержит служебную информацию и правила взаимодействия, как в самой статической части, так и в динамической.

Статическая часть онтологии содержит: знания о каждой из имеющихся угроз ИБ, включая: объект воздействия (ИА), способы имплементации уязвимости, предусмотренные данной угрозы; условия существования данной угрозы для защищаемого ИА.

Правила логического вывода для ответа на вопросы (запросы) строятся на основе следующих привязок параметров друг к другу: «Тип ИА» – «Угроза»; «Угроза» – «Условие возникновения угрозы (УВУ)»; «Угроза» – «Меры защиты от угрозы (МЗОУ)»; «МЗОУ» – «Требования»; «МЗОУ» – «Ограничения»; «МЗОУ» – «Средства защиты от угрозы (СЗОУ)»;

«МЗОУ» – «Стандарты и НПД»; «Требования» – «Ограничения»; «СЗОУ» – «Стандарты и НПД» и т.д.

Поскольку УВУ прямо связано с типом ИА и его угроз, то оценка параметра «УВУ» будет меняться в значениях (α): 0 – УВУ отсутствует, 1 – УВУ существует.

УВУ определяется по общему мнению всех экспертов.

Опираясь на знаниях об УВУ и типах ИА воздействия угрозы, на этапе наполнения онтологии формализуются МЗОУ, для каждой из которых заполняются СЗОУ. Для защиты от угрозы набором МЗОУ, каждая МЗОУ будет иметь свой вес в виде процентной градации степени защиты от угрозы (X).

Для каждого СЗОУ конкретной МЗОУ определяется максимальное числовое значение, определяющее полноту перекрытия МЗОУ данным СЗОУ (R). На основе мнения экспертов по ИБ, R может принимать следующие значения: 0 – СЗОУ отсутствует; 3 – СЗОУ не обладает подтверждением реализации требуемых функций, корректности их настройки; 5 – СЗОУ обладает подтверждением требуемых функций, но отсутствуют сведения о корректности; 7 – СЗОУ обладает подтверждением требуемых функций и есть подтверждения корректности настройки.

Расчет ВРУ V производится по формуле:

$$\begin{cases} V = \alpha * \dfrac{\left(\left(R_{1_{max}} - R_{1_{par}}\right) * \dfrac{X_1}{100}\right) + \left(\left(R_{2_{max}} - R_{2_{par}}\right) * \dfrac{X_2}{100}\right) + \left(\left(R_{n_{max}} - R_{n_{par}}\right) * \dfrac{X_n}{100}\right)}{R_{max}} \\ X_1 + X_2 + \cdots + X_n = 100, \\ \alpha \in [0,1]. \end{cases}$$

$R_{i_{par}}$– текущее значение R полноты перекрытия СЗОУ i-ой МЗОУ.

X_i– коэффициент X степени защиты от угрозы v-ой МЗОУ.

α – наличие УВУ.

Достоинства подхода на базе ОДЭС ППР

Достоинства подхода на основе ОДЭС ППР состоят в: онтологической формализации ПрО «Управление рисками ИБ»; возможности применения современных спецификаций, технологий и средств онтологического инжиниринга, например, семантического веба и protégé, для управления рисками ИБ; исключении субъективной неопределенности экспертов по ИБ; объективности оценки ВРУ, с учетом общего мнения группы экспертов; формализации всех применяемых параметров оценки ВРУ ИБ; автоматизации и интеллектуализации выполнения процесса; учете требований и действующих средств и мер ЗИ; существенном сокращении времени проведения оценки; упрощении процессов проведения переоценки и актуализации; сокращении нагрузки на аналитиков по ИБ; установлении логической взаимосвязи параметров, применяемых при оценки ВРУ ИБ; статистической достоверности полученных результатов; минимизации человеческого фактора; возможности дальнейшего использования и актуализации собранной информации, достигаемой посредством применения средств автоматизации; Гибкость метода управления рисками ИБ, учитывающего не только положения Российской и международной нормативно-правовой документации по ИБ, но и внутренние и отраслевые требования, характерные для специфики деятельности конкретной организации; Снижение временных и материальных затрат на поддержание процессов обеспечения ИБ;

Основой предлагаемого метода является онтологический подход и принципы ДЭС ППР, а МЭО используется только для учета мнения экспертов и как один из многих путей заполнения онтологии данными наряду с автоматическими методами. Система базируется на детально разработанной онтологии ПрО «Управление рисками ИБ». Кроме МЭО существует расширение онтологической модели по принципу объединения онтологий разного уровня представления или обучения. Также существует расширение аналитических способностей

ОДЭС ППР добавлением новых правил логического вывода экспертным методом, методами DataMining и машинного обучения.

Наш онтологический подход позволит эффективно управлять знаниями для эффективного управления ИР.

Применение правил логического вывода обеспечивает получение детальной картины принятого решения и предоставляет возможность оценить степень влияния на систему взаимодействующих факторов.

3.3 Структура подхода на основе ОДЭС ППР

Структура типовой ОДЭС ППР в управлении рисками ИБ включает следующие компоненты.

- Онтологию ПрО «Управление рисками ИБ» с разработанными классификаторами от экспертной группы, поддержанной группой инженеров по знаниям. Онтология содержит все о ПрО, в том числе методики определения актуальности угроз, уязвимостей и рисков ИБ; методики оценки эффективности работы СЗИ и её составных частей; методики проведения аудита и оценки степени соответствия требованиям внешних и внутренних регулирующих органов; правила логического вывода. Она может быть создана в Protégé.
- Пользовательский интерфейс для пользователей и экспертов, который может быть создан средствами Java и Jena или на базе Protégé.
- Интеллектуальный редактор онтологии, который может быть создан средствами Java и Jena или на базе Protégé.
- Целевая аудитория пользователей системы.
- Экспертная группа, участвующая в процессе формирования требований исходных данных по самой системе и её составным частям.
- Группа инженеров по знаниям, осуществляющих онтологический инжиниринг под контролем экспертной группы.
- Рабочая память системы, обеспечивающая взаимодействие данных.
- Механизмы логического вывода, осуществляющего анализ исходных и актуализированных знаний на основе сформированных экспертами логических правил поведения системы.
- Подсистема объяснений, отвечающая за аналитическое обоснование принятых решений по результатам обработки знаний механизмами логического вывода.

3.4 Оценка эффективности СЗИ на основе ОДЭС ППР

Для оценки эффективности подхода необходимо сравнить эффективность СЗИ на его основе и СЗИ без его применения. Для этого требуется определить главные факторы, влияющие на эффективность СЗИ. Сравнение будем проводить математически, суммируя частные коэффициенты эффективности (суммируются взвешенные суммы частных факторов влияния на эффективность СЗИ), т.е. Коэффициент эффективности СЗИ рассчитывается по формуле:

$$W(S_i) = \sum_{f=1}^{N} \alpha_f s_f(S_i),$$

где: S_1 – исходный подход к управлению рисками ИБ в компании EDM SA Республики Мали;

S_2 – подход к управлению рисками ИБ на основе ОДЭС ППР;

$W(S_i)$ – коэффициент эффективности СЗИ подхода;

f – индекс фактора влияния;

N – количество факторов влияния;

αf – вес факторов $s_f(S_i)$ в общем весе эффективности СЗИ, при этом $\sum_{f=1}^{N} \alpha_f = 1,\ 0 \le \alpha_f \le 1$.

Факторы, имеющие высокое влияние на эффективность СЗИ предоставлены экспертами по ИБ малийской энергетической компании EDM SA после опроса. В результате получены следующие критерии для оценки эффективности: время на внедрение процесса управления рисками ИБ; время на сбор исходных данных от процесса управления рисками ИБ; время на обработку собранной от процесса управления рисками ИБ информации; время на актуализацию собранной от процесса управления рисками ИБ информации; время на анализ полученных результатов; время на ПУР.

Эксперты оценили каждый фактор, при этом высокий коэффициент получил тот подход СЗИ, в котором время на выполнение полного цикла по процессу управления рисками ИБ было минимальным. Оценки экспертов указаны в табл. 3. Уточняем, что данные значения были выставлены экспертами малийской энергетической компании EDM SA и авторы данной работы не отвечают за их корректность.

Табл. 3. Данные для расчета и анализа эффективности исходного подхода к ИБ и подхода к ИБ на основе ОДЭС ППР

Table 3. Data for calculating and analyzing the effectiveness of the initial approach to IS and an approach to IS based on decision support with ontological dynamic expert systems

	Факторы, влияющие на эффективность СЗИ	**Степень значимости коэффициента**	**Частный показатель исходного подхода управления ИБ**	**Частный показатель подхода ОДЭС ППР управления ИБ**
1	**Временные затраты на внедрение процесса ИБ**	0,25	1	0,1
2	**Временные затраты на сбор исходных данных от процесса ИБ**	0,10	0,5	0,9
3	**Временные затраты на обработку собранной от процесса ИБ информации**	0,15	0,3	0,8
4	**Временные затраты на актуализацию собранной от процесса ИБ информации**	0,15	0,4	1
5	**Временные затраты на принятие управленческого решения**	0,25	0,6	1
6	**Временные затраты на анализ полученных результатов**	0,10	0,3	0,8
	Итоговый коэффициент эффективности		**0,58**	**0,95**

Расчет коэффициента эффективности СЗИ с использованием исходного подхода управления ИБ $S1$:

$$W(S_1) = 0{,}25 * s_1(S_1) + 0{,}1 * s_2(S_1) + 0{,}15 * s_3(S_1) + 0{,}15 * s_4(S_1) + 0{,}25 * s_5(S_1) + 0{,}1 * s_6(S_1) \quad \leftrightarrow$$
$$W(S_1) = 0{,}25 * 1 + 0{,}1 * 0{,}5 + 0{,}15 * 0{,}3 + 0{,}15 * 0{,}4 + 0{,}25 * 0{,}6 + 0{,}1 * 0{,}3 \quad \leftrightarrow$$
$$W(S_1) = 0{,}58$$

Расчет коэффициента эффективности СЗИ с использованием подхода управления ИБ на основе ОДЭС ППР $S2$:

$$W(S_2) = 0{,}25 * s_1(S_2) + 0{,}1 * s_2(S_2) + 0{,}15 * s_3(S_2) + 0{,}15 * s_4(S_2) + 0{,}25 * s_5(S_2) + 0{,}1 * s_6(S_2) \quad \leftrightarrow$$
$$W(S_2) = 0{,}25 * 1 + 0{,}1 * 0{,}9 + 0{,}15 * 0{,}8 + 0{,}15 * 1 + 0{,}25 * 1 + 0{,}1 * 0{,}8 \quad \leftrightarrow$$
$$W(S_2) = 0{,}94$$

Из расчетов видно преимущество подхода управления ИБ на основе ОДЭС ППР. Этот подход сильно снижает временные затраты и увеличивает эффективность СЗИ на 36%.

4. Сравнительный анализ существующих подходов и методик к управлению рисками ИБ и ОДЭС ППР

Сравнение проводится на основе указания преимуществ и недостатков каждого подхода. В табл. 4 указаны описание, сильные и слабые стороны некоторых современных подходов к управления рисками ИБ, в том числе ОДЭС ППР.

Табл. 4. Анализ некоторых современных подходов к управлению рисками ИБ
Table 4. Analysis of some modern approaches to information security risk management

Подход/ Методика	**Описание**	**Сильные стороны**	**Слабые стороны**
Метод Дельфи [31], [32]	Формальный метод чисто экспертного прогнозирования; основывается только на экспертных оценках.	Позволяет оценить специфику каждой конкретной ситуации, способствует выработке независимого мышления членов экспертной группы и позволяет получить взвешенную оценку рассматриваемого вопроса.	Сложность в выборе большой группы компетентных экспертов, чрезмерную субъективность оценок.
Статистические методы (RiskMetrics Value-at-Risk VaR, Имитационное моделирование и т.п.) [30],[31]	Заключаются в определении вероятности возникновения потерь на основе статистических данных предшествующего периода и установлении области (зоны) риска, коэффициента риска и других параметров риска.	Позволяет проводить анализ и оценку различных вариантов развития событий и учитывать разные факторы рисков в рамках одного подхода с учетом накопленных статистических данных	Необходимость использования в данных методах вероятностных характеристик.
Методика RiskWatch [30]. США.	Обнаружение уязвимостей на основе анкетирования с использованием более чем 600 вопросов, разделенных на классы. В качестве показателей для оценки и управления рисками используются	Позволяет обнаруживать угрозы и уязвимости, оценивать уровни выделенных факторов,	Не учитывает организационные и административные уровни, не реализует все аспекты комплексной защиты информации,

	прогнозируемые среднегодовые потери (Annualized Loss of Expectancy, ALE) и оценка возврата от инвестиций (Return on Investment, ROI). Ожидаемая частота реализации угроз определяется в терминах среднегодовой оценочной частоты угрозы (Annual Frequency Estimate, AFE). База знаний RiskWatch определяет для каждой угрозы стандартную оценочную частоту (Standard Annual Frequency Estimate, SAFE).	оценивать соблюдение требований стандартов, прогнозировать величину ущерба и вырабатывать контрмеры с максимальным возвратом вложений. Содержит большую базу знаний, содержащую информацию по активам, угрозам, уязвимостям, видам ущерба, мерам защиты, а также опросные листы для оценки факторов риска. Для вычисления величины риска используется локальная оценочная частота угрозы (Local Annual Frequency Estimate, LAFE), определяемая пользователем самостоятельно на основании значения SAFE.	математическое ожидание ущерба не отражает с системных позиций концепты риска.
OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) [31]	Разработана в университете Карнеги-Мелон в 2007 году, описывает подход к качественной оценке рисков. Осуществляет оперативную оценку критических угроз, активов и уязвимостей.	Предусматривает регулярное проведение оценки рисков и обновление их величин как части процесса оценки рисков, простота и прозрачность, итеративный подход к анализу, невысокие трудозатраты.	Не дает количественной оценки рисков, отсутствие подробных баз знаний, отсутствие возможности оценки рисков в деньгах.
COBIT for Risk [32],[34]	Разработана ассоциацией ISACA (Information Systems Audit and Control Association) в 2013 году и базируется на лучших практиках управления	Доступ к общей библиотеке COBIT, апробированный метод, признается международными	Высокая сложность и трудоемкость сбора исходных данных, ресурсоемкость, отсутствие

	рисками (COSO ERM, ISO 31000, ISO\IEC 27xxx и др.).	институтами, наличие каталогов рисковых сценариев и «ИТ-контролей», может быть использован при аудите.	возможности оценки рисков в деньгах.
FRAP (Facilitated Risk Analysis Process) [32],[34]	Разработана компанией Peltier and Associates в 2000 году, описывает подход к качественной оценке рисков. Целью методики является выявление, оценка и документирование состава рисков информационной безопасности для заранее определенной области исследования. В качестве области исследования может быть выбрана информационная система, приложение, бизнес-процесс или другая часть инфраструктуры организации, нуждающаяся в оценке рисков информационной безопасности.	Простота и прозрачность, минимальные трудозатраты, минимальное количество участников проектной команды.	Отсутствие жестко регламентированного процесса управления рисками ИБ и детальных баз знаний, отсутствие возможности глубокой декомпозиции, подробной и точной оценки рисков, отсутствие возможности оценки рисков в деньгах.
Метод CRAMM (CCTA Risk Analysis and Management Method). [31],[32] Великобритания	Применяется как государственный стандарт, обладает большой библиотекой мер защиты, состоящей примерно из 3500 наименований, разделенной на 70 логических групп. Выбор мер защиты также может определяться в соответствии с требованиями стандартов BS7799:2005/ISO 27001.	Комплексный подход к оценке рисков, применение технологии оценки угроз и уязвимостей по косвенным факторам с возможностью верификации результатов, хорошо апробирован, хорошая система моделирования информационных технологий, большая БД для оценки рисков и выбора контрмер, использование в качестве средства аудита, универсальность и адаптируемость под профили	Необходимость высокой квалификации аудитора, плохая адаптивность к новым информационным системам, относительно трудоемкий, сложная отчетность, трудности при адаптации к потребностям конкретной компании в связи с недоступностью пользователям в ПО CRAMM модификации базы знаний и шаблонов отчетов, ПО CRAMM имеется только на английском языке, дорогая лицензия от 2000 до 5000 долларов.

		разных организаций.	
MSAT «Microsoft Security Assessment Tools» [33]	Предложена корпорацией Microsoft в 2006 году. Оценка риска состоит из определения профиля риска для бизнеса и оценки индекса эшелонированной защиты, которые сравниваются для измерения распределения риска во всех областях анализа.	Прозрачность, комбинирование качественного и количественного подходов, охват всех аспектов управления рисками, возможность оценки рисков ИБ в деньгах, наличие базы знаний.	Отсутствие типовых рисковых сценариев, высокая трудоемкость процесса управления рисками, плохо учитывает особенностей компаний на территории РФ, не работает по требованиям российского законодательства и стандартов, не измеряет эффективность применяемых защитных мер.
PTA (Practical Threat Analysis) [32]	Количественная оценка рисков. На входе поступают сведения об активах, угрозах, уязвимостях и контрмерах. На выходе получаем общую стоимость активов, стоимость контрмер, уровень рисков, перечень из 5 наиболее опасных угроз.	Можно использовать как средство непрерывного управления рисками. Можно отслеживать динамику состояния защищенности объекта изменяя входные данные (данные активов, угроз, уязвимостей и контрмер), а также на основе результатов оценки рисков, можно повысить эффективность средств защиты.	Результаты оценки рисков в процентах плохо интерпретируются. Также не всегда возможна оценка актива или ущерба, наносимого угрозой, в денежных единицах как это требуется.
CORAS [32],[34]	Суть состоит в адаптации, уточнении и комбинировании таких методов проведения анализа рисков, как Event-Tree-Analysis, цепи Маркова, HazOp и FMECA. Используется технология UML, базируется на австралийском/новозеландском стандарте AS/NZS 4360: 1999 Risk Management и ISO/IEC 17799-1: 2000 Code of Practice for	Очень простой подход, потребляет мало ресурсов, проводит классификацию рисков для оценки, снижения или устранения, ликвидация угроз.	Не предусмотрена периодичность проведения оценки рисков и обновление их величин, не позволяет оценить эффективность инвестиций для внедрения мер безопасности, не дает возможности найти

	Information Security Management. В этом стандарте учтены рекомендации, изложенные в документах ISO/IEC TR 13335-1: 2001 Guidelines for the Management of IT Security и IEC 61508: 2000 Functional Safety of Electrical/Electronic/Programmable SafetyRelated.		необходимый баланс между мерами, направленными на предотвращение, выявление, исправление или восстановление информационных активов.
AURUM (Automated Risk and Utility Management) [34]	Позволяет автоматизировать управление рисками ИБ, включая объективные меры по снижению рисков с учетом всей обстановки организации. Он основан на онтологии предметной области ИБ, чтобы гарантировать, что знания об ИБ предоставляются менеджеру рисков последовательным и всеобъемлющим образом.	Онтологический подход к анализу рисков.	Информация, представленная в онтологии, включает в себя только понятия угрозы, уязвимости и контроля, что очень ограниченно по сравнению с онтологическим подходом, предложенным в данной работе.
ОДЭС ППР. Россия	Описываемая в этой статье система	См. 2.1 и 2.2.2. (Достоинства подхода на базе ОДЭС ППР). И также ОДЭС ППР учитывает все преимущества и недостатки конкурентов.	Очень большая и детальная онтология, что может в дальнейшем усложнить запросы и логический вывод, при ненадлежащем контроле и не автоматическом управлении ею методами DataMining и машинного обучения.

Преимущества ОДЭС ППР над конкурентами говорят сами за себя.

5. Заключение

В работе: приведены основные концепты ПрО «Управление рисками ИБ»; разработана онтологическая модель ПрО; приведены основные свойства концептов онтологии ПрО; указаны примерные вопросы к онтологии для извлечения знаний; разработана онтологическая модель процесса управления рисками ИБ; указаны место и роль управления рисками ИБ на основе ОДЭС ППР; описана трудность принятия управленческих решений при управлении ИБ; описан процесс управления рисками ИБ на основе ОДЭС ППР и указаны его преимущества; дано описание структуры СЗИ на основе ОДЭС ППР; выполнена оценка эффективности СЗИ на основе ОДЭС ППР. проведен сравнительный анализ существующих подходов и методик к управлению рисками ИБ и ОДЭС ППР.

На основе разработанной онтологии и подхода могут быть созданы высокоинтеллектуальные системы управления рисками ИБ и системой защиты информации в целом. Подобная система сейчас разрабатывается и будет полностью реализована и протестирована.

Список литературы / References

[1] ГОСТ Р ИСО/МЭК 27000-2021. Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Общий обзор и терминология / ISO/IEC 27000-2021. Information technology. Security techniques. Information security management systems. Overview and vocabulary.

[2] ГОСТ Р ИСО/МЭК 27001-2006. Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Требования. / ISO/IEC 27001:2005. Information technology. Security techniques. Information security management. Requirements.

[3] ГОСТ Р ИСО/МЭК 27002-2021. Информационная технология. Методы и средства обеспечения безопасности. Свод норм и правил менеджмента информационной безопасности. / ISO/IEC 27000-2013. Information technology. Security techniques. Code of practice for information security controls.

[4] ГОСТ Р ИСО/МЭК 27003-2021. Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Руководство по реализации системы менеджмента информационной безопасности. / ISO/IEC 27003:2017. Information technology. Security techniques. Information security management systems. Guidance for implementation.

[5] ГОСТ Р ИСО/МЭК 27004-2021. Информационная технология. Методы и средства обеспечения безопасности. Менеджмент информационной безопасности. Измерения. / ISO/IEC 27004:2016. Information technology. Security techniques. Information security management. Monitoring, measurement, analysis and evaluation.

[6] ГОСТ Р ИСО/МЭК 27005-2010. Информационная технология. Методы и средства обеспечения безопасности. Менеджмент риска информационной безопасности. / ISO/IEC 27005:2008. Information technology. Security techniques. Information security risk management.

[7] ГОСТ Р ИСО/МЭК 27006-2020. Информационная технология. Методы и средства обеспечения безопасности. Требования к органам, осуществляющим аудит и сертификацию систем менеджмента информационной безопасности. / ISO/IEC 27006:2015. Information technology. Security techniques. Requirements for bodies providing audit and certification of information security management systems.

[8] ГОСТ Р ИСО/МЭК 27007-2014. Информационная технология. Методы и средства обеспечения информационной безопасности. Руководство по аудиту системы менеджмента информационной безопасности. / ISO/IEC 27007:2011. Information technology. Security techniques. Guidelines for information security management systems auditing

[9] ГОСТ Р ИСО/МЭК 11799-2005. Информационная технология. Практические правила управления информационной безопасностью. / ISO/IEC 11799-2000. Information technology. Code of practice for information security management.

[10] ГОСТ Р ИСО/МЭК 13335-1-2006. Информационная технология. Методы и средства обеспечения безопасности. Часть 1. Концепция и модели менеджмента безопасности информационных и телекоммуникационных технологий. / ISO/IEC 13335-1-2004. Information technology. Security techniques. Part 1. Concepts and models for information and communications technology security management.

[11] Управление рисками в соответствии с международным стандартом ISO 31000 / Risk management in accordance with the international standard ISO 31000. Available at http://diag.by/iso-31000, accessed 27.05.2020 (in Russian).

[12] ISO 31000. Available at https://en.wikipedia.org/wiki/ISO_31000, accessed 27.05.2020.

[13] Сидоренко А.И. Новый ISO 31000:2018. / Sidorenko A.I. New ISO 31000: 2018. Available at https://riskacademyrus.wordpress.com/2018/03/10/%D0%BD%D0%BE%D0%B2%D1%8B%D0%B9-iso-310002018/, accessed 27.05.2020 (in Russian).

[14] Федеральный закон от 27.07.06 г. № 149 – ФЗ «Об информации, информационных технологиях и защите информации» / Federal Law of 27.07.06, № 149 – FZ «On Information, Information Technologies and Information Protection» (in Russian).

[15] Закон Российской Федерации «О государственной тайне» от 21.07.93 № 5485-1 (с изменениями и дополнениями) / Law of the Russian Federation "On state secrets" dated 21.07.93 № 5485-1 (with amendments and additions) (in Russian).

[16] «Положение о государственной системе защиты информации в Российской Федерации от иностранных технических разведок и от ее утечки по техническим каналам» от 15 сентября 1993 г. № 912-51 / «Regulations on the state system of information protection in the Russian Federation from foreign technical intelligence services and from its leakage through technical channels» dated September 15, 1993 № 912-51 (in Russian).

[17] Федеральный закон от 27.07.06 г. № 152-ФЗ «О персональных данных» / Federal Law of 27.07.06, № 152-FZ «On Personal Data» (in Russian).

[18] Указ Президента Российской Федерации от 06.03.97 № 188 «Об утверждении Перечня сведений конфиденциального характера» / Decree of the President of the Russian Federation dated 06.03.97 № 188 «On approval of the List of confidential information» (in Russian).

[19] Федеральный закон от 28.08.2004 г. № 98-ФЗ «О коммерческой тайне» / Federal Law of 28.08.2004, № 98-FZ «On Commercial Secrets» (in Russian).

[20] Постановление Правительства РФ от 01.11.2012 № 1119 «Об утверждении требований к защите персональных данных при их обработке в информационных системах персональных данных» / Decree of the Government of the Russian Federation of 01.11.2012 № 1119 «On approval of requirements for the protection of personal data during their processing in personal data information systems» (in Russian).

[21] Постановление Правительства Российской Федерации от 21 марта 2012 г. № 211 г. «Об утверждении перечня мер, направленных на обеспечение выполнения обязанностей, предусмотренных Федеральным законом «О персональных данных» и принятыми в соответствии с ним нормативными правовыми актами, операторами, являющимися государственными или муниципальными органами» / Decree of the Government of the Russian Federation of March 21, 2012 № 211 «On approval of the list of measures aimed at ensuring the fulfillment of obligations stipulated by the Federal Law «On Personal Data» and regulatory legal acts adopted in accordance with it, operators who are state or municipal authorities» (in Russian).

[22] Приказ ФСТЭК России от 11 февраля 2013 г. № 17 «Об утверждении требований о защите информации, не составляющей государственную тайну, содержащейся в государственных информационных системах» / Order of the FSTEC of Russia dated February 11, 2013 № 17 «On approval of requirements for the protection of information that does not constitute a state secret contained in state information systems» (in Russian).

[23] Приказ ФСТЭК России от 18 февраля 2013 г. № 21 «Об утверждении состава и содержания организационных и технических мер по обеспечению безопасности персональных данных при их обработке в информационных системах персональных данных» / Order of the FSTEC of Russia dated February 18, 2013 № 21 «On approval of the composition and content of organizational and technical measures to ensure the security of personal data during their processing in personal data information systems» (in Russian).

[24] Конституция Российской Федерации / Constitution of the Russian Federation (in Russian).

[25] Стратегия национальной безопасности Российской Федерации до 2020 года, утвержденная указом Президента Российской Федерации от 12.09.2012 г. № 537 / The National Security Strategy of the Russian Federation until 2020, approved by the decree of the President of the Russian Federation of 12.09.2012 № 537 (in Russian).

[26] Доктрина информационной безопасности Российской Федерации, утверждена Указом Президента Российской Федерации от 5 декабря 2016 г. № 646 / The Doctrine of Information Security of the Russian Federation, approved by the Decree of the President of the Russian Federation of December 5, 2016 № 646 (in Russian).

[27] ГОСТ 34.003-90 Информационная технология. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения / GOST 34.003-90 Information technology. Set of standards for automated systems. Automated systems. Terms and Definitions (in Russian).

[28] ГОСТ 34.601-90 Информационная технология (ИТ). Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Стадии создания / GOST 34.601-90 Information technology. Set of standards for automated systems. Automated systems. Stages of development (in Russian).

[29] ГОСТ Р 51624-2000 Автоматизированные информационные системы в защищенном исполнении / GOST R 51624-2000 Automated information systems in a secure design (in Russian).

[30] RiskWatch International. Available at http://www.riskwatch.com, accessed 08.11.2020.

[31] Разумников С.В. Анализ возможности применения методов OCTAVE, RISKWATCH, CRAMM для оценки рисков ИТ для облачных сервисов. Современные проблемы науки и образования, no. 1, 2014 г. / Razumnikov S.V. Analysis application methods OCTAVE, RISKWATCH, CRAMM for risk assessment for it cloud services. Modern problems of science and education, пою 1, 2014 (in Russian).

[32] Куканова Н. Современные методы и средства анализа и управления рисками информационных систем компаний / Kukanova N. Modern methods and tools for analysis and risk management of information systems of companies. Available at http://citforum.ru/products/dsec/cramm/, accessed 08.11.2020 (in Russian).

[33] Средство оценки безопасности Microsoft Security Assessment Tool / Microsoft Security Assessment Tool. Available at https://technet.microsoft.com/ru-ru/security/cc185712.aspx, accessed 08.11.2020.

[34] Ekelhart A., Fenz S., Neubauer T. AURUM: A Framework for Information Security Risk Management. In Proc. of the 42nd Hawaii International Conference on System Sciences, 2009, pp. 1-10.

Информация об авторах / Information about authors

Ибрагим БУБАКАР – магистрант. Темы научных интересов: онтологическое обеспечение информационной безопасности, управление информационной безопасностью, криптографические методы защиты информации.

Ibrahim BOUBACAR – Master's Student. Research interests: ontological support of information security, information security management, cryptographic methods of information security.

Марина Борисовна БУДЬКО – кандидат технических наук, доцент, старший научный сотрудник. Научные интересы: моделирование и анализ процессов информационного взаимодействия вычислительных систем.

Marina Borisovna BUDKO – PhD, associate professor, senior researcher. Research interests: modeling and analysis of information interaction processes of computer systems.

Михаил Юрьевич БУДЬКО – кандидат технических наук, доцент. Научные интересы: информационная и функциональная безопасность телекоммуникационных сетей.

Mikhail Yurievich BUDKO – PhD, associate professor. Research interests: information and functional security of telecommunication networks.

Алексей Валерьевич ГИРИК – кандидат технических наук, доцент. Научные интересы: системное и функциональное программирование средств обеспечения информационной безопасности.

Alexei Valerievich GUIRIK – PhD, associate professor. Research interests: system and functional programming of information security tools.

DOI: 10.15514/ISPRAS-2021-33(5)-4

CC BY

Методы маркирования текстовых документов при печати посредством вертикального сдвига и изменения яркости фрагментов слов

[1] *Д.О. Обыденков, ORCID: 0000-0002-9296-6333 <obydenkov@ispras.ru>*
[1] *А.Е. Фролов, ORCID: 0000-0001-7616-2354 <aefrolov@ispras.ru>*
[1] *Ю.В. Маркин, ORCID: 0000-0003-1145-5118 <ustas@ispras.ru>*
[1] *С.А. Фомин, ORCID: 0000-0002-1151-2189 <fomin@ispras.ru>*
[2] *Б.В. Кондратьев, ORCID: 0000-0001-6348-117X <gae@mil.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. Солженицына, д. 25*
[2] *Министерство обороны Российской Федерации, 119160, г. Москва, ул. Знаменка, д. 19.*

Аннотация. В статье представлены результаты разработки методов маркирования текстовых документов, представленных как растровое изображение. Важной особенностью алгоритмов является возможность обратного преобразования документа, что позволяет заменять метку на маркированном документе. Разработка относится к структурным алгоритмам маркирования на основе вертикальных сдвигов слов и изменения яркости отдельных фрагментов слов. В работе используются инструменты сегментирования для получения разметки текста в документе, БЧХ-коды для коррекции ошибок, метод максимизации правдоподобия для извлечения метки, нейронная сеть для восстановления искаженных слов. Тестирование показало практическую применимость разработанных алгоритмов маркирования при печати и сканировании текстовых документов.

Ключевые слова: маркирование документов; стеганография; обработка изображений; коррекция ошибок; алгоритмы машинного обучения; защита от утечек информации; слепое извлечение цифрового водяного знака

Для цитирования: Обыденков Д.О., Фролов А.Е., Маркин Ю.В., Фомин С.А., Кондратьев Б.В. Методы маркирования текстовых документов при печати посредством вертикального сдвига и изменения яркости фрагментов слов. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 65-82. DOI: 10.15514/ISPRAS–2021–33(5)–4

Printed text documents watermarking based on vertical word shift and word fragments brightness changing

[1] *D.O. Obydenkov, ORCID: 0000-0002-9296-6333 <obydenkov@ispras.ru>*
[1] *A.E. Frolov, ORCID: 0000-0001-7616-2354 <aefrolov@ispras.ru>*
[1] *Y.V. Markin, ORCID: 0000-0003-1145-5118 <ustas@ispras.ru>*
[1] *S.A. Fomin, ORCID: 0000-0002-1151-2189 <fomin@ispras.ru>*
[2] *B.V. Kondrat'ev, ORCID: 0000-0001-6348-117X <gae@mil.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *Ministry of Defence of the Russian Federation, 19, Znamenka Str., Moscow, 119160, Russia*

Abstract. This paper describes the results of the development of methods for marking text documents represented as a raster image. An important feature of the algorithms is the possibility wipe current document mark and embed another one. The study refers to structural marking algorithms based on vertical word shifts and brightness changes of certain areas of the words. Segmentation tools are used to obtain document layout, BCH codes for error correction, a likelihood maximization method for label extraction, and a neural network for perturbed words recovery. Testing has proved the practical applicability of the algorithms with printing and scanning.

Keywords: data leakage prevention; documents watermarking; print-scan robust watermarking; blind watermarking method; stenography; image processing; error detection and correction; neural networks

For citation: Obydenkov D. O., Frolov A.E., Markin Y.V., Fomin S.A., Kondrat'ev B.V. Printed text documents watermarking based on vertical word shift and word fragments brightness changing**.** Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 65-82 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–4

1. Введение

Многие частные компании и государственные учреждения несут финансовые и репутационные потери в связи с утечками конфиденциальных данных. Утечки можно разделить на две категории:

- *умышленные* – преднамеренные утечки, реализуемые с целью получения личной выгоды, промышленного шпионажа и прочего;
- *случайные* – прямо или косвенно произошедшие по неосторожности или неосведомленности сотрудников.

Согласно статистике [1] около половины утечек относится к первой категории. Для борьбы с неправомерным распространением конфиденциальной информации существуют различные программные системы, которые также можно разделить на два типа:

- *Системы предотвращения утечек* – системы, работающие превентивно и призванные не допустить утечку информации. Такие системы анализируют потоки данных, пересекающих границы внутренних сетей организации, потенциально способные содержать конфиденциальную информацию. Программные системы отслеживают различную «подозрительную активность» с использованием различных техник: анализ сетевого трафика, контроль чтения и записи информации на компьютерах сотрудников, отслеживание изменений сетевых настроек и т. д.;
- *Системы расследование инцидентов утечек* – системы, предназначенные для локализации и выяснения причин уже произошедшей утечки данных. Чаще всего подобные системы специальным образом помечают распространяемые данные, что позволяет по полученным в ходе расследования образцам определить источник утечки. Также схожие принципы могут применяться в системах защиты авторских прав.

Большинство систем предотвращения утечек данных, представленных на рынке, нацелены на работу исключительно с цифровыми каналами утечки информации: сеть, электронная почта, мессенджеры и прочие. Действительно, по ряду причин большинство утечек осуществляется именно по этим каналам. Однако методы противодействия утечкам информации на бумажных носителях по-прежнему актуальны – согласно отчету компании InfoWatch за 2020 год до 10% утечек приходится именно на этот канал [1]. Более того, в некоторых организациях доля утечек информации на бумажных носителях может быть выше в силу специфики рабочих процессов, как например, изолированная внутренняя сеть или запрет на использование личных мобильных смартфонов.

Текущая работа выполнялась в ходе проекта по созданию системы комплексной защиты от утечек информации. В данной статье рассматриваются методы ***внедрения*** метки в изображение выводимого на печать документа. *Меткой* в данном контексте является последовательность бит определенной длины, с помощью которой можно идентифицировать компьютер, с которого документ был отправлен на печать. Встраивание метки происходит путем малозаметного изменения внешнего вида текста документа. При расследовании инцидентов утечек выполняется ***извлечение*** метки из изображения маркированного (или *помеченного*) документа, полученного путем сканирования или фотографирования бумажного документа. При использовании бумажных носителей в системах документооборота зачастую возникает необходимость повторной печати отсканированных документов, потенциально уже маркированных, поэтому требуется обеспечить возможность замены метки на документах с уже встроенной меткой. Процесс замены метки можно разбить на две операции: ***стирание*** старой метки и внедрение новой метки. На Рис. 1 представлена схема маркирования документов, интегрированная в подсистему печати ОС [2], цветом выделены реализуемые алгоритмами маркирования операции.

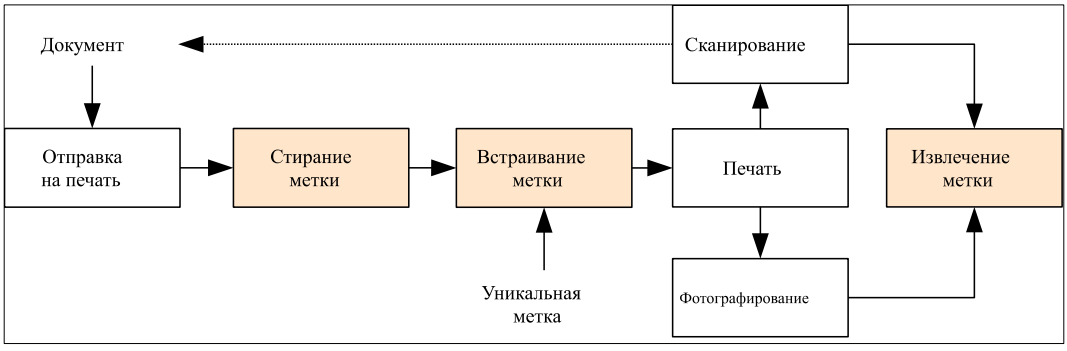


Рис. 1. Схема маркирования документов
Fig. 1. Document marking scheme

2. Связанные работы

Задача встраивания цифровых меток в изображения достаточно хорошо изучена и описана в ряде публикаций. Большинство авторов выделяют методы встраивания цифровых меток в область преобразования (*transform domain*) или в пространственная область (*spatial domain*).

К первой группе относятся методы встраивания цифровых меток в изображения, полученные путем некоторого преобразования – в частности, посредством дискретного косинусного преобразования (DCT) [3], дискретного преобразования Фурье (DFT) [4], дискретного вейвлет-преобразования (DWT). В данной статье эти методы рассматриваться не будут, поскольку их применение к изображениям текстовых документов (визуально близких к бинарным изображениям в отличие от цветных графических изображений) вносит существенные искажения.

Вторая группа включает методы маркирования, основанные на модификации оригинального изображения – в частности, внедрение некоторого периодического шаблона для графических изображений [5], сдвиг слов, изменение межстрочного интервала, добавление ошибок (орфографических, пунктуационных) для текстовых документов. Дальнейшая классификация методов маркирования пространственного домена выделяет лингвистические и структурные методы [6][7]:

- **Лингвистические методы** — изменяют семантические или синтаксические свойства текстового содержимого документа;
- **Структурные методы** — изменяют параметры визуального представления текстового слоя документа, но не меняют смысл текста.

2.1. Лингвистические методы

Лингвистические методы маркирования документов подразделяются на категории:

- *Семантические* — включают в себя различные методы сокрытия информации в тексте посредством изменения таких атрибутов текста как правописание, использованием аббревиатур и акронимов, замена на синонимы и другие.
- *Синтаксические* — сокрытие информации без значительного изменения смысла текста. В значительной степени такие методы маркирования полагаются на синтаксические свойства языка или использует присущие ему особенности. Например, замена определенных букв некоторого алфавита на визуально схожие буквы другого алфавита.

2.2. Структурные методы

Структурные методы маркирования документов подразделяются на следующие категории.

- *Непечатаемые символы*

  Различные непечатаемые символы широко используются при маркировании документов. Существуют работы, где такие символы используются как для замены, так и для вставки содержимого в исходном тексте. Пробельные символы хорошо подходят для встраивания информации в текстовый слой, поскольку крайне сложно визуально отличить различные вариации пробела (в Unicode-кодировке имеется почти два десятка пробельных символов). Встречаются различные вариации алгоритмов, когда информация встраивается между словами, предложениями, абзацами, пустыми строками и иные [8]-[10].

  Также существуют алгоритмы, использующие для кодирования символы нулевой ширины, например, пробел нулевой длины или символы, выполняющие сервисные функции в лигатурных шрифтах. Всего в Unicode существует 10 непечатаемых символов. Кодирующие метку символы могут размещаться на самых различных позициях, например, после пунктуационных символов, между пустыми линиями и параграфами [11][12].

- *Смещение текстовых элементов*

  Одна из первых работ по маркированию документов на основе вертикального смещения текстовых линий появилась еще в 1994 году [13]. Смещение текстовой линии вверх или вниз кодировало единичный бит [15]. Позднее появились работы, в которых для кодирования применялось горизонтальное смещение слов [14]: слова объединяются в группы по три и слово, стоящее посередине, смещается вправо или влево для кодирования одного бита метки [16][17].

- *Свойства форматирования текста*

Большим разнообразием обладает группа методов, использующих для кодирования различные свойства форматирования текста: размер, цвет, особенности начертания шрифта и другие свойства. В частности, разработано множество методов кодирования метки на основе искажения начертания глифов. В ранних работах вносимые искажения довольно грубые и визуально заметны. В последующих работах изменения менее заметны, поскольку модифицируются лишь контуры отдельных глифов [18]. Метод на основе искажения шрифта использует нейронную сеть для считывания вариации конкретной буквы и визуально различим только при близком рассмотрении и прямом сравнении с оригиналом [19].

Также существует группа методов, специализирующаяся на кодировании с использованием особенностей арабского языка. Метка кодируется посредством сдвигов точек в определенных словах, используемых при начертании букв арабского алфавита, а также удлинения вертикальных черт в других буквах, что не меняет смысла написанного [20].

2.3. Существующие решения

В настоящий момент на российском рынке существует несколько продуктов, которые реализуют возможность внедрения уникальных меток в выводимые на печать документы: *TraceDoc* [21] и *EveryTag* [22]. Информация об используемых методах маркирования представляет коммерческую тайну и отсутствует в открытом доступе, однако описание продуктов на сайтах производителя позволяет сделать вывод, что для извлечения метки из маркированного документа системам требуется оригинальная, немаркированная версия документа. Такой подход имеет свои преимущества – возможность сравнивать маркированный документ с оригиналом потенциально повышает точность извлечения, а также снижает заметность метки на документе. В то же время необходимость содержания и поддержки базы данных оригиналов и всех уникальных копий документов является значительным недостатком данного подхода, так как такая база становится единой точкой отказа, а также централизованный характер такой базы может быть несовместим со структурой некоторых организаций.

2.4. Резюме

Среди рассмотренных методов маркирования был выбран структурный подход как наиболее подходящий для решения поставленной задачи – маркирование текстовых документов. Лингвистические методы не подходят, поскольку изменяют содержимое документа, что в имеющейся постановке недопустимо. Методы встраивания цифровых меток в область преобразования также не подходят, так как существенно ухудшают визуальное представление документа. Подходы к маркированию на основе изменения свойств отдельных глифов чрезвычайно требовательны к качеству изображения при извлечении, что сильно ограничивает их возможное практическое применение. Методы встраивания цифровых меток, использующие особенности определенных форматов, ограничивают применение сравнительно небольшим числом форматов документов. Также при встраивании используется структура формата, предоставляющая информацию о текстовом слое, форматировании и других атрибутах документа, тогда как при извлечении или повторном встраивании сканированный или фотографированный документ представлен как растровое изображение.

Разработка метода кодирования требует поиска компромисса между противоречивыми критериями:

- **Незаметность** – стойкость метода к обнаружению метки в маркированном документе;

- **Надежность** – стойкость метода к различным искажениям, возникающим при печати/сканировании/фотографировании маркированного документа;
- **Емкость** – количество бит, которые можно встроить в маркируемый документ.

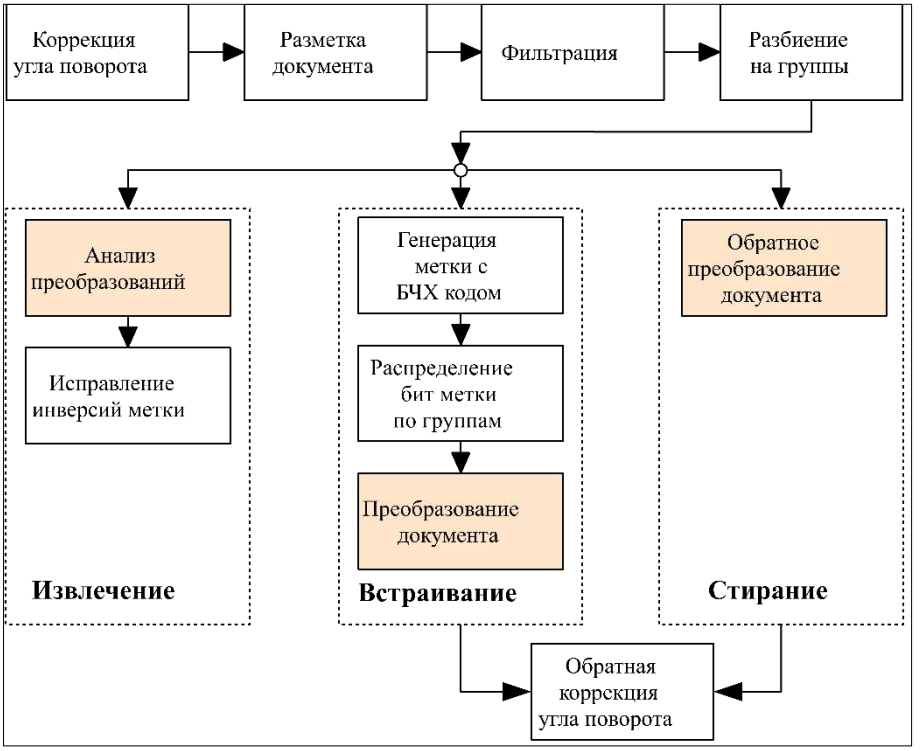


Рис. 2. Этапы маркирования документа для различных операций (цветом выделены этапы, реализация которых зависит от алгоритма)
Fig. 2. Document marking process pipeline for different operations (algorithm specific stages colored)

3. Описание алгоритмов маркирования

В рамках работы над проектом было разработано и реализовано два метода маркирования текстовых документов. В основе первого метода — незначительное **смещение** слова по вертикали. Второй метод изменяет яркость начертания шрифта вдоль линии, проходящей посередине слова – визуально данный эффект напоминает **перечеркивание** слова.

Алгоритмы рассчитаны на работу с растровыми изображениями текстовых документов, разрешение которых превышает 150 DPI. Каждая страница многостраничного текстового документа маркируется по отдельности. На каждой странице документа должно присутствовать не менее 12 строк текста, а в каждой строке – не менее пяти слов для маркирования посредством первого алгоритма и не менее трех слов для маркирования с помощью второго.

3.1. Общая схема работы

Несмотря на то, что механизмы работы разработанных алгоритмов кодирования различны, значительная часть логики обработки документов совпадает. На Рис. 2 представлена последовательность выполнения этапов для различных операций маркирования документа. Общие этапы обработки документа, представленные на схеме, включают:

- *коррекция угла поворота*: корректировка осуществляется в небольшом диапазоне углов и не исправляет ошибки ориентации документа;
- *разметка документа*: выделение на изображении документа слов, текстовых линий, текстовых блоков;
- *фильтрация*: составление набора элементов разметки (слов или текстовых линий), которые не будут использоваться при кодировании;
- *разбиение на группы*: объединение слов в текстовых линиях в группы, кодирующих один бит метки;
- *обратная коррекция угла поворота*: восстановление исходного угла поворота документа.

3.1.1. Встраивание метки

Входные данные: растровое изображение документа, 32-битная метка;

Выходные данные: растровое изображение документа с внедренной меткой.

Краткое описание этапов обработки во время встраивания метки:

- *Генерация метки с БЧХ кодом*: преобразование входной метки в последовательность битов, добавление блока коррекции ошибок на основе БЧХ-кода;
- *Распределение бит метки по группам*: вычисление соответствий между битами метки и группами, которые их кодируют;
- *Преобразование документа* в соответствии с выбранным методом представления метки: перечеркивания (раздел 4) или смещения (раздел 5).

3.1.2. Извлечение метки

Входные данные: растровое изображение документа с внедренной меткой;

Выходные данные: 32-битная метка.

Краткое описание этапов обработки во время извлечения метки:

- *Анализ преобразований*: извлечение закодированных бит из каждой группы, объединение в битовую последовательность. Реализация данного этапа зависит от выбранного метода: перечеркивания (раздел 4) или смещения (раздел 5);
- *Исправление инверсий метки*: разделение извлеченной битовой последовательности на метку и БЧХ-коды коррекции. Обнаружение инверсий бит, если количество ошибок не слишком велико.

3.1.3. Стирание метки

Входные данные: растровое изображение документа с внедренной меткой;

Выходные данные: растровое изображение документа со стертой меткой.

Краткое описание этапов обработки во время стирания метки:

- *Обратное преобразование документа* осуществляется по группам. Реализация данного этапа зависит от выбранного метода: перечеркивания (раздел 4) или смещения (раздел 5)

3.2. Разметка документа

Разработанные алгоритмы маркирования оперируют над словами и текстовыми линиями, поэтому один из этапов работы алгоритмов – получение разметки текста. *Результат выполнения разметки текста* — это информация о минимальных ограничивающих слова прямоугольниках, базовых линиях и взаимном расположении элементов на изображении текстового документа. *Минимальный ограничивающий прямоугольник (МОП)* —

прямоугольник минимальной площади, внутри которого слово или текстовая линия находится целиком. *Базовая линия* — воображаемая линия, проходящая по нижнему краю слова без учета нижних выносных элементов (пример на Рис. 3).

Для получения разметки текста используются инструменты сегментации. В данный момент поддерживается два инструмента:

- **Tesseract OCR** — это свободно распространяемая программа для распознавания текста [23];
- **DocParser** — решение, разработанное партнерами ИСП РАН на основе нейронной сети архитектуры UNet [24].

При работе с инструментами сегментации был выявлен ряд проблем. Главная проблема – несовпадение разметки оригинально документа и разметки сканированного изображения того же документа. Такое несовпадение может сильно повлиять на корректность внедрения и точность извлечения метки. В ходе работы были выделены следующие типы ошибок:

- разбиение одного слова на два;
- слияние двух слов в одно;
- выделение различных дефектов (шум, дефекты сканирования) как отдельных текстовых элементов;
- некорректная работа с документами, содержащими таблицы;
- некорректная работа с документами, маркированными *методом перечеркивания*.

3.2. Механизмы обработки ошибок

Вышеизложенные проблемы инструментов сегментации текста влияют на корректность извлечения метки. Для повышения точности алгоритмов было разработано несколько техник, которые позволяют значительно уменьшить влияние ошибок разметки на корректность извлечения метки. Среди них:

- **Группировка слов в текстовых линиях**. Суть данного метода заключается в том, что один бит метки кодируется не одним словом, а группой из нескольких слов. Каждая строка документа делится на N (по умолчанию $N = 3$) равных по ширине частей, каждое слово в текстовой линии попадет в определенную группу в зависимости от своего положения. Группа может включать несколько слов или быть пустой. Данный метод позволяет частично нивелировать ошибки разметки, связанные с некорректным определением границ слов.
- **Коды коррекции ошибок**. Метка, которая внедряется в документ, при необходимости может быть дополнена блоком обнаружения и исправления битовых инверсий. Для коррекции ошибок используется БЧХ-код [25], который в теории позволяет корректировать произвольное количество инверсий, однако увеличение количества обнаруживаемых ошибок требует увеличения длины блока коррекции. По умолчанию корректируется 3 ошибки – длина блока с кодами коррекции при этом составляет 18 бит.
- **Фильтрация элементов**. Некоторые слова или текстовые линии могут не подходить для кодирования информации, ввиду того что они являются источником ошибок при разметке или из-за специфики конкретного алгоритма кодирования. При внедрении и извлечении метки такие элементы пропускаются и не используются для кодирования. Используются следующие критерии: ширина слов, количество слов в текстовой линии, нахождение слова в таблице и т.д.
- **Детектирование таблиц**. Инструменты сегментации плохо справляются с разметкой документов, в которых присутствуют таблицы. Для решения этой проблемы все слова,

которые находятся внутри границ таблиц, отфильтровываются. Разработанный алгоритм детектирования таблиц находит внешние контуры всех таблиц в документе и помечает слова, находящиеся внутри этих контуров. Поиск внешних контуров осуществляется на основе топологического анализа бинаризованных изображений [26].

3.3. Позиционирование метки в документе

Если в документе присутствует больше текстовых линий, чем необходимо для однократного внедрения метки, выполняется поиск наиболее подходящей области встраивания метки. Область встраивания представляет собой набор текстовых линий, следующих друг за другом, ограниченный специальными строками-маркерами. При выборе позиции для встраивания метки учитываются следующие факторы (в порядке приоритета):

1. наличие в строке слов, не являющихся машинописным текстом[1] (меньше — лучше);
2. количество пустых групп (меньше — лучше);
3. «заполненность» текстовой линии, вычисляемая по формуле ниже (больше — лучше).

$$tl^i_{fat} = \frac{tl^i_{width}}{tl^W_{width}} \cdot \frac{\sum^{\overline{tl}^i} w^{i,j}_{area}}{tl^i_{area}}$$

- tl^i_{width} — ширина МОП i текстовой линии;
- tl^i_{area} — площадь МОП i текстовой линии;
- $w^{i,j}_{area}$ — площадь МОП j слова в i текстовой линии;
- $\overline{tl}^i$ — количество слов в i текстовой линии;
- W — номер самой широкой текстовой линии.

Область встраивания метки – заданный диапазон строк – ограничивается посредством специальных строк-маркеров начала и конца метки. В качестве метода преобразования для строк-маркеров начала и конца метки используется алгоритм перечеркивания слов.

4. Алгоритм кодирования на основе перечеркивания слов

4.1. Внедрение метки

Алгоритм кодирования использует подход с изменением яркости глифов в областях пересечения горизонтальной линии, проходящей по длине слова между *базовой* и *медианной* линиями, а также имеющей толщину σ. Визуально данный эффект напоминает перечеркивание слова (рис. 3). Толщина линии σ вычисляется на основе средней высоты слов во всем документе. Перечеркнутое слово кодирует единичный бит, неперечеркнутое слово – нулевой (рис. 4).

Рис. 3. Расположение перечеркивающей линии
Fig. 3. Strikethrough line location

[1] Для определения типа содержимого слова используется нейронная сеть на основе архитектуры ResNet [27].

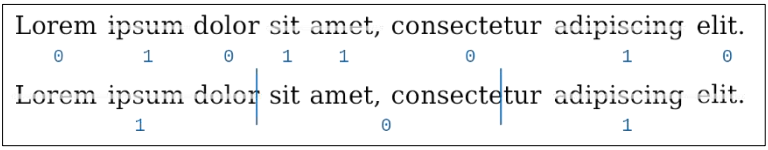


Рис. 4. Перечеркивание слов: обычное (выше), с группировкой (ниже)
Fig. 4. Words strikethrough: simple (upper), with grouping (below)

4.2. Извлечение метки

Инструменты получения разметки документа плохо работают с документами, в которые внедрена метка посредством перечеркивания слов. Для решения этой проблемы перед получением разметки проводится операция **восстановления документа**, которая основана на морфологической операции *эрозии*[2]. Для корректной работы алгоритма восстановления требуется знать значение толщины перечеркивающей линии в пикселях σ. На этапе внедрения метки данная величина динамически вычисляется, используя полученное в ходе разметки значение средней высоты строк. Для алгоритма восстановления был выбран подход, который аппроксимирует среднюю высоту строк без использования разметки документа.

После получение разметки алгоритм извлечения метки для каждого слова определяет, перечеркнуто оно или нет. Алгоритм детектирования перечеркивания в слове работает следующим образом:

1. Фрагмент изображения, содержащий слово, бинаризуется;
2. Проводится множество линий от левого края МОП слова к правому. Если нашлась хотя бы одна линия, не пересекающая черных пикселей, то слово перечеркнуто и кодирует 1, иначе – 0.

4.3. Стирание метки

На текущий момент существует два подхода к стиранию метки: морфологический и нейросетевой. *Морфологический* подход использует описанный в разделе 4.2 алгоритм восстановления документа. К восстановленному документу применяется операция *эрозии*, вычисляются области перечеркивания слов, после чего фрагменты слов из восстановленного документа накладываются на исходный документ. Качество работы алгоритма стирания метки зависит, в том числе от содержащихся в слове букв. Например, в кириллических буквах *а, в, е, н, ю, э, и, й, ж, з* высока вероятность появление дефектов.

Нейросетевой алгоритм стирания метки, использующий сверточную нейронную сеть UNet [24], дает меньшее количество дефектов (рис. 5). Архитектура UNet была выбрана среди прочих, поскольку хорошо справляется с задачей сегментации изображений. Полная сверточность нейросети позволяет в случае необходимости изменить размер входного изображения. Для обучения нейронной сети был собран набор данных, состоящий из 20427 изображений слов. На каждое изображение слова накладывалось перечеркивание с определенными параметрами и генерировалась соответствующая *маска перечеркивания*. Бинаризованное изображение перечеркнутого слова является входом нейронной сети, а ожидаемым выходом является *маска перечеркивания*.

[2] Операция свертки изображения с прямоугольным ядром. Визуально данная операция вызывает расширение темных областей на изображении.

Рис. 5. Сравнение методов стирания. Сверху слово с внедренной меткой, посередине - морфологический подход, снизу - нейросетевой подход
Fig. 5. Wipe methods comparison. On top word with label injected, on the middle - morphological approach, bottom - neural network approach

5. Алгоритм кодирования на основе вертикального смещения слов

Как упоминалось ранее, среди структурных методов кодирования распространены алгоритмы, использующие смещение текстовых линий и параграфов по вертикали или слов по горизонтали. Однако не встречается работ на основе вертикального смещения слов, поэтому была предпринята попытка реализовать подобный алгоритм.

5.1 Внедрение метки

В ходе работы был разработан алгоритм кодирования метки на основе вертикального смещения слов. Первое и последнее слово в строке обозначаются *опорными* и не смещаются, что позволяет задать нулевой уровень смещения и оценивать смещение остальных слов относительно *опорных*. Уровень слова определяется по его *базовой линии*. Данный метод использует следующую схему кодирования: если слово кодирует 1, то оно смещается вверх на β пикселей, если 0 — остается на месте. Значение β вычисляется на основе средней высоты слов во всем документе. При использовании группировки все слова, находящиеся в одной группе, кодируют один бит, а значит, имеют одно и то же смещение. Пример кодирования вертикальным смещением представлен на Рис. 6, величина β намеренно увеличена для наглядности.

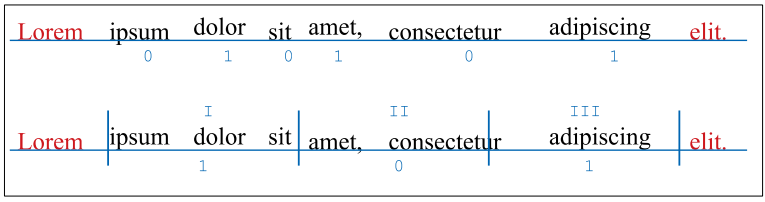


Рис. 6. Вертикальное смещение слов: обычное (выше), с группировкой (ниже)
Fig. 6. Vertical word-shift: simple (upper), with grouping (below)

При смещении все содержимое МОПа слова смещается вместе с ним. Фрагмент изображения, который при смещении перекрывается словом, также смещается и замещает освободившееся пространство.

5.2. Извлечение метки

Алгоритм извлечения метки работает построчно и определяет смещение для каждого слова в строке. При считывании закодированной последовательности используется смещение базовой линии слова относительно предыдущего. Наиболее вероятная последовательность смещений слов в строке вычисляется при помощи алгоритма Витерби. Информация о том, что первое и последнее слова в строке не смещены, позволяет отсечь множество вариантов и выбрать только один. После считывания последовательности из строки определяется соответствие между словом и закодированным битом. В силу искажений различного

характера и ошибок разметки возможна ситуация, когда словам в одной группе соответствуют разные биты, поэтому при выборе бита группы вычисляется взвешенная сумма для каждого значения. Вес для каждого слова вычисляется индивидуально на основе следующих факторов:

1. ширина слова (больше — лучше);
2. количество символов в слове (больше — лучше);
3. наличие символов с выносными элементами[3] (меньше — лучше).

5.3. Стирание метки

При стирании метки используется информация о смещениях в каждой строке, полученная при помощи алгоритма Витерби аналогично тому, как это делается при извлечении метки. После извлечения битовой последовательности из строки выполняется обратное смещение для каждого слова индивидуально, без учета распределения слов по группам.

Строки-маркеры начала и конца области встраивания метки восстанавливаются при помощи тех же алгоритмов, что используются при стирании в алгоритме маркирования методом перечеркивания слов.

6. Результаты тестирования

Разработанные в ходе исследования алгоритмы были реализованы и протестированы. Инструментов маркирования текстовых документов на основе структурных алгоритмов найдено не было. Для тестирования алгоритмов была собрана выборка из 40 документов различного содержимого и форматирования. Документы для данной выборки были загружены с сайта Министерство образования и науки РФ (URL: https://minobrnauki.gov.ru/). В каждый документ внедрялось 4 метки:

1) случайная метка (сгенерированная перед тестированием),
2) инверсия случайной метки,
3) метка, состоящая только из единиц,
4) метка, состоящая только из нулей.

Испытания проводились по трем сценариям (**П** – печать, **С** – сканирование, **Ф** – фотографирование):

- **ПС**. В документ внедряется метка, документ с меткой печатается, распечатанный документ сканируется, из скана документа извлекается метка;
- **ПСПС**. Из скана документа из первого сценария стирается метка, в документ со стертой меткой внедряется инверсия предыдущей метки, документ с новой меткой печатается, распечатанный документ сканируется, из скана документа извлекается метка. В данном сценарии использовались только случайная метка и ее инверсия;
- **ПФ**. В документ внедряется метка, документ с меткой печатается, распечатанный документ фотографируется, из фотографии документа извлекается метка. В данном сценарии применялись только случайная метка и ее инверсия.

Для оценки результатов используются следующие метрики:

- **Внедрено** – доля документов, в которые удалось встроить метку. Некоторые документы не могут быть использованы для встраивания метки определенным алгоритмом ввиду недостаточного количества текста;
- **Извлечено** – доля документов, из которых удалось извлечь метку;
- **Точность** — средняя доля одинаковых бит при внедрении и извлечении метки (без

[3] Наличие нижних выносных элементов повышает вероятность корректного вычисления базовой линии.

блока с кодами коррекции);

- **Полнота относительная** — доля извлеченных меток с точностью равной 1;
- **Полнота абсолютная** – доля извлеченных меток с точностью равно 1 среди всех документов.

Алгоритм маркирования на основе вертикального смещения слов показывает лучшие значения метрик абсолютной полноты, что дает кумулятивную оценку точности и универсальности (табл. 1, 2, рис. 7-9)). Большее число успехов при внедрении метки первым алгоритмом обуславливается меньшим количеством слов в текстовой линии, необходимым для разбиения на группы. Снижение точности извлечения в сценарии *ПФ* по сравнению с другими сценариями объясняется значительными искажениями, в частности, аберрацией и дисторсией.

Табл. 1. Результаты тестирования алгоритмом на основе смещения слов
Table 1. Test results of the algorithm based on word shift

Тестовый сценарий	*Смещение*				
	Внедрено	*Извлечено*	*Точность*	*Полнота относительная*	***Полнота абсолютная***
ПС	88/160	74/160	0.980997	0.864865	0.4000
ПСПС	88/160	71/160	0.898768	0.521127	0.2312
ПФ	44/80	35/80	0.829464	0.285714	0.1249

Табл. 2. Результаты тестирования алгоритмом на основе перечеркивания слов
Table 2. Test results of the algorithm based on word strikethrough

Тестовый сценарий	*Перечеркивание*				
	Внедрено	*Извлечено*	*Точность*	*Полнота относительная*	***Полнота абсолютная***
ПС	148/160	147/160	0.966837	0.897959	0.8249
ПСПС	148/160	148/160	0.944046	0.871622	0.8062
ПФ	40/40	39/40	0.737981	0.282051	0.2749

РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНОБРНАУКИ РОССИИ)

ПРИКАЗ

16 декабря 2010 г. № 803/нк

Москва

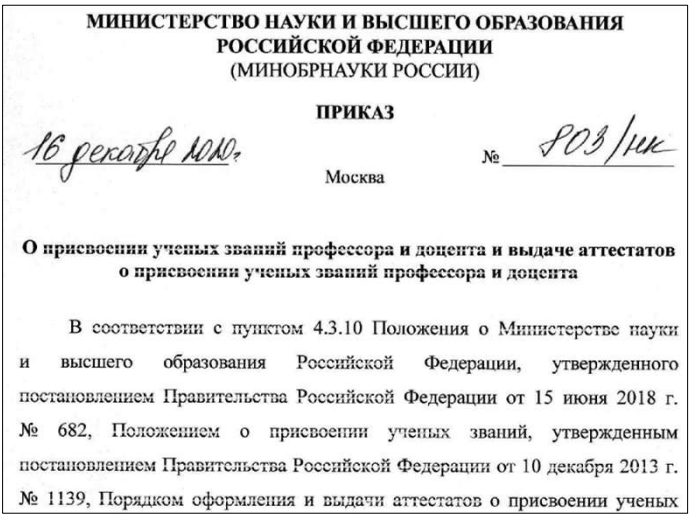
О присвоении ученых званий профессора и доцента и выдаче аттестатов о присвоении ученых званий профессора и доцента

В соответствии с пунктом 4.3.10 Положения о Министерстве науки и высшего образования Российской Федерации, утвержденного постановлением Правительства Российской Федерации от 15 июня 2018 г. № 682, Положением о присвоении ученых званий, утвержденным постановлением Правительства Российской Федерации от 10 декабря 2013 г. № 1139, Порядком оформления и выдачи аттестатов о присвоении ученых

Рис. 7. Фрагмент отсканированного документа без метки
Fig. 7. Fragment of scanned document without label

7. Заключение

В рамках исследования были разработаны алгоритмы маркирования выводимых на печать текстовых документов на основе вертикального сдвига и перечеркивания слов. Возможность извлечения метки без использования оригинального документа является преимуществом перед продуктами, которые уже присутствуют на рынке, а возможность стереть метку из маркированного документа является нововведением по сравнению с подходами, которые описаны в опубликованных ранее работах. В ходе работы было проведено тестирование программных реализаций алгоритмов на всех предполагаемых сценариях использования, включая стирание метки и ее повторное внедрение. Результаты тестирования показали практическую применимость разработанных методов маркирования.



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНОБРНАУКИ РОССИИ)

ПРИКАЗ

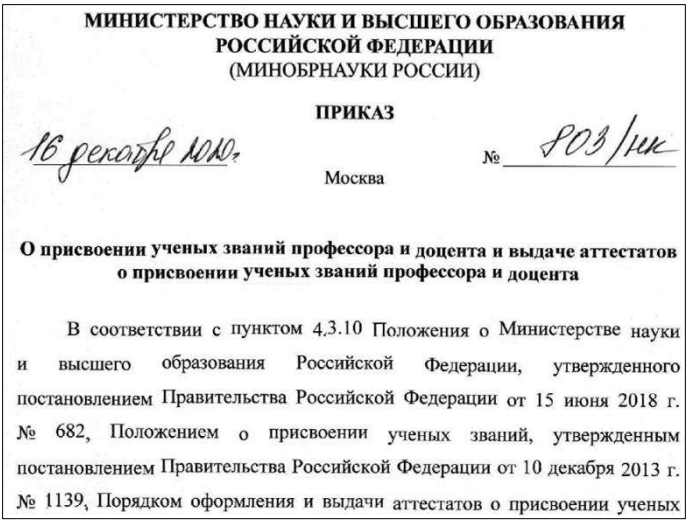
16 декабря 2010 г. № 803/нк

Москва

О присвоении ученых званий профессора и доцента и выдаче аттестатов о присвоении ученых званий профессора и доцента

В соответствии с пунктом 4.3.10 Положения о Министерстве науки и высшего образования Российской Федерации, утвержденного постановлением Правительства Российской Федерации от 15 июня 2018 г. № 682, Положением о присвоении ученых званий, утвержденным постановлением Правительства Российской Федерации от 10 декабря 2013 г. № 1139, Порядком оформления и выдачи аттестатов о присвоении ученых

Рис. 8. Фрагмент отсканированного документа с меткой на основе алгоритма перечеркивания слов
Fig. 8. Fragment of scanned document with label based on word strikethrough algorithm



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
(МИНОБРНАУКИ РОССИИ)

ПРИКАЗ

16 декабря 2010 г. № 803/нк

Москва

О присвоении ученых званий профессора и доцента и выдаче аттестатов о присвоении ученых званий профессора и доцента

В соответствии с пунктом 4.3.10 Положения о Министерстве науки и высшего образования Российской Федерации, утвержденного постановлением Правительства Российской Федерации от 15 июня 2018 г. № 682, Положением о присвоении ученых званий, утвержденным постановлением Правительства Российской Федерации от 10 декабря 2013 г. № 1139, Порядком оформления и выдачи аттестатов о присвоении ученых

Рис. 9. Фрагмент отсканированного документа с меткой на основе алгоритма вертикального смещения слов
Fig. 9. Fragment of scanned document with label based on word shift algorithm

Пути дальнейшего развития включают следующие направления:

- повышение качества работы при извлечении метки из фотографии;
- уменьшение визуальной заметности метки;
- разработка методов эмуляции дефектов, которые могут вносить сканеры, что позволит значительно упростить тестирование;
- повышение стабильности работы инструментов сегментации текста на сканах и фотографиях.

Список литературы / References

[1] Утечки информации ограниченного доступа: отчет за 9 месяцев 2020 г. Экспертно-аналитический центр InfoWatch, 2020 г. / Restricted information leaks: report for 9 months of 2020. InfoWatch Analytical Center, 2020 (in Russian).

[2] Козлов С.В., Копылов С.А. и др. Реализация маркирования в подсистеме печати ОС семейства Windows на основе виртуального XPS-принтера. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 95-110 / Kozlov S.V., Kopylov S.A. et al. Implementing watermarking based on a virtual XPS printer for Windows operating systems. rudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 95-110 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–7.

[3] Dong P., Galatsanos N. P. Affine transformation resistant watermarking based on image normalization. In Proc. of the International Conference on Image Processing, 2002, pp. 489-492.

[4] Pramila A., Keskinarkaus A., Seppänen T. Multiple domain watermarking for print-scan and JPEG resilient data hiding. Lecture Notes in Computer Science, vol. 5041, 2007, pp. 279-293.

[5] Ahmed Q., Munib S., Mirza M. T., Khan A. Smart phone based online medicine authentication using print-cam robust watermarking. In Proc. of the 13th International Conference on Frontiers of Information Technology (FIT), 2015, pp. 222-227.

[6] Ahvanooey M.T., Li Q. et al Modern text hiding, text steganalysis, and applications: a comparative analysis. Entropy, vol. 21, no. 4, 2019, article 355.

[7] Khadam U., Iqbal M.M. et al. Digital watermarking technique for text document protection using data mining analysis. IEEE Access, vol. 7, 2019, pp. 64955-64965.

[8] Por L. Y., Wong K., Chee K. O. UniSpaCh: A text-based data hiding method using Unicode space characters. Journal of Systems and Software, vol. 85, no. 5, 2012, pp. 1075-1082.

[9] Bender W., Gruhl D. et al. Techniques for data hiding. IBM Systems Journal, vol. 35, issue 3.4, 1996, pp. 313-336.

[10] Leea I.S. Secret communication through web pages using special space codes in HTML files. International Journal of Applied Science and Engineering, vol. 6, no. 2, 2008, pp. 141-149.

[11] Ahvanooey M.T., Tabasi S.H. A new method for copyright protection in digital text documents by adding hidden unicode characters in persian/english texts. International Journal of Current Life Sciences, vol. 4, no. 8, 2014, pp. 4895-4900.

[12] Ahvanooei M.T., Tabasi S.H., Rahmani S. A novel approach for text watermarking in digital documents by zero-width interword distance changes. DAV International Journal of Science, vol. 4, no. 3, 2015, pp. 550-558.

[13] Low S.H., Maxemchuk N.F. et al. Document marking and identification using both line and word shifting. In Proc. of the INFOCOM'95, 1995, pp. 853-860.

[14] Brassil J. T., Low S. et al. Electronic marking and identification techniques to discourage document copying. IEEE Journal on Selected Areas in Communications, vol. 13, issue 8, 1995, pp. 1495-1504.

[15] Alattar A.M., Alattar O.M. Watermarking electronic text documents containing justified paragraphs and irregular line spacing. Proceedings of the SPIE, vol. 5306, Security, Steganography, and Watermarking of Multimedia Contents VI, 2004, pp. 685-695.

[16] Kim Y.W., Moon K.A., Oh I.S. A Text Watermarking Algorithm based on Word Classification and Inter-word Space Statistics. In Proc. of the Seventh International Conference on Document Analysis and Recognition, 2003, pp. 775-779.

[17] Kozachok A.V., Kopylov S. Estimation of Watermark Embedding Capacity with Line Space Shifting. In Proc. of the 2020 Ivannikov Memorial Workshop (IVMEM), 2020, pp. 29-34.

[18] Tan L., Hu K. et al. Print-scan invariant text image watermarking for hardcopy document authentication. Multimedia Tools and Applications, vol. 78, no. 10, 2018, pp. 13189-13211.
[19] Xiao C., Zhang C., Zheng C. Fontcode: Embedding information in text documents using glyph perturbation. ACM Transactions on Graphics (TOG), vol. 37, no. 2, 2017, pp. 1-16.
[20] Gutub A., Fattani M. A novel Arabic text steganography method using letter points and extensions. In Proc. of the WASTET International Conference on Computer, Information and System Science and Engineering (ICCISSE), 2007, pp. 28-31.
[21] Secret Technologies – Trace Doc. Available at https://secretgroup.ru/trace-doc/, accessed 09.04.2021.
[22] EVERYTAG – Information Leaks Detection (ILD). Available at https://everytag.ru/, accessed 09.04.2021.
[23] Smith R. An overview of the Tesseract OCR engine. In Proc. of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 2007, pp. 629-633.
[24] Ronneberger O., Fischer P., Brox T. U-net: Convolutional networks for biomedical image segmentation. Lecture Notes in Computer Science, vol. 9351, 2015, pp. 234-241.
[25] Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. Техносфера, 2005, 320 стр. / Morelos-Zaragoza R. The Art of Error Correcting Coding. Willey, 2002, 238 p.
[26] Suzuki S. Topological structural analysis of digitized binary images by border following. Computer vision, graphics, and image processing, vol. 30, no. 1, 1985, pp. 32-46.
[27] He K., Zhang X. et al. Deep residual learning for image recognition. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770-778.

Информация об авторах / Information about authors

Дмитрий Олегович ОБЫДЕНКОВ – аспирант. Научные интересы: методы сокрытия и защищённой передачи информации, компьютерные сети, технологии анализа сетевого трафика.

Dmitry Olegovich OBYDENKOV is a graduate student. Research interests: methods for information hiding and secure transmission, computer networks, technologies of network traffic analysis.

Александр Евгеньевич ФРОЛОВ – студент-магистр. Научные интересы: стеганография, методы анонимизанции сетевого трафика, компьютерные сети, методы машинного обучения.

Alexander Evgenevich FROLOV is a master student. Research interests: steganography, traffic anonymization methods, computer networks, machine learning.

Юрий Витальевич МАРКИН, научный сотрудник, кандидат технических наук. Область научных интересов: информационная безопасность, анализ сетевого трафика, обработка изображений, алгоритмы машинного обучения.

Yury Vitalievich MARKIN, researcher, PhD. Research interests: information security, network traffic analysis, image processing, machine learning algorithms.

Станислав Александрович ФОМИН — ведущий программист. Область научных интересов: теория сложности, алгоритмы дискретной оптимизации, верификация ПО, архитектура информационных систем. URL: https://discopal.ispras.ru/stas.

Stanislav Alexandrovich FOMIN — leading programmer. Research interests: complexity theory, discrete optimization algorithms, information systems architecture.

Борис Владимирович КОНДРАТЬЕВ; сфера научных интересов: безопасность информации, защита информации от несанкционированного доступа и утечки по техническим каналам, построение информационных систем в защищённом исполнении, сертификация программного обеспечения по требованиям безопасности информации.

Boris Vladimirovich KONDRAT'EV, research interests: information security, protection of information from unauthorized access and leakage through technical channels, building information systems in a secure design, certification of software for information security requirements.

DOI: 10.15514/ISPRAS-2021-33(5)-5

CC BY

Методика сбора обучающего набора данных для модели обнаружения компьютерных атак

[1,2] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*
[3] *М.Н. Горюнов, ORCID: 0000-0003-0284-690X <max.gor@mail.ru>*
[3] *А.Г. Мацкевич, ORCID: 0000-0001-9557-3765 <mag3d.78@gmail.com>*
[3] *Д.А. Рыболовлев, ORCID: 0000-0003-4524-655X <dmitrij-rybolovlev@yandex.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Национальный исследовательский университет «Высшая школа экономики», 101978, Россия, г. Москва, ул. Мясницкая, д. 20*
[3] *Академия ФСО России 302015, Россия, г. Орел, ул. Приборостроительная, д. 35*

Аннотация. В работе рассмотрены вопросы обучения моделей обнаружения компьютерных атак, основанных на применении методов машинного обучения. Последовательно представлены результаты анализа общедоступных обучающих наборов данных и инструментов анализа сетевого трафика и выделения признаков сетевых сессий. Отмечены недостатки существующих инструментов и возможные ошибки в формируемых с их помощью наборах данных. Сделан вывод о необходимости сбора собственных обучающих данных в условиях отсутствия гарантий достоверности общедоступных наборов данных и ограниченного применения предобученных моделей в сетях с характеристиками, отличными от характеристик сети, в которой производился сбор обучающего трафика. Предложен практический подход к формированию данных обучения для моделей обнаружения компьютерных атак. Произведена апробация предлагаемых решений с целью оценки качества обучения модели на собранных данных и качества обнаружения атак в условиях реальной сетевой инфраструктуры.

Ключевые слова: информационная безопасность; система обнаружения атак; машинное обучение; набор данных; перенос обучения; случайный лес; сетевой трафик; компьютерная атака

Для цитирования: Гетьман А.И., Горюнов М.Н., Мацкевич А.Г., Рыболовлев Д.А. Методика сбора обучающего набора данных для модели обнаружения компьютерных атак. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 83-104. DOI: 10.15514/ISPRAS–2021–33(5)–5

Methodology for Collecting a Training Dataset for an Intrusion Detection Model

[1,2] *A.I. Getman, ORCID: 0000-0002-6562-9008<ever@ispras.ru>*
[3] *M.N. Goryunov, ORCID: 0000-0003-0284-690X <max.gor@mail.ru>*
[3] *A.G. Matskevich, ORCID: 0000-0001-9557-3765 <mag3d.78@gmail.com>*
[3] *D.A. Rybolovlev, ORCID: 0000-0003-4524-655X <dmitrij-rybolovlev@yandex.ru>*

[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *HSE University 20, Myasnitskaya Ulitsa, Moscow, 101978, Russia*
[3] *The Academy of Federal Security Guard Service of the Russian Federation, 35, Priborostroitelnaya st., Oryol, 302015, Russia*

Abstract. The paper discusses the issues of training models for detecting computer attacks based on the use of machine learning methods. The results of the analysis of publicly available training datasets and tools for analyzing network traffic and identifying features of network sessions are presented sequentially. The drawbacks of existing tools and possible errors in the datasets formed with their help are noted. It is concluded that it is necessary to collect own training data in the absence of guarantees of the public datasets reliability and the limited use of pre-trained models in networks with characteristics that differ from the characteristics of the network in which the training traffic was collected. A practical approach to generating training data for computer attack detection models is proposed. The proposed solutions have been tested to evaluate the quality of model training on the collected data and the quality of attack detection in conditions of real network infrastructure.

Keywords: information security; network intrusion detection system; machine learning; dataset; transfer learning; random forest; network traffic; computer attack

For citation: Getman A.I., Goryunov M.N., Matskevich A.G., Rybolovlev D.A. Methodology for Collecting a Training Dataset for an Intrusion Detection Model. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 83-104 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–5

1. Введение

Развитие современного общества неразрывно связано с применением широкого спектра информационных технологий: поисковых систем, веб-сервисов, служб обмена файлами, сервисов онлайн-платежей и многих других, что неизбежно сопровождается ростом числа угроз информационной безопасности. Одной из угроз, представляющих наибольшую опасность, является угроза осуществления компьютерных атак на информационные сервисы. Данный вид воздействия, как правило, приводит к нарушению корректного функционирования сервисов, раскрытию конфиденциальной информации пользователей, финансовым потерям и другим негативным последствиям.

Для обнаружения компьютерных атак в основном используются сигнатурные анализаторы сетевого трафика, эффективность которых ограничена полнотой базы решающих правил обнаружения известных информационных воздействий. Очевидным недостатком такого подхода является практически нулевая вероятность обнаружения неизвестных или модифицированных сетевых атак, а также невозможность анализа зашифрованного трафика (например, защищенного с помощью протокола TLS), что подтверждает необходимость разработки новых эвристических алгоритмов классификации сетевого трафика.

Перспективным подходом к созданию несигнатурных методов обнаружения компьютерных атак является применение технологий машинного обучения. Вместе с тем, разработка эвристического анализатора сетевого трафика на основе машинного обучения требует решения ряда сложных задач:

- выбора признакового пространства для адекватного описания того или иного сетевого

трафика;
- формирования обучающего набора данных;
- выбора и обучения модели машинного обучения;
- разработки алгоритма классификации трафика, оценки его эффективности;
- практической реализации, апробации и внедрения алгоритма.

Решению одной из вышеперечисленных задач – задачи формирования обучающего набора данных – и посвящена данная статья. Цель настоящего исследования состоит в разработке методики сбора данных для обучения модели обнаружения сетевых компьютерных атак. Для достижения сформулированной цели необходимо решить следующие задачи:

- провести анализ существующих общедоступных наборов данных, предназначенных для обучения эвристических алгоритмов обнаружения компьютерных атак;
- выявить особенности функционирования инструментов анализа сетевого трафика и выделения признаков сетевых сессий;
- выработать требования, предъявляемые к создаваемым наборам данных для обучения;
- разработать стенды для моделирования компьютерных атак и фонового трафика пользователей;
- предложить методику сбора обучающего набора данных и провести ее апробацию в отношении выбранного объекта защиты с последующей оценкой качества обнаружения атак.

Новизна работы заключается в применении системного подхода к решению вопроса обучения модели обнаружения компьютерных атак в условиях отсутствия гарантий достоверности общедоступных обучающих наборов данных, а также вариативности характеристик защищаемой сети и возникающих по этой причине сложностей практического применения предобученных моделей.

2. Анализ релевантных работ

Вопросы применения методов машинного обучения для обнаружения компьютерных атак и возникающие при этом сложности в получении аккуратно размеченных данных для обучения активно обсуждаются в последние годы. По указанной тематике опубликовано достаточное количество работ, которые могут служить основой дальнейших исследований.

В статье [1] сформулированы перспективные направления исследований в области кибербезопасности, среди которых выделена задача развития практики применения методов искусственного интеллекта и машинного обучения. Отмечается важность выбора признакового пространства, разметки данных для обучения. Приводится список наиболее используемых общедоступных наборов данных для задач кибербезопасности, однако отсутствуют сведения о практической применимости моделей, предобученных на таких данных.

В работе [2] подчеркивается, что в условиях постоянного появления новых типов компьютерных атак актуальной является задача разработки наборов данных, содержащих современные типы атак. Такая задача подразумевает наличие у разработчика соответствующих экспертных знаний в области построения распределенных тестовых стендов, современных сетевых технологий, моделирования компьютерных атак и др. Проведен анализ общедоступных наборов данных, формализованы требования к создаваемым наборам данных для обучения: разнородности представленных атак, наличия полной конфигурации сети, полного сетевого взаимодействия, разнородности протоколов и др. Сформирован публичный набор данных CICIDS2017, который впоследствии получил широкое распространение в исследовательских проектах. Вместе с тем в статье не приводятся результаты апробации предлагаемых решений в реальной сети и не оценивается возможное снижение качества обнаружения атак по причине различия характеристик защищаемой сети и сети, в которой производился сбор обучающего трафика.

В исследовании [3] представлен обзор 34 общедоступных наборов данных с указанием их отличительных особенностей, используемых сценариев атак, имеющихся недостатков. Отдельно отмечается недостаточная репрезентативность существующих наборов данных для обучения, что, по мнению авторов, является одним из основных препятствий при построении систем обнаружения атак. Предложена методология оценки применимости наборов данных к различным задачам информационной безопасности. Среди проанализированных наборов данных для использования в практических задачах авторы рекомендуют наборы CICIDS 2017, CIDDS-001, UGR-16 и UNSW-NB15. Вместе с тем не приводятся практические предложения по устранению известных недостатков этих наборов данных, а представленные рекомендации по сбору собственных наборов носят общий характер.

В работе [4] рассматриваются задача классификации сетевого трафика и возможности применения методов машинного обучения для ее решения. Исследуется вопрос формирования признакового пространства, обсуждаются существующие проблемы получения данных для обучения и основные компромиссы в этом вопросе. Перечисляются часто используемые общедоступные наборы данных и их характеристики. Авторы отмечают, что одним из вариантов получения адекватных данных для обучения является формирование своего собственного набора данных. Однако в статье не представлена система требований к создаваемым наборам данных.

В статье [5] рассматривается один из аспектов проблемы transfer learning в исследуемой предметной области – изменение качества работы классификатора сетевого трафика, предобученного в сети с характеристиками, отличными от характеристик защищаемой сети. Продемонстрировано снижение качества обнаружения сетевых атак при переносе предобученной модели в другую сеть. Проведены эксперименты по определению дополнительного объема данных дообучения, достаточного для восстановления исходного качества предобученного классификатора. Вместе с тем в качестве данных для обучения в работе используются публичные наборы NIMS2018 и UNB2015 с соответствующими архитектурами сетей, и не представлены практические предложения по дообучению моделей обнаружения атак в сетях с отличными характеристиками.

В работе [6] представлены результаты анализа актуальных наборов данных для обучения систем обнаружения сетевых атак. Авторы рекомендуют оценивать качество обнаружения атак, применяя несколько наборов данных, чтобы избежать переобучения. Для использования в практических задачах авторы рекомендуют наборы UNSW-NB15, CIDDS-001, CICIDS 2017 и CSE-CIC-IDS 2018, указывая при этом их недостатки.

В исследовании [7] подчеркивается важность этапа формирования признакового пространства, сбора и разметки данных. На примере широко используемого, но устаревшего набора данных KDD Cup 1999, отмечаются недостатки существующих публичных наборов данных. Однако для оценки предлагаемых решений авторы также используют публичный набор данных NGIDS-DS, не подтверждая возможность применения предобученной модели в защищаемой сети с характеристиками, отличными от характеристик сети, в которой производился сбор трафика.

Достаточная интерпретируемость широко распространенных моделей машинного обучения позволяет применить к их практическим реализациям известный в информатике принцип «garbage in, garbage out» и подчеркнуть при этом важность использования при обучении адекватных, аккуратно размеченных данных. В отмеченных выше работах подробно рассматриваются вопросы применения моделей машинного обучения для решения задачи классификации сетевого трафика в различных постановках, подчеркивается значимость этапа сбора и подготовки обучающего набора данных в виду прямой зависимости качества обнаружения финальной модели от качества данных для обучения. Однако опубликованные результаты носят недостаточно полный и системный характер с точки зрения формализации требований к создаваемым наборам данных; практической реализации этапов сбора и

разметки; апробации моделей, обученных на собственных наборах данных; встраивания разрабатываемых программных модулей в действующие системы и комплексы и др.

Настоящая работа является логическим продолжением исследования [8], в котором по результатам апробации синтезированной модели обнаружения атак на реальных данных показана ее состоятельность только при условии обучения на данных, собранных в конкретной защищаемой сети, в виду зависимости ряда значимых признаков от физической структуры сети и настроек используемого оборудования. В качестве одного из основных выводов статьи отмечена необходимость предварительного обучения моделей на наборах данных, полученных на основе анализа сетевого трафика в защищаемой сети (аналоге с соответствующими характеристиками) и содержащем признаки классифицируемых компьютерных атак.

Решаемая в данном исследовании основная задача заключается в разработке практического подхода к формированию данных обучения для моделей обнаружения компьютерных атак, основанных на применении методов машинного обучения. Отдельной важной подзадачей при этом является апробация предлагаемых решений с целью оценки качества обучения модели на собранных данных и качества обнаружения атак в условиях реальной сетевой инфраструктуры.

3. Общедоступные наборы данных

Для обучения систем обнаружения компьютерных атак уровня сети, основанных на применении несигнатурных методов обнаружения, применяются специализированные наборы размеченных данных. К наиболее известным и опубликованным в открытых источниках наборам данных можно отнести следующие: DARPA1998, KDD Cup 1999, Kyoto 2006, NSL-KDD 2009, ISCX 2012, CTU-13, UNSW-NB15, CIDDS-001, UGR-16, CICIDS 2017, CICIDS 2018 и другие [3]. Данные наборы используются подавляющим большинством исследователей для апробации исследуемых алгоритмов обнаружения. Для описания наборов данных воспользуемся следующими характеристиками набора данных.

- Число признаков в наборе данных. В число признаков входят: признаки, характеризующие общую информацию о соединении / потоке (например, время начала соединения; IP адрес источника атаки; порт источника атаки и т.п.), информативные признаки (например, длительность соединения, число переданных / принятых байт и т.п.), а также признаки, которые используются для описания атаки или нормального сетевого соединения (например, метка класса трафика, описание атаки, реакция антивирусного средства на соединение).
- Природа информативных признаков. В табл. 1 используются следующие обозначения:
  ПСС – признаки, характеризующие сетевое соединение (например, длительность сетевойисессии);
  ПНП – признаки, характеризующие направление передачи данных (например, число байт переданных в направлении сервера; среднее время между сетевыми пакетами в направлениииклиента);
  ППУ – признаки, характеризующие операции, выполняемые на прикладном уровне (например, успех операции удаленной аутентификации пользователя, число операций с файлами в данном соединении и т.п.);
- Инструмент, который был использован для выделения признаков из сетевого трафика.
- Типы сетевых атак в наборе данных.

Табл. 1. Описание наборов данных, предназначенных для обучения несигнатурных систем обнаружения компьютерных атак уровня сети
Table 1. Description of datasets designed for training non-signature network intrusion detection systems

№	Набор данных	Число признаков в наборе данных	Природа информативных признаков	Инструмент для выделения признаков	Типы сетевых атак в наборе данных
1	DARPA 1998 [9]	10	ПСС	Нет сведений	DoS, Remote to User, User-to-Root, Surveillance/probing attacks
2	KDD Cup 1999 [10]	42	ПСС, ППУ	MADAM ID	DoS, Remote to User, User-to-Root, Surveillance/probing attacks
3	Kyoto 2006+ [11]	24	ПСС	Нет сведений	Различные атаки на honeypots (backscatter, DoS, exploits, malware, port scans, shellcode)
4	NSL-KDD 2009 (создан на основе KDD Cup 1999) [12]	42	ПСС, ППУ	MADAM ID	DoS, Remote to User, User-to-Root, Surveillance/probing attacks
5	ISCX 2012 [13]	19	ПСС, ПНП	Не известно	Brute Force SSH, HTTP DoS, DDoS using an IRC Botnet, Infiltrating the network from inside
6	CTU-13 [14]	33	ПСС, ПНП	Argus	botnets (Menti, Murlo, Neris, NSIS, Rbot, Sogou, Virut)
7	UNSW-NB15 [15]	45	ПСС, ПНП, ППУ	Argus, Bro-IDS	Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconaissance, Shellcode, Worms
8	CIDDS-001 [16]	14	ПСС, ПНП	NetFlow	Port scanning, DoS, BruteForce, Ping Scan
9	UGR-16 [17]	132 (Feature as Counter), 13 (в CSV файле)	ПСС, ПНП	FCParser	low- and high-rate DoS, Port scanning, UDP port scanning, SSH scanning, Botnet, Spam
10	CICIDS 2017 [18]	85	ПСП, ПСС, ПНП	CICFlowMeter	DoS Hulk, PortScan, DDoS, DoS GoldenEye, FTP-Patator, SSH-Patator, DoS slowloris, DoS Slowhttptest, Bot, Infiltration, Heartbleed, Web Attack – Brute Force, Web Attack – XSS, Web Attack – SQL Injection
11	CICIDS 2018 и другие наборы данных, созданные в	80	ПСП, ПСС, ПНП	CICFlowMeter-v3	Brute Force, Heartbleed, Botnet, DoS, DDoS, Web attacks, Infiltration of the network from inside

	Canadian Institute for Cybersecurity [19]				

Анализ публикаций с результатами оценки качества различных классификаторов сетевого трафика, обучение которых осуществлялось на представленных выше наборах данных, показал следующее [6 , 8]:

- для построения систем обнаружения компьютерных атак уровня сети использование признаков, характеризующих операции, которые выполняются на прикладном уровне, невозможно в силу того, что в настоящее время практически весь сетевой трафик является зашифрованным (используются протоколы TLS / SSL и IPSEC);
- в большинстве приведенных выше наборов данных используются признаки, характеризующие только сетевое соединение в целом (адресная информация, длительность соединения, число переданных байт), а также дополнительные признаки, полученные при анализе данных прикладного уровня;
- для улучшения качества классификаторов сетевого трафика необходимо использование признакового пространства, которое включает в себя множество признаков, характеризующих сетевое соединение в целом, каждое направление в отдельности, а также особенности конкретного соединения (потока) на транспортном уровне.

В настоящем исследовании за основу взяты признаки, представленные в наборах данных CICIDS 2017 и CICIDS 2018.

4. Инструменты выделения признаков сетевых сессий

Качество формируемого набора данных непосредственно зависит от качества инструментов, используемых на этапе сбора и анализа сетевого трафика, выделения признаков сетевых сессий. В ходе исследования публичных наборов данных отдельное внимание уделялось анализу используемых авторами инструментов.

Инструменты выделения признаков сетевых сессий обычно решают ряд задач и предоставляют следующие возможности.

1) Анализ сетевого трафика, выделение сетевых сессий (обычно применяются настраиваемые таймауты ожидания, активности и др. протокола TCP, от которых зависит момент логического завершения сетевой сессии инструментом).
2) Обработка сетевых сессий и выделение признаков.
3) Обработка трафика реального времени и предварительно сохраненного трафика (PCAP - файлы и др.).
4) Сохранение сформированных наборов данных в одном из форматов экспорта (CSV, XML, TXT и др.).

Наиболее распространенными инструментами анализа трафика и выделения признаков сетевых сессий являются следующие (табл. 2).

- Argus (Audit Record Generation and Utilization System) [20] – свободно распространяемый инструмент сетевого аудита, разработан одним из первых в своем классе. Позволяет обрабатывать сетевой трафик с выделением широкого спектра признаков сетевых сессий (всего 125), анонимизировать трафик, обогащать вектора признаков сессий дополнительными данными (например, данными о геолокации и др.). Поддерживает платформы Mac OS X, Linux, Unix, Windows; протоколы SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, Telnet, SSH, Gopher, NFS, DNS, Radius, IAX2, SIP, SunRPC, Whois, Rwhois, LPD, NTP; поддерживает IPv4 и IPv6. Предоставляет возможность расширения функционала с помощью пользовательских скриптов. Использовался при сборе наборов данных CTU-13 и UNSW-NB15.

- CICFlowMeter (первая версия называлась ISCXFlowMeter) [21] – свободно распространяемый анализатор сетевого трафика, который позволяет выделить сетевые сессии и для каждой сессии сформировать вектор признаков в формате CSV (всего 80). Разработан на языках программирования Java/C. Использовался при сборе наборов данных CICIDS 2017, CICAAGM 2017, CICAndMal 2017, CICIDS 2018, CICDDoS 2019 и др. Разработан в Canadian Institute for Cybersecurity.
- NFStream [22] – свободно распространяемый Python-фреймворк, предназначенный для высокопроизводительного анализа сетевого трафика. Расширяется с помощью системы плагинов, что позволяет добавлять функционал выделения новых признаков и встраивать модели машинного обучения в общий тракт обработки трафика.
- FCParser [23] – парсер потоков данных, реализующий методологию конструирования признаков «feature as a counter» (FaaC): каждый признак представляет собой счётчик числа наблюдений определённого события в заданный промежуток времени. Использовался при сборе набора данных UGR-16. Разработан на языке Python.
- MADAM ID (Mining Audit Data for Automated Models for Intrusion Detection) – сетевая система обнаружения атак, использующая интеллектуальный анализ данных для обнаружения аномалий. Устаревший инструмент, использовался при сборе одного из первых наборов данных – KDD Cup 1999.

Для проверки корректности разметки набора данных CICIDS 2017 авторами настоящего исследования были воспроизведены эксперименты [2, 24**]**. По результатам обработки исходных PCAP-файлов с захваченными пакетами набора данных CICIDS 2017 собственным инструментом выделения признаков сетевых сессий были обнаружены расхождения полученных данных и данных набора CICIDS 2017. Дальнейшие исследования показали наличие следующих ошибок в исходном коде инструмента CICFlowMeter, который использовался при сборе и формировании набора данных CICIDS 2017.

1) Ошибки при расчетах значений признаков «Packet Length Mean» (средняя длина полезной нагрузки в пакетах всего потока), «Packet Length Std», «Packet Length Variance» и «Average Packet Size». Ошибка связана с двойным учетом первого пакета в структуре данных со статистикой длин пакетов сетевой сессии.
2) Некорректное завершение сессий – при первом появлении в сессии пакета с флагом FIN. Возможные следующие пакеты с флагами FIN ACK и ACK, фактически относящиеся к первой незавершенной сессии, попадают во вторую сессию. Это приводит к появлению в наборе данных большого количества сессий, состоящих из одного-двух пакетов, с нулевой длиной полезной нагрузки.
3) Дублирование признака «Fwd Header Length».
4) Ошибка при расчете длины TCP пакета – длина дополнения (padding) фрейма Ethernet прибавляется к длине TCP пакета.
5) Признаки «Packet Length Mean» и «Average Packet Size» должны иметь одинаковое значение, однако по причине логической ошибки имеют различные значения. Ошибка состоит в том, что при завершении сессии по таймеру граничный пакет попадает в статистику длин пакетов, а счетчик количества пакетов (знаменатель в выражении для расчета значения признаков «Packet Length Mean» и «Average Packet Size») увеличивается только для одного из признаков.

Отчет об ошибках отправлен авторам набора данных в марте 2021 года, в том числе в виде issue в репозиторий с исходным кодом инструмента CICFlowMeter (https://github.com/ahlashkari/CICFlowMeter/issues/111), однако на момент подготовки публикации настоящего исследования (ноябрь 2021 года) ошибки не исправлены.

Указанные обстоятельства в отношении одного из наиболее цитируемых в мире наборов данных подтверждают необходимость как обязательной верификации используемых данных

для обучения моделей машинного обучения, так и предъявления соответствующего требования к создаваемым общедоступным наборам данных: возможность верификации публикуемых данных.

Наличие ошибок в общедоступных инструментах выделения признаков является причиной возможных скрытых ошибок в создаваемых с их помощью наборах данных. В таких условиях оправданными являются разработка своих собственных инструментов с последующим сравнением результатов их работы на публичных наборах данных с результатами общедоступных инструментов.

Табл. 2. Общедоступные инструменты выделения признаков сетевых сессий
Table 2. Public generators and analyzers of network traffic flows

№	Название инструмента, лицензия	Поддерживаемые платформы	Язык программирования	Количество выделяемых признаков	Наборы данных, сформированные с использованием инструмента
1	Argus, GPLv2	Linux, Solaris, BSD, OS X, IRIX, AIX, Windows, OpenWrt	C	125	CTU-13, UNSW-NB15
2	CICFlowMeter, производная от MIT	Сведения отсутствуют	Java/C	80	CICIDS 2017, CICAAGM 2017, CICAndMal 2017, CICIDS 2018, CICDDoS 2019 и др.
3	NFStreams, LGPL-3.0	Linux, MacOS, ARM	Python	48	Сведения отсутствуют
4	FCParser, сведения отсутствуют, исходный код открыт	Unix	Python	Переменное количество (методология FaaC)	UGR-16
5	MADAM ID, сведения отсутствуют	Сведения отсутствуют	Сведения отсутствуют	Сведения отсутствуют	KDD Cup 1999

5. Методика сбора обучающего набора данных

5.1 Требования к наборам данных

Большинство публичных наборов данных для обучения систем обнаружения компьютерных атак были разработаны с главной целью – предоставить исследователям возможность сравнения различных методов обнаружения в одинаковых условиях. На практике часто возникает и другая задача: оценить качество синтезированной модели машинного обучения на нескольких наборах данных. И в первом, и во втором случае исследователю предстоит обосновать выбор используемых при обучении данных. При сравнении характеристик различных наборов данных важным является вопрос формализации единых требований к ним.

Основополагающими требованиями к публикуемым наборам данных можно считать перечисленные в исследовании [25]:

- возможность однозначной идентификации – набор данных должен быть уникальным, содержать подробное описание, быть проиндексированным в соответствующих поисковых системах;
- доступность – должен быть предоставлен свободный доступ к набору данных по его

идентификатору;

- возможность сравнения метаданных – наборы данных должны использовать единые словари метаданных;
- многократное использование – данные должны быть точно описаны совокупностью релевантных атрибутов и соответствовать отраслевым стандартам, должны быть указаны происхождение данных и лицензионные условия их использования.

В статье [26] рассматриваются следующие требования к синтетическим наборам данных для обучения моделей обнаружения атак.

- Полная конфигурация сети. В моделируемой сети должны быть представлены различные устройства: модемы, брандмауэры, коммутаторы, маршрутизаторы, с различными операционными системами.
- Полный трафик. Набор данных должен включать и «чистый» трафик, и трафик компьютерных атак.
- Наличие разметки.
- Полное взаимодействие. При сборе набора данных необходимо моделировать взаимодействие внутри конкретной локальной сети, между несколькими локальными сетями и связь через Интернет.
- Полный захват трафика и сохранение.
- Разнообразие сетевых протоколов. Набор должен включать данные взаимодействия по различным протоколам: HTTP, HTTPS, FTP, SSH, протоколам электронной почты и др.
- Разнообразие атак. При сборе трафика атак должны моделироваться наиболее распространенные атаки, такие как веб-атаки, bruteforce, DoS, DDoS, попытки проникновения, активность ботнетов, сканирование и др.
- Разнородность анализируемых данных. Должен осуществляться захват сетевого трафика и анализ дампа памяти и системных вызовов со всех машин-жертв во время выполнения атак.
- Представление данных. Собранные и размеченные данные следует публиковать в одном из общепринятых форматов представления данных.

В работе [3] предложены дополнительные требования к создаваемым наборам данных для обучения моделей обнаружения компьютерных атак: актуальности и разнородности представленных атак, наличия «чистого» пользовательского трафика и в нём – трафика полезной нагрузки, корректности разметки. Кроме того, при создании наборов данных должен охватываться значительный временной интервал для включения в набор достаточного количества записей, соответствующих атакам и «чистому» трафику. Авторы отмечают, что в метаданных набора должны указываться сведения о его разбиении на отдельные логические блоки данных (если есть), а также сведения о балансе классов объектов выборки.

В условиях наличия возможных скрытых ошибок в разметке данных необходимо предъявлять дополнительное требование к публикуемым наборам данных: возможность их верификации.

5.2 Выбор объекта защиты и класса атак

Качество модели выявления атак напрямую зависит от качества обучающего набора данных, которое тем выше, чем ближе моделируемый объект к реальному. В роли объекта защиты может рассматриваться как корпоративная сеть в целом, так и ее отдельные сервисы. Вместе с тем правильным выглядит подход, при котором в качестве объекта защиты рассматриваются именно конкретные продукты информационных технологий,

функционирующие в исследуемой сети. Например, в роли объектов защиты могут выступать SSH, FTP, TELNET и другие сетевые службы, веб-приложения, включая веб-консоли управления различных устройств сети, служба удаленных рабочих столов и т. д. Сбор обучающего набора данных для выделенных объектов защиты должен осуществляться в реальной сети, в которой они функционируют, или (при отсутствии такой возможности) на близком по структуре, составу и настройкам сетевых устройств макете сети. При этом для каждого объекта защиты перечень реализуемых в отношении них классов атак будет индивидуальным и определяется исходя из его особенностей. Также необходимо отметить, что некоторые классы атак будут характерны только для определенных типов объектов защиты, например, класс веб-атак характерен только для такого типа объектов защиты как веб-приложения.

В представленном исследовании вопрос формирования обучающего набора данных рассматривался на примере класса веб-атак.

5.3 Особенности генерации веб-атак в наборе данных CICIDS 2017

В ходе анализа информации об особенностях генерации веб-атак для набора данных CICIDS 2017 [2] было установлено, что в качестве атакуемого приложения в стенде использовался проект Damn Vulnerable Web Application (DVWA) – «чертовски» уязвимое веб-приложение, в отношении которого моделировалось три вида атак. Информация о количестве веб-атак каждого вида, продолжительности их генерации и используемых для этого средствах представлена в табл. 3.

Табл. 3. Информация о веб-атаках в CICIDS 2017
Table 3. Number of attacks in the CICIDS 2017 dataset

№	Тип атаки	Количество атак	Продолжительность генерации атак	Средство генерации
1	Web Attack – Brute Force	1507	40 мин.	Patator
2	Web Attack – XSS	652	20 мин.	Selenium
3	Web Attack – SQL Injection	21	2 мин.	Selenium

Итоговый набор данных CICIDS 2017 содержит 2180 записей, касающихся веб-атак, что составляет 0,07% от всего датасета. При этом продолжительность генерации атак говорит о том, что в ходе этого процесса не учитывались возможные временные параметры моделирования атак, т.е. задержки, а также другие возможные настройки средств генерации. Кроме того, было установлено, что при моделировании веб-атак использовались два субъекта – источник атаки (205.174.165.73/172.16.0.1) и атакуемое веб-приложение (192.168.10.50/205.174.165.68). То есть в данном датасете моделировалась ограниченная схема реализации XSS-атак (без участия третьей стороны). В сведениях о наборе данных не указано, по какому протоколу осуществлялся доступ к веб-приложению (только на основе анализа PCAP файла можно установить, что использовался протокол HTTP). Также отсутствует информация о типе и настройках используемого веб-сервера.

Проведенные исследования показали, что параметры сетевых соединений при работе с веб-приложениями существенно зависят от типа протокола (HTTP/HTTPS) и от типа веб-сервера (например, Apache/Nginx) и его настроек (например, параметр timeout). В частности, в ходе экспериментов установлено следующее:

- модели, обученные на HTTP трафике, не способны качественно работать на HTTPS трафике и наоборот;
- модели, обученные на трафике с веб-сервером Apache (KeepAliveTimeout 5), не способны качественно работать на трафике с веб-сервером Nginx (keepalive_timeout 65) и наоборот.

Также в ходе проведенных экспериментов было установлено, что при реализации веб-атак с использованием браузера в сетевом потоке будут содержаться как сессии, которые содержат элементы атаки, например, GET/POST-запросы со зловредной нагрузкой, так и сессии, которые относятся к фоновому трафику, например, штатная процедура авторизации или переходы по ссылкам. В этой связи такой трафик должен подвергаться тщательному анализу и разметке. Данная процедура не является сложной и может быть реализована с использованием механизма регулярных выражений. Вместе с тем для HTTPS трафика такая процедура потребует проведения расшифровки трафика, для чего необходимо наличие соответствующих SSL ключей. Такой вопрос в CICIDS 2017 не учитывался.

Фоновый трафик в CICIDS 2017 моделировался на основе использования статистических профилей работы пользователей (B-Profile) [27].

На основе проведенного анализа были сформулированы следующие предложения по улучшению качества формируемого обучающего набора данных для выявления веб-атак:

- расширить перечень атак и используемых средств генерации трафика;
- использовать при моделировании трафика возможности применения временных задержек и других параметров (опций) средств генерации;
- осуществлять разметку трафика, явно содержащего как атаки, так и элементы легитимных действий пользователей;
- расширить объем набора данных;
- учитывать тип используемого протокола реального объекта;
- учитывать тип и настройки веб-сервера реального объекта.

5.4 Испытательный стенд для моделирования веб-атак

В качестве тестового веб-приложения для моделирования атак и фонового трафика было принято решение об использовании DVWA. Для моделирования был определен список из шести видов веб-атак:

- Cross-Site Scripting (XSS);
- Cross Site Request Forgery (CSRF);
- SQL Injection (SQLi);
- Upload a Web Shell to a Web Server (Malware);
- Password Brute Forcing (Brute);
- OS Command Injection (COMMi).

Далее был проведен анализ программных средств, которые могут использоваться для реализации этих атак, а также генерации фонового трафика, и выбраны наиболее подходящие (табл . 4).

Табл. 4. Базовые инструментальные средства моделирования веб-трафика
Table 4. Basic web traffic modeling tools

Тип веб-трафика	Базовые инструментальные средства моделирования
Фоновый трафик	Браузер; Selenium IDE;
Атака: Cross-Site Scripting (XSS)	Браузер; Selenium IDE; Hackapp; Xsser
Атака: Cross Site Request Forgery (CSRF)	Браузер; Selenium IDE; Hackapp
Атака: SQL Injection	Браузер; Selenium IDE; Sqlmap
Атака: Upload a Web Shell to a Web Server	Браузер; Selenium IDE; Weevely
Атака: Password Brute Forcing	Patator
Атака: OS Command Injection	Браузер; Selenium IDE; Commix

Итоговый список используемых для генерации обучающего трафика программных средств включает следующие: Браузер, Selenium IDE, Xsser, Hackapp, Sqlmap, Weevely, Patator, Commix. Все перечисленные средства, за исключением Hackapp, входят в состав дистрибутива ОС Kali Linux. Hackapp – это веб-приложение, которое было специально разработано для моделирования XSS (Reflected) и CSRF атак. Оно включает страницы со скрытыми формами, при посещении которых отправляются запросы от имени авторизованного пользователя атакуемого веб-приложения и тем самым осуществляются атаки соответствующих классов. В итоге, в состав стенда моделирования веб-атак были включены три компонента (рис. 1):

- Attaker – хост генерации и реализации атак;
- Web – хост, на котором установлено атакуемое веб-приложение (DVWA);
- Victim – хост пользователя, в отношении которого реализуются XSS (Reflected) и CSRF атаки.

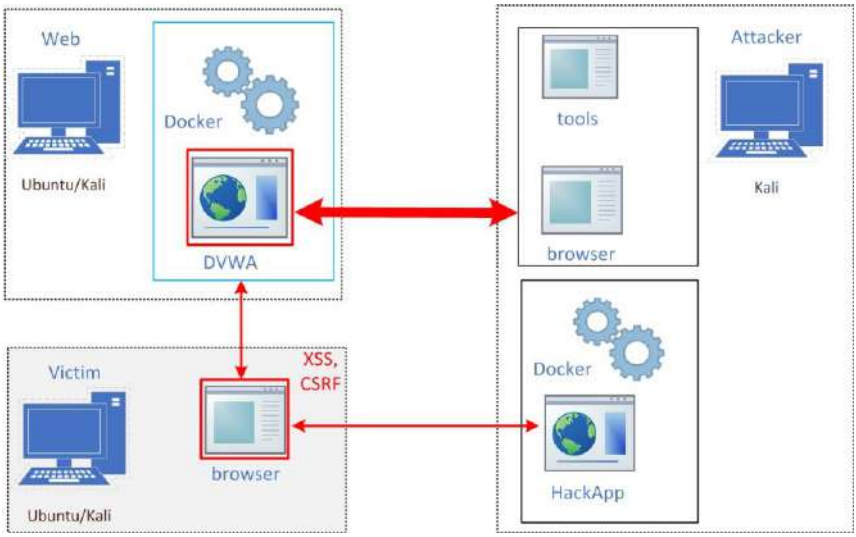


Рис. 1. Обобщенная схема моделирования веб-атак и фонового трафика
Fig. 1. A generalized modeling scheme for web attacks and background traffic

Для ускорения процесса развертывания веб-приложений использовалась технология контейнеризации Docker.

5.4 Сценарии генерации фонового веб-трафика и атак

С учетом сформированного перечня инструментальных средств, анализа их возможностей были подготовлены сценарии генерации десяти типов трафика (табл. 5).

Фоновый трафик, а также некоторые виды атак моделировались с использованием плагина для браузера Selenium IDE (SeIDE), позволяющего создавать и воспроизводить тесты, содержащие последовательность действий пользователя в браузере. Общий алгоритм моделирования трафика на основе SeIDE состоит в следующем.

1) Создается базовый тест моделирования действий пользователя.

2) Базовый тест тиражируется с одновременной вставкой секций пауз между секциями операций теста. Паузы заполняются случайным образом, а при необходимости конкретизации вводимых пользователем данных используется заранее подготовленный пул полезной нагрузки (например, для моделирования атак XSS, SQLi, COMMi).

3) Осуществляется последовательный запуск всех тиражированных тестов. При этом браузер для воспроизведения очередного теста выбирается случайным образом (в исследовании использовались браузеры Chrome и Firefox).

Табл. 5. Типы генерируемого веб-трафика
Table 5. Types of generated web traffic

Класс трафика		**Обозначение**	**Средство генерации**	**Тип трафика**
Benign	Фоновый трафик	Benign	SeIDE	Трафик на основе тестов SeIDE, осуществляющих выполнение в DVWA общих действий, не содержащих атак
		Benignxss	SeIDE	Трафик на основе тестов SeIDE, осуществляющих выполнение на уязвимой странице DVWA, содержащей уязвимость XSS, действий, не содержащих атак
		Benignsqli	SeIDE	Трафик на основе тестов SeIDE, осуществляющих выполнение на уязвимой странице DVWA, содержащей уязвимость SQLi, действий, не содержащих атак
Attack	Cross-Site Scripting (XSS)	XSS	Xsser	Трафик, генерируемый с использованием утилиты Xsser, осуществляющей поиск XSS-уязвимостей в DVWA
			SeIDE	Трафик на основе тестов SeIDE, осуществляющих реализацию атак (внедрение зловредной полезной нагрузки) на уязвимой странице DVWA, содержащей уязвимость XSS
			HackApp	Трафик на основе тестов SeIDE, осуществляющих посещение недоверенного сайта (приложение HackApp), содержащего зловредный код, реализующий XSS-атаки в отношении DVWA
	SQL Injection	SQLi	Sqlmap	Трафик, генерируемый с использованием утилиты Sqlmap, осуществляющей поиск SQLi-уязвимостей в DVWA
			SeIDE	Трафик на основе тестов SeIDE, осуществляющих реализацию атак (внедрение зловредной полезной нагрузки) на уязвимой странице DVWA, содержащей уязвимость SQLi
	Password Brute Forcing	Brute	Patator	Трафик, генерируемый с использованием утилиты Patator, осуществляющей подбор пароля DVWA
	Cross Site Request Forgery (CSRF)	CSRF	HackApp	Трафик на основе тестов SeIDE, осуществляющих посещение недоверенного сайта (приложение HackApp), содержащего зловредный код, реализующий CSRF-атаки в отношении DVWA
	Upload a Web Shell to a Web Server	Malware	Weevely	Трафик, генерируемый с использованием утилиты Weevely, устанавливающей соединение с внедренным в DVWA Shell-кодом (BackDoor.php).

	OS Command Injection	COMMi	Commix	Трафик, генерируемый с использованием утилиты Commix, осуществляющей эксплуатацю уязвимости OS Command Injection в DVWA
			SeIDE	Трафик на основе тестов SeIDE, осуществляющих реализацию атак (внедрение зловредной полезной нагрузки) на уязвимой странице DVWA, содержащей уязвимость OS Command Injection

Для успешной работы специализированных программных средств генерации веб-атак требуется знание авторизационной cookie (для DVWA это PHPSESSID), которая может быть извлечена из хранилища браузера или из информационного обмена с веб-приложением. В представленном исследовании cookie PHPSESSID извлекалась по второму варианту и передавалась в качестве одного из входных параметров для работы скриптов генерации, запускающих соответствующие программные средства (Sqlmap, Xsser, Patator, Weevwly, Commix). Кроме того, при генерации атак применялись соответствующие опции данных программ, определяющие значения задержек, потоков и других параметров, позволяющих вариативно реализовывать зловредные действия.

5.5 Комплекс скриптов генерации веб-трафика и формирования на его основе обучающего набора данных

Для написания всех скриптов использовался язык Bash.

В процессе проведения исследования были выработаны требования к модулям генерации фонового веб-трафика и трафика веб-атак, позволяющие автоматизировать данный процесс.

1) Скрипты генерации фонового веб-трафика и трафика веб-атак должны позволять:
   - генерировать отдельный трафик по выбору;
   - генерировать весь обучающий трафик сразу.
2) Трафик должен генерироваться одним скриптом (TrafficGen.sh), в котором будут определены все задачи по формированию обучающего трафика.
3) Выполнение задач по генерации конкретного вида трафика осуществляется с использованием разработанных скриптов и тестов генерации веб-атак и фонового веб-трафика, запуск которых прописывается в основном скрипте (TrafficGen.sh).
4) Скрипты генерации трафика должны использовать параметры командной строки, позволяющие передавать в них все необходимые для их работы параметры, что обеспечит отсутствие необходимости внесения изменений в сами скрипты.
5) Для скриптов генерации трафика должна быть предусмотрена возможность определения параметров или их части в конфигурационных файлах.

В результате был разработан комплекс скриптов, позволяющих полностью автоматизировать процесс генерации, сбора и обработки заданного обучающего веб-трафика. Обобщенная схема работы комплекса представлена на рис. 2.

Перед запуском основного модуля генерации трафика TrafficGen.sh необходимо:

- определить значения общих параметров, требуемых для работы скриптов (IP адреса, пути к каталогам с ресурсами, настройки SSH соединений, опции выполняемых действий и т.д.), которые прописываются в едином конфигурационном файле TrafficGen.conf;
- уточнить в основном модуле генерации трафика TrafficGen.sh параметры запускаемых вспомогательных модулей (осуществляют подготовительные действия, заключающиеся в настройке приложения Hackapp, выделении PHPSESSID, тиражировании тестов и вставке в них пауз) и модулей генерации трафика (выполняют основной функционал по формированию обучающего набора данных).

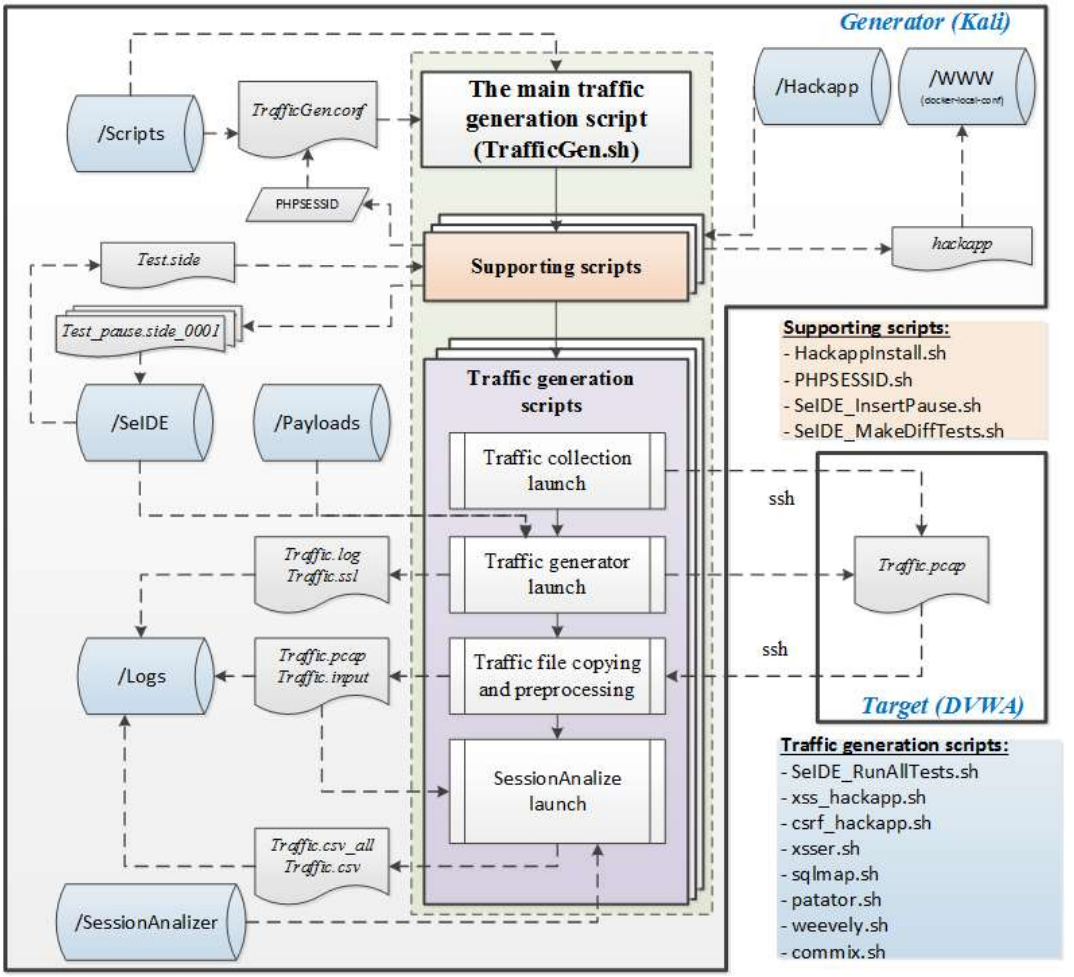


Рис. 2. Обобщенная схема работы комплекса скриптов генерации веб-трафика и формирования на его основе обучающего набора данных

Fig. 2. Web traffic generation scripts and training dataset generation

Общий алгоритм работы модулей генерации трафика состоит в следующем (рис. 2):

1) Осуществляется SSH-подключение к мишени, где запускается процедура сбора трафика с помощью программного средства Tshark. Для минимизации объема собираемых данных регистрация трафика может выполняться на интерфейсе docker0.
2) Осуществляется запуск генератора трафика (запуск соответствующего программного средства, которое моделирует трафик определенного вида). При этом ведется лог работы, в котором регистрируется выполняемое действие и время его начала и завершения. Кроме того, для HTTPS трафика в хранилище сохраняются SSL-ключи. По окончании работы генератора сбор трафика прекращается.
3) Выполняется копирование PCAP-файла трафика в центральное хранилище и его предобработка. В частности, для трафика, содержащего веб-атаки с элементами легитимных действий пользователей, выделяются номера пакетов, в которых содержится зловредная полезная нагрузка (для HTTPS трафика перед этим с помощью программного средства Tshark выполняется процедура расшифровки на основе сохраненных ранее SSL-ключей).

4) Осуществляется обработка PCAP -файла трафика на предмет выделения сессий и расчета их параметров с помощью разработанного в ходе исследования программного модуля SessionAnalizer. В случае трафика, содержащего веб-атаки с элементами легитимных действий пользователей, на вход данного модуля также подается файл с номерами пакетов, содержащих зловредную полезную нагрузку, что позволяет пометить сессии, содержащие данные пакеты. Результатом работы модуля SessionAnalizer является файл в формате CSV, где каждая строка соответствует выделенной сессии, а столбец соответствует параметру сессии (признаку). За основу взято признаковое пространство набора данных CICIDS 2017. При этом в обучающий набор данных включается только тот трафик, который генерируется с IP адреса хоста-генератора на заданный порт (80/443), по заданному протоколу (HTTP/HTTPS), а также содержащий интересующие пакеты. Последним столбцом результирующего файла включается метка, которая определяет принадлежность сессии к фоновому трафику или к одной из моделируемых веб-атак. Данная метка предопределена заранее и соответствует типу трафика, который моделируется скриптом. Результатом работы каждого модуля генерации трафика будет являться фрагмент обучающего набора данных, принадлежащий только одному классу.

5.6 Развертывание испытательного стенда и сбор данных

Как отмечалось выше, сбор данных при моделировании атак и фонового трафика для выбранного объекта защиты должен осуществляться в реальной сети. В этой связи было осуществлено развертывание элементов предложенного испытательного стенда в реальной инфраструктуре в виде соответствующих виртуальных машин:

- Web – виртуальная машина с операционной системой Ubuntu 20.04 с развернутыми в виде docker-контейнеров мишенями DVWA, использующими разные типы веб-серверов (DVWA_apache и DVWA_nginx);
- Attacker, Victim – виртуальная машина с операционной системой Kali Linux с установленными средствами генерации (моделирования) веб-трафика (Chrome, FireFox, Selenium IDE, Xsser, HackApp, Sqlmap, Weevely, Patator, Commix).

Результаты сбора обучающего набора данных для протокола HTTPS, полученные на развернутом испытательном стенде, представлены в табл. 6.

Табл. 6. Результаты сбора обучающего набора данных для протокола https
Table 6. The results of collecting the training dataset for the https protocol

Класс	Тип трафика	Обозначение	Инструмент	Кол-во	Всего по типу трафика	Всего по классам	Кол-во	Всего по типу трафика	Всего по классам
				Apache			Nginx		
Benign	Фоновый веб-трафик	benign	SelDE	19901	29807	29807	4752	8498	8498
		benignsqli	SelDE	5023			1861		
		benignxss	SelDE	4883			1885		
Web Attack	CAPEC-63: Cross-Site Scripting (XSS)	xss	SelDE	1286	13925	28223	944	11353	36027
			xsser	11637			9022		
			hackapp	1002			1387		
	CAPEC-62: Cross Site	csrf	hackapp	931	931		1422	1422	

	Request Forgery								
	CAPEC-66: SQL Injection	sqli	SeIDE	537	1455		521	1431	
			sqlmap	918			910		
	CAPEC-49: Password Brute Forcing	brute	patator	2693	2693		49	49	
	CAPEC-650: Upload a Web Shell to a Web Server	malware	weevely	6452	6452		6748	6748	
	CAPEC-88: OS Command Injection	commi	SeIDE	101	2767		100	15024	
			commix	2666			14924		
Итого:						58030			44525

6. Тестирование и апробация методики

На основе сформированного набора данных была синтезирована модель выявления веб-атак, в основе которой использовался хорошо зарекомендовавший себя классификатор RandomForestClassifier. При построении модели осуществлялся подбор гиперпараметров модели.

Модель строилась в предположении того, что веб-приложение функционирует по протоколу HTTPS на веб-сервере Apache, т.е. для ее обучения использовался соответствующий набор данных (табл. 6). Полученная модель (табл. 7, эксперимент № 4) по метрикам качества превосходит модели, представленные в [8], которые использовали при обучении набор данных CICIDS 2017 (табл. 7, эксперимент № 1-2), а также набор данных, собранный на реальной сети, но в ограниченных условиях (табл. 7, эксперимент № 3).

Табл. 7. Протокол эксперимента
Tabl. 7. Experiment protocol

Эксперимент / Характеристика	**Эксперимент №1**	**Эксперимент №2**	**Эксперимент №3**	**Эксперимент №4**
Этап обучения модели				
Используемый набор данных	Сбалансированная и предобработанная подвыборка веб-атак WebAttacks набора данных CICIDS 2017. 7267 записей, из них 5087 экземпляров класса «нет атаки» и 2180 экземпляров класса «есть атака».		Сформированный набор данных, соответствующих реальному сетевому трафику	Сформированный набор данных, соответствующих реальному сетевому трафику
Обучающая выборка	70% записей используемого набора данных		70% записей используемого набора данных	70% записей используемого набора данных

Модель машинного обучения	RandomForestClassifier (max_depth=17, max_features=10, min_samples_leaf=3, n_estimators=50)		RandomForestClassifier (max_depth=None, max_features='auto', min_samples_leaf=1, n_estimators=250)	RandomForestClassifier (max_depth=17, max_features=10, min_samples_leaf=3, n_estimators=50)
Наиболее значимые признаки	1. Average Packet Size 2. Flow Bytes/s 3. Max Packet Length 4. Fwd Packet Length Mean 5. Fwd IAT Min 6. Total Length of Fwd Packets 7. Fwd IAT Std 8. Flow IAT Mean 9. Fwd Packet Length Max 10. Fwd Header Length		Flow Packets/s Flow IAT Max Bwd Packet Length Min Flow Duration Flow IAT Mean Flow IAT Std Average Packet Size Fwd Packet Length Max Total Packets Fwd Header Length	Fwd IAT Total Fwd Packet Length Max Fwd Packet Length Std Bwd IAT Total Bwd Packet Length Std Total Length of Fwd Packets Flow IAT Max Bwd Packet Length Max Bwd Packet Length Mean Total Length of Bwd Packets
Этап тестирования модели				
Тестовая выборка	30% записей используемого набора данных. Тестовая и обучающая выборка не имеют пересечений	100% записей сформированного набора данных, соответствующих реальному сетевому трафику	30% записей используемого набора данных. Тестовая и обучающая выборка не имеют пересечений.	30% записей используемого набора данных. Тестовая и обучающая выборка не имеют пересечений
Значения метрик качества классификации				
Accuracy	0.983	0.456	0.858	0.983
Precision	0.982	0.812	0.812	0.998
Recall	0.961	0.033	0.966	0.961
F1	0.971	0.064	0.882	0.979

Полученные результаты свидетельствуют о необходимости обучения модели обнаружения атак на наборе данных, собранном на основе анализа сетевого трафика в отношении конкретного защищаемого объекта реальной сети. При этом перед процедурой формирования обучающего набора данных должна быть проведена всесторонняя оценка объекта защиты. По ее результатам должны быть определены:

- условия, в которых функционирует объект защиты;
- особенности конфигурации объекта защиты;
- перечень актуальных для объекта защиты атак;
- перечень инструментальных средств, используемых для моделирования актуальных атак в отношении объекта защиты;
- перечень инструментальных средств, используемых для моделирования фонового трафика объекта защиты;
- мишень, в качестве которой может выступать сам объект защиты или его аналог;
- генератор(ы), с помощью которых осуществляется моделирование фонового трафика и атак;
- типы генерируемого трафика и порядок его разметки;

- структура испытательного стенда, позволяющая моделировать фоновый трафик и все атаки из сформированного перечня (все типы трафика), а также обеспечивающая возможность развертывания стенда в реальной инфраструктуре;
- пути автоматизации процедуры формирования обучающего набора данных.

7. Заключение

В предшествующем исследовании [8] для оценки применимости методов машинного обучения в системах обнаружения компьютерных атак был проведен эксперимент с настройкой модели «случайный лес», обучением на публичном наборе данных CICIDS 2017 и тестированием в реальных условиях. Настройка параметров выбранного классификатора позволила на валидационной выборке получить оценку F1-меры 0.971 для набора данных CICIDS 2017. При этом была подчеркнута невозможность применения предобученной модели на тестовой выборке, полученной на основе анализа сетевого трафика в реальной компьютерной сети (F1-мера 0.064, неудовлетворительное качество). Для получения удовлетворительного качества обнаружения потребовалось выполнить переобучение модели на сформированном собственном наборе данных, что позволило получить значение F1-меры 0.882.

В настоящем исследовании предложена методика сбора такого обучающего набора данных, позволяющая синтезировать адекватную модель обнаружения компьютерных атак в отношении заранее известного объекта защиты. Основа методики заключается в том, что сбор обучающего набора данных ведется в реальной сети в отношении конкретного объекта защиты или его модели, максимально учитывающей все его характеристики. При этом методика предполагает использование при моделировании трафика широкого круга программных средств, запускаемых с применением временных задержек и других параметров (опций), позволяющих вариативно реализовывать негативные действия, а также требует осуществлять разметку трафика, явно содержащего как атаки, так и элементы легитимных действий пользователей, в том числе и при использовании протокола HTTPS. Это позволяет повысить точность выявления атак: результаты апробации предлагаемых решений демонстрируют возможности дальнейшего повышения качества обнаружения атак, на тестовой выборке получено значение F1-меры 0.979 (точность значения F1-меры ограничивается, в том числе, объемом выборки).

Для поддержания набора данных в актуальном состоянии необходимо включать в него данные по всем существующим на текущий момент актуальным генераторам атак и постоянно дополнять по мере появления новых средств реализации атак.

Кроме того, построенная модель обнаружения компьютерных атак должна дообучаться по мере расширения набора данных, а также апробироваться на атаках, реализуемых из разных точек расположения сетевой инфраструктуры.

Направлениями дальнейших исследований являются: возможная автоматизация процедуры обучения для новых объектов защиты; глубокий анализ признакового пространства с целью определения признаков, независящих от физической структуры сети, настроек используемого оборудования, используемых программных средств; всесторонняя оценка полученных результатов в сравнении с существующими средствами защиты информации.

Список литературы / References

[1]. Sarker I.H., Furhad M.H., Nowrozy R. AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions. SN Computer Science, vol. 2, issue 3, 2021, article no: 173.

[2]. Sharafaldin I., Lashkari A.H., Ghorbani Ali A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proc. of the 4th International Conference on Information Systems Security and Privacy (ICISSP), 2018, pp. 108-116.

[3]. Ring M., Wunderlich S. et al. Computers & Security, vol. 86, 2019, pp. 147-167.

[4]. Гетьман А.И., Иконникова М.К. Обзор методов классификации сетевого трафика с использованием машинного обучения. Труды ИСП РАН, том 32, вып. 6, 2020 г., стр. 137-154 / Getman A.I., Ikonnikova M.K. A survey of network traffic classification methods using machine learning. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 6, 2020, pp. 137-154 (in Russian). DOI: 10.15514/ISPRAS–2020–32(6)–11.
[5]. Khatouni A.S., Heywood N.Z. How much training data is enough to move a ML-based classifier to a different network? Procedia Computer Science, vol. 155, 2019, pp. 378-385.
[6]. Ghurab M., Gaphari G. et al. A Detailed Analysis of Benchmark Datasets for Network Intrusion Detection System. Asian Journal of Research in Computer Science, 2021, vol. 7, issue 4, pp. 14-33.
[7]. Magán-Carrión R., Urda D. et al. Towards a Reliable Comparison and Evaluation of Network Intrusion Detection Systems Based on Machine Learning Approaches. Applied Sciences, 2020, vol. 10, issue 5.
[8]. Горюнов М.Н., Мацкевич А.Г., Рыболовлев Д.А. Синтез модели машинного обучения для обнаружения компьютерных атак на основе набора данных CICIDS2017. Труды ИСП РАН, том 32, вып. 5, 2020 г., стр. 81-94 / Goryunov M.N., Matskevich A.G., Rybolovlev D.A. Synthesis of a machine learning model for detecting computer attacks based on the CICIDS2017 dataset. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2020, pp. 81-94 (in Russian). DOI: 10.15514/ISPRAS–2020–32(5)–6.
[9]. 1998 DARPA Intrusion Detection Evaluation Dataset. URL: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset, accessed 24.10.2021.
[10]. KDD Cup 1999 Data. URL: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, accessed 24.10.2021.
[11]. Traffic Data from Kyoto University's Honeypots. URL: http://www.takakura.com/Kyoto_data/, accessed 24.10.2021.
[12]. NSL-KDD dataset. URL: https://www.unb.ca/cic/datasets/nsl.html, accessed 24.10.2021.
[13]. Intrusion detection evaluation dataset (ISCXIDS2012). URL: https://www.unb.ca/cic/datasets/ids.html, accessed 24.10.2021.
[14]. CTU-13 Dataset. URL: https://mcfp.felk.cvut.cz/publicDatasets/CTU-13-Dataset/, accessed 24.10.2021.
[15]. UNSW-NB15 Dataset. URL: https://ieee-dataport.org/documents/unswnb15-dataset#files, accessed 24.10.2021.
[16]. CIDDS-001 Coburg Intrusion Detection Data Set. URL: https://www.hs-coburg.de/fileadmin/hscoburg/WISENT-CIDDS-001.zip, accessed 24.10.2021.
[17]. UGR'16: A New Dataset for the Evaluation of Cyclostationarity-Based Network IDSs. URL: https://nesg.ugr.es/nesg-ugr16/index.php, accessed 24.10.2021.
[18]. Intrusion Detection Evaluation Dataset (CIC-IDS2017). URL: https://www.unb.ca/cic/datasets/ids-2017.html, accessed 24.10.2021.
[19]. Canadian Institute for Cybersecurity datasets. URL: https://www.unb.ca/cic/datasets/index.html, accessed 24.10.2021.
[20]. Argus. URL: https://openargus.org/, accessed 24.10.2021.
[21]. CICFlowMeter. URL: https://github.com/CanadianInstituteForCybersecurity/CICFlowMeter, accessed 24.10.2021.
[22]. NFStream: a Flexible Network Data Analysis Framework. URL: https://github.com/nfstream/nfstream, accessed 24.10.2021.
[23]. FCParser: Feature as a Counter Parser for Networkmetrics. URL: https://github.com/josecamachop/FCParser, accessed 24.10.2021.
[24]. Kostas K. Anomaly Detection in Networks Using Machine Learning. Master's Thesis. University of Essex, 2018, 70 p.
[25]. Wilkinson M., Dumontier M. et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data, vol. 3, 2016, article number 160018.
[26]. Gharib A., Sharafaldin I. et al. An Evaluation Framework for Intrusion Detection Dataset. In Proc. of the International Conference on Information Science and Security (ICISS), 2016, pp. 1-6.
[27]. Sharafaldin I., Gharib A. et al. Towards a reliable intrusion detection benchmark dataset. Software Networking, issue 1, 2017, pp. 177–200.

Информация об авторах / Information about authors

Александр Игоревич ГЕТЬМАН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, доцент ВШЭ. Сфера научных интересов: анализ бинарного кода, восстановление форматов данных, анализ и классификация сетевого трафика.

Aleksandr Igorevich GETMAN – PhD in physical and mathematical sciences, senior researcher at ISP RAS, associate professor at HSE. Research interests: binary code analysis, data format recovery, network traffic analysis and classification.

Максим Николаевич ГОРЮНОВ – кандидат технических наук. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, системы анализа защищенности, машинное обучение, безопасная разработка программного обеспечения.

Maxim Nikolaevich GORYUNOV – Ph.D. Research interests: information security, intrusion detection systems, security analysis systems, machine learning.

Андрей Георгиевич МАЦКЕВИЧ – кандидат технических наук, доцент. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, системы антивирусной защиты, машинное обучение, криптографические методы защиты информации.

Andrey Georgievich MATSKEVICH – Ph.D., associate professor. Research interests: information security, intrusion detection systems, anti-virus protection systems, machine learning, cryptographic methods for protecting information.

Дмитрий Александрович РЫБОЛОВЛЕВ – кандидат технических наук. Сфера научных интересов: информационная безопасность, системы обнаружения вторжений, машинное обучение, криптографические методы защиты информации.

Dmitry Aleksandrovich RYBOLOVLEV – Ph.D. Research interests: information security, intrusion detection systems, machine learning, cryptographic methods for protecting information.

DOI: 10.15514/ISPRAS-2021-33(5)-6

Оптимизация программ на ProVerif для криптографических протоколов выработки общего ключа

[1,2] *Е.М. Винарский, ORCID:* 0000-0002-7328-0942 *<vinarskii@ispras.ru>*
[1] *А.В. Демаков, ORCID: 0000-0001-6573-7925 <demakov@ispras.ru>*
[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Национальный исследовательский университет «Высшая школа экономики», 109028, г. Москва, Покровский бульвар, д. 11*

Аннотация. Криптографические протоколы используются для установления безопасного соединения между “честными” агентами, которые общаются строго в соответствии с правилами протокола. Для того, чтобы убедиться, что спроектированный криптографический протокол является криптографически стойким, обычно используются различные программные инструменты. Однако, адекватная спецификация криптографического протокола обычно представляется в виде набора требований к последовательностям передаваемых сообщений, включая формат таких сообщений. Выполнение всех этих требований приводит к тому, что формальная спецификация для реального криптографического протокола становится громоздкой, в следствии чего её трудно анализировать формальными методами. Одним из таких стремительно развивающихся инструментов для формальной верификации криптографических протоколов является ProVerif. Отличительной особенностью инструмента ProVerif является то, что при больших протоколах он часто не справляется с их анализом, т.е. не может ни доказать безопасность протокола, ни опровергнуть её. В таких случаях прибегают либо к апроксимации задачи, либо к эквивалентным преобразованиям модели программы на языке ProVerif, упрощающих ProVerif-модель. В этой статье мы предлагаем способ для упрощения ProVerif-спецификаций для AKE-протоколов, использующих схему шифрования Эль-Гамаля. А именно, мы предлагаем эквивалентные преобразования, позволяющие построить ProVerif-спецификацию, которая упрощает анализ спецификации для инструмента ProVerif. Экспериментальные результаты для криптопротоколов Нидхема-Шрёдера и Yahalom показывают, что такой подход может быть перспективным для автоматической проверки реальных протоколов.

Ключевые слова: криптографические протоколы; эквивалентные преобразования; ProVerif

Для цитирования: Винарский Е.М., Демаков А.В. Оптимизация программ на ProVerif для криптографических протоколов выработки общего ключа. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 105-116. DOI: 10.15514/ISPRAS–2021–33(5)–6

Благодарности: Исследование поддержано грантом Минобрнауки РФ №075-15-2020-788

Optimization of ProVerif programs for AKE-protocols

[1,2] *E.M. Vinarskii, ORCID: 0000-0002-7328-0942 <vinaskii@ispras.ru>*
[1] *A.V. Demakov, ORCID: 0000-0001-6573-7925 <demakov@ispras.ru>*
[1] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[2] *National Research University Higher School of Economics, 11, Pokrovsky Boulevard, Moscow, 109028, Russia*

Abstract. Cryptographic protocols are used to establish a secure connection between "honest" agents who communicate strictly in accordance with the rules of the protocol. In order to make sure that the designed cryptographic protocol is cryptographically strong, various software tools are usually used. However, an adequate specification of a cryptographic protocol is usually presented as a set of requirements for the sequences of transmitted messages, including the format of such messages. The fulfillment of all these requirements leads to the fact that the formal specification for a real cryptographic protocol becomes cumbersome, as a result of which it is difficult to analyze it by formal methods. One of such rapidly developing tools for formal verification of cryptographic protocols is ProVerif. A distinctive feature of the ProVerif tool is that with large protocols, it often fails to analyze them, i.e. it can neither prove the security of the protocol nor refute it. In such cases, they resort either to the approximation of the problem, or to equivalent transformations of the program model in the ProVerif language, simplifying the ProVerif model. In this article, we propose a way to simplify the ProVerif specifications for AKE protocols using the El Gamal encryption scheme. Namely, we suggest equivalent transformations that allow us to construct a ProVerif specification that simplifies the analysis of the specification for the ProVerif tool. Experimental results for the Needham-Schroeder and Yahalom cryptoprotocols show that such an approach can be promising for automatic verification of real protocols.

Keywords: cryptographic protocols; equivalent transformations; ProVerif

For citation: Vinarskii E.M., Demakov A.V. Optimization of ProVerif programs for AKE-protocols. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 105-116 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–6

Acknowledgements. The research is supported by the grant of the Ministry of Education and Science of the Russian Federation No. 075-15-2020-788.

1. Введение

Криптографические протоколы (КП) широко используются в современных сетях. Они гарантируют аутентификацию участников, секретность сеансовых ключей и т.д. Инструмент для формальной верификации криптопротоколов ProVerif широко используется для проверки свойств безопасности для практически важных протоколов таких, как TLS 1.3, ARINC823 и др. [1, 2, 3]. Поскольку для криптографических протоколов необходимо выполнить множество требований, их адекватные формальные модели очень громоздки, в следствие чего трудно провести их формальную верификацию. Однако, если использовать специфические особенности инструмента ProVerif и проводить анализ только относительно заранее определённых свойств безопасности (например, только секретности общих сессионных ключей), то громоздкость протокола и, соответственно, сложность анализа может быть уменьшена. В частности, для инструмента ProVerif такое упрощение может быть проведено при помощи эквивалентных преобразований (ЭП), то есть таких преобразований, которые не уменьшают адекватность модели, но при этом упрощают анализ криптопротокола.

Одним из основных криптографических примитивов является функция шифрования. Однако, чтобы построить адекватную модель, описывающую операцию шифрования/расшифровния, нам необходимо добавить в ProVerif уравнение, связывающее эти две операции, пример такого уравнения изображён на рис. 1 и 2. Такие уравнения могут сильно усложнить ProVerif-модель, и, следовательно, проверку свойства безопасности для построенной ProVerif-модели.

В этой статье мы предлагаем эквивалентные преобразования, упрощающие ProVerif-модель с операциями шифрования/расшифрования. Мы используем следующие шаги для оптимизации представления криптографического протокола на языке ProVerif, чтобы упростить проверку свойств безопасности.

В настоящее время в формальной верификации криптографических протоколов господствуют две основных модели.

- **Символьная модель.** Криптографические примитивы рассматриваются как идеальные черные ящики, моделируемые функциональными символами в алгебре термов. Все вычисления проводятся в фиксированной теории, образованной функциональными символами и правилами редукции (уравнения) над ними. Сообщения – это термы на этих примитивах. Противник может проводить вычисления, используя только в рамках фиксированной теории. Лучшими символьными солверами на текущий момент являются Tamarin-prover [4, 5, 6] и ProVerif [7, 8, 9].
- **Модель доказуемой стойкости.** В данной модели сообщения – это битовые строки, криптографические примитивы – функции от битовых строк, а противник - любая вероятностная машина Тьюринга. Стойкость протокола выражается через стойкости его примитивов. Самым успешным инструментом на текущий момент является CryptoVerif [10].

В этой статье мы работаем с инструментом ProVerif, который реализует символьный подход к проверке КП. ProVerif принимает на вход модель протокола безопасности, определяя действия честных агентов и спецификацию желаемых свойств протокола. Для описания честных агентов протокола используется алгебра процессов. ProVerif автоматически переводит процессы в систему дизъюнктов Хорна и использует алгоритм, основанный на свободном выборе дизъюнкта из доступного множества, чтобы определить, является ли утверждение верным в заданной системе правил вывода (теории). Если такого логического вывода не существует, то желаемое свойство безопасности считается доказанным. Если вывод существует, то рассматриваемое свойство может быть не стойко. Полной гарантии нестойкости быть не может, так как дизъюнкты Хорна могут применяться неограниченное число раз, что является аппроксимацией криптопротокола, в следствие чего логический вывод может соответствовать ложной атаке [7, 8].

Работы, связанные с оптимизацией представления моделей КП для программных инструментов верификации достаточно развита. Авторы в статье [5] предлагают 2 подхода к упрощению доказательства лемм для модели КП на Tamarin Prover – (i) использование source lemm, которые позволяют ограничить размеры модели, в которой ищутся решения (ii) использование оракулов, помогающих подсказать путь, следуя которому можно доказать/опровергнуть лемму. Мы предлагаем использовать стандартный математический способ (эквивалентные преобразования) для упрощения модели. В статье [7] разбирается метод представления КП дизъюнктами Хорна (правила вывода), получившиеся правила вывода дополняются правилами вывода, определёнными для противника, получившиеся правила вывода формируют теорию. Угрозой считается вывод терма, сигнализирующего, что противник знает секрет. При оптимизации ProVerif-модели для КП мы прежде всего упрощали систему правил вывода. Были сделаны следующие шаги, чтобы оптимизировать ProVerif-модель для КП.

1. Ограничиваем свойства безопасности свойством секретности сессионного ключа.
2. Вводим понятие ProVerif-моделей, эквивалентных относительно формул криптопротоколов.
3. Предлагаем эквивалентные преобразования (ЭП) для функции шифрования и доказываем, что получившийся в результате таких преобразований криптопротокол остаётся эквивалентным исходному относительно свойства секретности сессионного ключа.

4. Проводим эксперименты для протоколов Нидхема-Шрёдера и Yahalom, которые показывают, что предложенные эквивалентные преобразования, действительно, упрощают анализ криптопротокола.

Структура статьи выглядит следующим образом. В разд. 2 представлены основные определения и показано, как ProVerif преобразует криптопротокол в систему дизъюнктов Хорна. Разд. 3 посвящена описанию эквивалентных преобразований. В разд. 4 представлены экспериментальные результаты. В разд. 4 содержится заключение статьи и описаны некоторые направления будущей работы.

2. Представление криптографического протокола на ProVerif и дизъюнкты Хорна

В этой секции мы кратко опишем синтаксические конструкции, включённые в язык ProVerif. На рис. 1 [7] изображена грамматика, определяющая правила построения термов и процессов на языке ProVerif; полное описание можно найти в [7].

$M, N ::=$	terms
x, y, z	variable
a, b, c, k	name
$f(M_1, \ldots, M_n)$	constructor application
$P, Q ::=$	processes
$\overline{M}\langle N\rangle.P$	output
$M(x).P$	input
0	nil
$P \mid Q$	parallel composition
$!P$	replication
$(\nu a)P$	restriction
$let\ x = g(M_1, \ldots, M_n)\ in\ P\ else\ Q$	destructor application
$if\ M = N\ then\ P\ else\ Q$	conditional
$\mathtt{event}(M).P$	event

Рис. 1. Синтаксис ProVerif

Термы в ProVerif могут состоять из переменных, идентификаторов, конструкторов и деструкторов. Конструкторы используются для построения термов, деструкторы – частичные функции (определены не над всеми входными данными). Например, деструктор $let\ x = g(M_1, \ldots, M_n)\ in\ P\ else\ Q$ пытается вычислить $g(M_1, \ldots, M_n)$; если это удается, то в переменную x записывается результат вычисления g и выполняется процесс P, иначе выполняется процесс Q. Действие $(\nu a)P$ обозначает, что процесс P обладает случайным секретным значением a.

2.1 Представление шифрования в ProVerif

Мы рассматриваем оптимизацию криптопротоколов, содержащих процедуру симметричного шифрования. При моделировании схемы шифрования в ProVerif обычно используется связка конструктора $Encrypt$ с деструктором $Decrypt$. На рис. 2 и 3 изображено представление на ProVerif схем шифрования.

```
fun Encrypt(bitstring,bitstring):bitstring.
  reduc forall mess:bitstring,key:bitstring
    Decrypt(Encrypt(mess,key),key) = mess
```

Рис. 2. Стандартная функция симметричного шифрования
Fig. 2. Standard symmetric encryption function

```
fun Encrypt(bitstring,bitstring,bitstring):bitstring.
  reduc forall mess:bitstring,key:bitstring,iv:bitstring
    Decrypt(Encrypt(mess,key,iv),key,iv) = mess
```

Рис. 3. Функция симметричного шифрования с вектором инициализации
Fig. *3. Function of symmetric encryption with an initialization vector*

Схему шифрования, изображённую на рис. 2 мы будем называть *стандартной*. Для такой схемы ProVerif порождает следующие дизъюнкты Хорна (ДХ):

- $attacker(mess) \wedge attacker(key) \Rightarrow attacker\big(Encrypt(mess, key)\big)$;
- $attacker\big(Encrypt(mess, key)\big) \wedge attacker(key) \Rightarrow attacker(mess)$.

Схему шифрования, изображённую на рис. 3 мы будем называть *шифрованием с инициализирующим вектором*. Такая схема порождает следующие дизъюнкты Хорна

- $attacker(mess) \wedge attacker(key) \wedge attacker(iv) \Rightarrow attacker\big(Encrypt(mess, key, iv)\big)$;
- $attacker\big(Encrypt(mess, key, iv)\big) \wedge attacker(key) \wedge attacker(key) \Rightarrow attacker(mess)$.

2.2 Схема шифрования Эль-Гамаля

В статье мы предполагаем, что вычисления происходят в группе G, где g – образующий элемент в G [11]. На рис. 4 представлено уравнение $(g^x)^y = (g^y)^x$ ProVerif для группы G. Схема шифрования Эль-Гамаля [11] предлагает способ передать ключ симметричного шифрования другой стороне в зашифрованном виде. Опишем, как схема Эль-Гамаля может быть представлена на языке ProVerif. Предположим, что криптосистема состоит из клиента $Clnt(x, g^x)$ и сервера $Serv(y, g^y)$, где x (y) – долговременный закрытый ключ клиента (сервера) и g^x (g^y) долговременный открытый ключ клиента (сервера). Пусть k – симметричный ключ, сгенерированный на стороне клиента, который необходимо передать серверу защищённым способом. Тогда клиент генерирует случайный $a \in_U G$, вычисляет $s = k \cdot (g^y)^a$ и отправляет серверу сообщение

$$Clnt \overset{(g^a, s)}{\rightarrow} Serv.$$

Сервер принимает s и вычисляет $k = s \cdot (g^a)^{-y} = k \cdot g^{ya} \cdot g^{-ya} = k$. Таким образом сервер получает ключ симметричного шифрования k, отправленный клиентом.

```
equation forall a1:bitstring,a2:bitstring;
  Exp(Exp(g, a1), a2) = Exp(Exp(g, a2), a1).
```

Рис. 4. Уравнение на языке ProVerif
Fig. 4. Equation in ProVerif language

```
fun ModMult(bitstring,bitstring,bitstring):bitstring.
  reduc forall a0:bitstring,a1:bitstring,a2:bitstring;
    ModDiv0(ModMult(a0,a1,a2),a1,a2) = a0.
```

Рис. 5. Уравнение на языке ProVerif
Fig. 5. Equation in ProVerif language

Схема шифрования Эль-Гамаля на языке ProVerif может быть реализована с помощью конструктора $ModMult$ и деструктора $ModDiv0$ (рис. 5). Такая схема представляется следующими ДХ, где $vCurveN$ – модуль эллиптической кривой:

- $attacker(key) \wedge attacker(g^y) \wedge attacker(vCurveN) \Rightarrow attacker\big(ModMult(key, g^y, vCurveN)\big)$;
- $attacker\big(ModMult(a_0, a_1, a_2)\big) \wedge attacker(a_1) \wedge attacker(a_2) \Rightarrow attacker(a_0)$.

2.3 Свойства безопасности криптографического протокола

В данном подразделе мы рассматриваем свойство секретности ключа (свойство достижимости [7]), в ProVerif свойство секретности ключа key задаётся формулой $\varphi_{sec} = attacker(key)$.

Чтобы проверить свойство секретности $\varphi_{sec} = attacker(key)$, ProVerif строит систему дизъюнктов Хорна и ищет способ вывести терм $attacker(key)$. Если противник смог вывести терм $attacker(key)$, то это следует понимать, как "противник может знать key". С другой стороны, если противник не может вывести событие $attacker(key)$, то "противник НЕ знает key". Такая аппроксимация восходит к той особенности, что каждый дизъюнкт Хорна может быть применён сколь угодно много раз, что может порождать "ложные" атаки. Тем не менее, задача проверки выводимости терма $attacker(key)$ алгоритмически разрешима (см. [7]), и предложенный алгоритм [7] показал свою эффективность при проверке практически важных протоколов (см. [4, 5, 6]).

Пусть π – криптопротокол, тогда модель π на языке ProVerif мы будем обозначать через $\mathcal{M}_\pi$, $P(\mathcal{M}_\pi)$ – система дизъюнктов Хорна (ДХ), построенных по модели $\mathcal{M}_\pi$. Обозначим через $\mathcal{N}_{pub}(\mathcal{N}_{priv})$ множество открытых (приватных) имён. Таким образом, систему ДХ, построенных по протоколу π и возможностям противника обозначим $\mathcal{R}_\pi = \mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}} = P(\mathcal{M}_\pi) \cup [\{attacker(a[])|a \in \mathcal{N}_{pub}\} \cup \{(Rn),(Rh),(Rl),(Rs)\}$, где правила вывода $\{(Rn),(Rh),(Rl),(Rs)\}$ представлены на рис. 6 и терм $message(x,y)$ означает, что сообщение y было отправлено в канал x. Построенную таким образом $\mathcal{R}_\pi$ мы называем правилами вывода, порождаемые моделью протокола π или просто *теорией* π.

Мы говорим, что криптопротокол π *удовлетворяет свойству секретности* $\varphi_{sec} = attacker(key)$ тогда и только тогда, когда $\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}} \vDash \varphi_{sec}$, то есть событие $attacker(key)$ НЕ выводимо в теории $\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}}$. Дадим определение теорий криптопротоколов, эквивалентных относительно формул.

Определение. Пусть φ – формула, π_1 и π_2 – криптопротоколы, тогда мы говорим, что $\pi_1 \sim_\varphi \pi_2$ если $\mathcal{R}_{\pi_1,\mathcal{N}_{pub},\mathcal{N}_{priv}} \vDash \varphi$ тогда и только тогда, когда $\mathcal{R}_{\pi_2,\mathcal{N}_{pub},\mathcal{N}_{priv}} \vDash \varphi$.

$\text{attacker}(b_0[x])$	**(Rn)**
For each public constructor f of arity n, $\text{attacker}(x_1) \wedge \ldots \wedge \text{attacker}(x_n) \Rightarrow \text{attacker}(f(x_1,\ldots,x_n))$	**(Rf)**
For each public destructor g, for each rewrite rule $g(M_1,\ldots,M_n) \to M$ in $\text{def}(g)$, $\text{attacker}(M_1) \wedge \ldots \wedge \text{attacker}(M_n) \Rightarrow \text{attacker}(M)$	**(Rg)**
$\text{message}(x,y) \wedge \text{attacker}(x) \Rightarrow \text{attacker}(y)$	**(Rl)**
$\text{attacker}(x) \wedge \text{attacker}(y) \Rightarrow \text{message}(x,y)$	**(Rs)**

Рис. 6. Дизъюнкты Хорна для противника
Fig. 6. Horn's disjuncts for an enemy

3. Эквивалентные преобразования

В этом разделе мы описываем некоторые техники, позволяющие провести эквивалентные преобразования над ProVerif-моделью, которые позволяют упростить модель протокола π, оперирующий над открытыми именами $\mathcal{N}_{pub}$ и закрытыми именами $\mathcal{N}_{priv}$ ($\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}}$). Как показывают наши эксперименты (разд. 4), для оптимизированной ProVerif-модели приведённые здесь эквивалентные преобразования упрощают проверку свойства секретности ключей. Для простоты изложения мы пишем $\mathcal{R}_\pi$ вместо $\mathcal{R}_{\pi,\mathcal{N}_{pub},\mathcal{N}_{priv}}$.

3.1 Оптимизация в представлении схемы шифрования

Мы докажем, что, если мы проверяем безопасность криптопротокола на ProVerif относительно свойств секретности ключа φ_{sec}, определённых в подразд. 2.3, то схема шифрования с инициализирующем значением эквивалентна стандартной схеме шифрования.

Пусть $\pi^{e_{iv}}$ – криптопротокол, использующий схему шифрования с вектором инициализации iv, тогда $\mathcal{M}_{\pi}^{e_{iv}}$ содержит конструкторы/деструкторы, изображённые на рис. 2. Обозначим соответствующую систему ДХ криптопротокола π через $P^{e_{iv}}(\mathcal{M}_{\pi})$, соответственно, теорию π обозначим через $\mathcal{R}_{\pi}^{e_{iv}}$.

Обозначим через $\mathcal{M}_{\pi}^{\overline{e}}$ модель крипторотокола π, в которой все вхождения конструктора $Encrypt_{iv}$ (рис. 3) заменены на конструктор $Encrypt$ (рис. 2) и все вхождения деструктора $Decrypt_{iv}$ заменены на деструктор $Decrypt$. Соответствующую теорию обозначим через $\mathcal{R}_{\pi}^{\overline{e}}$. Тогда справедлива следующее утверждение, в которой утверждается, что теории $\mathcal{R}_{\pi}^{e_{iv}}$ и $\mathcal{R}_{\pi}^{\overline{e}}$ эквивалентны относительно формулы $\varphi_{sec} = attacker(key)$.

Утверждение 1. Пусть $iv \in \mathcal{N}_{pub}$, тогда $\mathcal{R}_{\pi}^{e_{iv}} \sim_{\varphi_{sec}} \mathcal{R}_{\pi}^{\overline{e}}$, т.е. $\mathcal{R}_{\pi}^{e_{iv}} \vDash \varphi_{sec}$ тогда и только тогда, когда $\mathcal{R}_{\pi}^{\overline{e}} \vDash \varphi_{sec}$.

Доказательство.

⇒ Пусть $\mathcal{R}_{\pi}^{e_{iv}} \vDash \varphi_{sec}$, но при этом неверно, что $\mathcal{R}_{\pi}^{\overline{e}} \vDash \varphi_{sec}$. Тогда терм $attacker(key)$ выводим в теории $\mathcal{R}_{\pi}^{\overline{e}}$, рассмотрим такой вывод ρ (последовательное применение ДХ) в теории $\mathcal{R}_{\pi}^{\overline{e}}$, $\rho = r_1, \dots, r_n$. Такой логический вывод ρ обязательно содержит ДХ, порождаемые конструкторами $Encrypt$ и деструкторами $Decrypt$, так как иначе такой же логический вывод ρ существовал бы и в теории $\mathcal{R}_{\pi}^{e_{iv}}$.

Рассмотрим такой вывод $\rho = r_1, \dots, r_n$, и пусть r_j – применение ДХ $attacker(mess) \wedge attacker(key) \Rightarrow attacker\big(Encrypt(mess, key)\big)$. Из того, что $iv \in \mathcal{N}_{pub}$, следует, что противнику доступен терм $attacker(iv)$, следовательно, логический вывод ρ может содержать ДХ $attacker(mess) \wedge attacker(key) \wedge attacker(iv) \Rightarrow attacker\big(Encrypt(mess, key, iv)\big)$. Аналогично, если логический вывод ρ содержит применение ДХ $attacker\big(Encrypt(mess, key)\big) \wedge attacker(key) \Rightarrow attacker(mess)$, то может быть применён и ДХ $attacker\big(Encrypt(mess, key, iv)\big) \wedge attacker(iv) \wedge attacker(key) \Rightarrow attacker(mess)$.

Таким образом, логический вывод $\overline{\rho}$ существует и в теории $\mathcal{R}_{\pi}^{e_{iv}}$, то есть неверно, что $\mathcal{R}_{\pi}^{e_{iv}} \vDash \varphi_{sec}$ – противоречие.

⇐ пусть теперь $\mathcal{R}_{\pi}^{\overline{e}} \vDash \varphi_{sec}$, тогда факт, что $\mathcal{R}_{\pi}^{e_{iv}} \vDash \varphi_{sec}$, доказывается аналогично.

Аналогичное утверждение может быть доказано и для других схем шифрования. Таким образом, далее мы будем рассматривать только теории, в которых используются только стандартные функции шифрования.

3.2 Использование одного ключа шифрования при схеме шифрования Эль-Гамаля

В данном разделе мы предлагаем эквивалентные преобразования, которые позволяют упростить модели криптопротоколов, использующих симметричное шифрование с использованием схемы Эль-Гамаля. Пусть одна сторона криптопротокола π передаёт другой зашифрованные сообщения $mess_1$ на симметричном ключе k_1 и $mess_2$ на симметричном ключе k_2. Мы рассматриваем свойство секретности криптопротокола π: $\varphi_{sec_1} = attacker(k_1)$.

Мы покажем, что схема с двумя сессионными симметричными ключами k_1, k_2, изображённая на рис. 7, эквивалентна схеме с одним сессионным симметричным ключом k_1, изображённой

на рис. 8. Обозначим теорию, построенную по $\mathcal{M}^{k_1,k_2}(\pi)$, через $\mathcal{R}_\pi^{k_1,k_2}$, в то время, как построенную по $\mathcal{M}^{k_1}(\pi)$ через $\mathcal{R}_\pi^{k_1}$. Мы доказываем, что теории $\mathcal{R}_\pi^{k_1,k_2}$ и $\mathcal{R}_\pi^{k}$ эквивалентны относительно формулы $\varphi_{sec} = attacker(k)$.

```
processC(skC, pkC) =
...
new k1:bitstring;
new a1:bitstring;
let g_pkS_a1 = Exp(pkS, a1) in
let cms_k1 = ModMult(k1, g_a1) in
let g_a1 = Exp(vBasePoint, a1) in
let enc_mess1 = Encrypt(mess1, k1) in
out(c, (g_a1, cms_k1, enc_mess1));
...
new k2:bitstring;
new a2:bitstring;
let g_pkS_a2 = Exp(pkS, a2) in
let cms_k2 = ModMult(k2, g_a2) in
let g_a2 = Exp(vBasePoint, a2) in
let enc_mess2 = Encrypt(mess2, k2) in
out(c, (g_a2, cms_k2, enc_mess2));
...

processS(skS, pkS) =
...
in(c,(g_a1:bitstring,cms_k1:bitstring,enc_mess1:bitstring));
let k1 = ModDiv0(cms_k1,Exp(g_a1,skS)) in
let mess1 = Decrypt(enc_mess1, k1) in
...
in(c,(g_a2:bitstring,cms_k2:bitstring,enc_mess2:bitstring));
let k2 = ModDiv0(cms_k2,Exp(g_a2,skS)) in
let mess2 = Decrypt(enc_mess2, k2) in
...
```

Рис. 7. Схема протокола с двумя сессионными ключами симметричного шифрования
Fig. 7. Scheme of the protocol with two session keys of symmetric encryption

```
processC(skC, pkC) =
...
new k:bitstring;
new a:bitstring;
let g_pkS_a = Exp(pkS, a) in
let cms_k = ModMult(k, g_a) in
let g_a = Exp(vBasePoint, a) in
let enc_mess = Encrypt(mess1, k) in
out(c, (g_a, cms_k, enc_mess1));
...
new k2:bitstring;
new a2:bitstring;
let g_pkS_a2 = Exp(pkS, a2) in
let cms_k2 = ModMult(k2, g_a2) in
let enc_mess2 = Encrypt(mess2, k) in
let g_a2 = Exp(vBasePoint, a2) in
out(c, enc_mess2);
...

processS(skS, pkS) =
...
in(c,(g_a:bitstring,cms_k:bitstring,enc_mess:bitstring));
let k = ModDiv0(cms_k,Exp(g_a,skS)) in
let mess1 = Decrypt(enc_mess1, k) in
...
in(c,enc_mess2:bitstring);
let mess2 = Decrypt(enc_mess2, k) in
...
```

Рис. 8. Схема протокола с одним сессионным ключом симметричного шифрования
Fig. 8. Scheme of the protocol with one session symmetric encryption key

Утверждение 2. $\mathcal{R}_{\pi}^{k_1,k_2} \sim_{\varphi_{sec_1}} \mathcal{R}_{\pi}^{k}$.

Доказательство. Отметим, что обе теории определены над одним и тем же пространством имён: $\mathcal{N}_{pub} \cup \mathcal{N}_{priv}$.

Пусть верно $\mathcal{R}_{\pi}^{k_1} \vDash \varphi_{sec}$, но при этом неверно $\mathcal{R}_{\pi}^{k_1,k_2} \vDash \varphi_{sec_1}$. Тогда терм $attacker(k_1)$ выводим в теории $\mathcal{R}_{\pi}^{k_1,k_2}$, рассмотрим такой вывод $\rho = r_1, \dots, r_n$ (последовательное применение ДХ) в теории $\mathcal{R}_{\pi}^{k}$. ДХ, порождаемые теориями $\mathcal{R}^{k}(\pi)$ и $\mathcal{R}^{k_1,k_2}(\pi)$ отличаются тем, что теория $\mathcal{R}^{k_1,k_2}(\pi)$ содержит термы $attacker\big(Exp(g,a_2)\big)$, $attacker\Big(ModMult\big(k_2, Exp(g,a_2)\big)\Big)$, $attacker\big(Encrypt(mess2, k_2)\big)$, которых не содержит теория $\mathcal{R}^{k_1}(\pi)$.

Таким образом, так как логический вывод ρ существует в теории $\mathcal{R}^{k_1,k_2}(\pi)$, но не существует в теории $\mathcal{R}^{k_1}(\pi)$, то ρ содержит термы $attacker\big(Exp(g,a_2)\big)$, $attacker\Big(ModMult\big(k_2, Exp(g,a_2)\big)\Big)$, $attacker\big(Encrypt(mess_2, k_2)\big)$, так как иначе такой вывод существовал бы и в теории $\mathcal{R}^{k_1}(\pi)$.

При этом термы $attacker\big(Exp(g,a_2)\big)$, $attacker\Big(ModMult\big(k_2, Exp(g,a_2)\big)\Big)$ могут быть выведены противником согласно правилу вывода (Rh) с рис. 6. Аналогично, если в выводе ρ, приводящий к обладанию противником терма $attacker(k_1)$, то есть терм $attacker\big(Encrypt(mess_2, k_2)\big)$ и существует вывод $\overline{\rho} = \rho[k_1/k,\ k_2/k]$, следовательно, неверно, что $\mathcal{R}^{k_1}(\pi) \vDash attacker(k_1)$ – противоречие.

Теперь рассмотрим, как ещё может быть упрощена схема шифрования Эль-Гамаля. Пусть клиент использует симметричный ключ шифрования k_C, переданный серверу с использованием схемы Эль-Гамаля. В то время, как сервер использует симметричный ключ шифрования k_S, переданный клиенту с использованием схемы Эль-Гамаля. Мы предлагаем эквивалентные преобразования, которые позволяют перейти к модели, в которой используется только один ключ симметричного шифрования. Обозначим теорию, построенную по $\mathcal{M}^{k_C,k_S}(\pi)$ через $\mathcal{R}_{\pi}^{k_C,k_S}$, в то время, как построенную по $\mathcal{M}^{k_C}(\pi)$ через $\mathcal{R}_{\pi}^{k_C}$. Тогда справедливо утверждение 3, которое доказывается аналогично утверждению 2.

Утверждение 3. $\mathcal{R}_{\pi}^{k_C,k_S} \sim_{\varphi_{sec_C}} \mathcal{R}_{\pi}^{k_C}$.

Таким образом, мы показали, что при проверке свойства секретности сессионного ключа в протоколе со схемой шифрования Эль-Гамаля, в которой используются несколько сессионных ключей можно перейти к проверке свойства секретности сессионного ключа в протоколе, в котором используется всего один сессионный ключ. Также мы показали, что схемы шифрования с вектором инициализации эквивалентны стандартным схемам шифрования для ProVerif модели. В разд. 4 мы покажем, как использовать эти результаты для оптимизации ProVerif-кода.

4. Экспериментальные результаты

В данном разделе представлены эксперименты с протоколами Нидхема-Шрёдера и Yahalom, которые показывают эффективность предложенных эквивалентных преобразований для ProVerif-кода криптопротокола.

Мы работали с модификацией протокола Нидхема-Шрёдера, изображённой на рис. 9. В оригинале в этой схеме используется 3 сессионных ключа. Мы проверяли свойство секретности сессионного ключа $\varphi_{sec} = attacker(key)$. Мы модифицировали эту схему, сначала согласно Теореме 1 перешли к стандартной функции шифрования, а затем применили теоремы 2 и 3, чтобы перейти к эквивалентным ProVerif-моделям с меньшим числом правил вывода. Далее мы запустили Proverif, чтобы проверить 3 версии ProVerif-моделей: с тремя сессионными ключами, с двумя сессионными ключами и с одним

сессионным ключом. Все исходные коды экспериментов доступны по ссылке [12]. В качестве метрики мы использовали количество правил, которые использовал ProVerif для доказательства формулы секретности сессионного ключа $\varphi_{sec} = attacker(key)$.

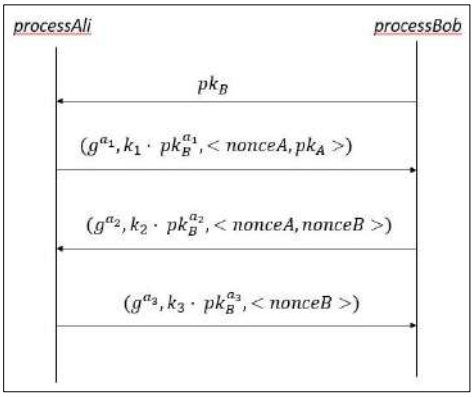


Рис. 9. Схема протокола Нидхема-Шрёдера с тремя сессионным ключами симметричного шифрования

Fig. 9. Scheme of the Needham-Schroeder protocol with three session keys of symmetric encryption

В результате, если используется 3 сессионных ключа, то ProVerif использует 45800 правил, если 2 сессионных ключей, то ProVerif использует 3000 правил и если 1 сессионных ключа, то ProVerif использует 200 правил. Таким образом, мы видим существенный выигрыш при использовании оптимизаций, предложенных в статье.

Аналогичные эксперименты мы провели с ProVerif-моделью протокола Yahalom (мхема изображена на рис. 10). Мы проверяли свойство секретности сессионного ключа $\varphi_{sec} = attacker(k)$, все исходные коды экспериментов доступны по ссылке [12]. Мы модифицировали эту схему, сначала согласно утверждениям 2 и 3, перейдя к схеме шифрования Эль-Гамаля с одним сессионным ключом. В результате, если используется 2 сессионных ключа, то ProVerif использует 2600 правил, а если 1 сессионный ключ, то 200 правил. Таким образом, снова, мы видим существенный выигрыш при использовании оптимизаций, предложенных в статье для доказательства свойства секретности сессионного ключа с использованием ProVerif.

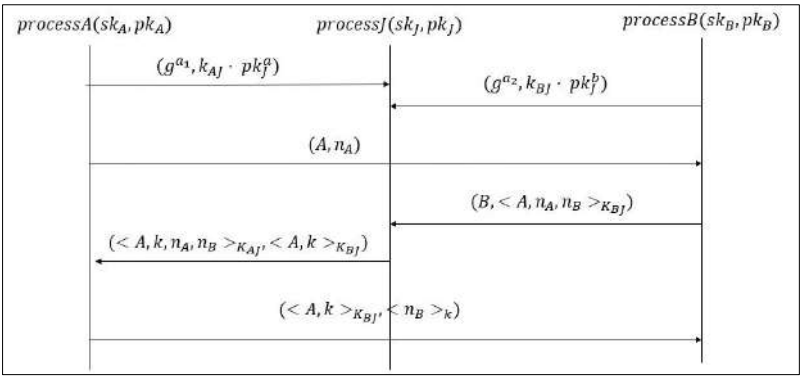


Рис. 10. Схема протокола Yahalom с двумя сессионным ключами симметричного шифрования

Fig. 10. Scheme of the Yahalom protocol with two session keys of symmetric encryption

5. Заключение

В этой статье мы предложили техники для оптимизации ProVerif-модели криптографического протокола. Мы предложили несколько преобразований ProVerif-модели и доказали, что они являются эквивалентными относительно формулы секретности сессионного ключа. Эффективность данных преобразований была проверена на реализации протоколов Нидхема-Шрёдера и Yahalom.

Далее мы планируем предложить различные техники оптимизации для других конструкций языка ProVerif и провести эксперименты с практически важными протоколами.

Список литературы / References

[1]. B. Blanchet. Symbolic and Computational Mechanized Verification of the ARINC823 Avionic Protocols. In Proc. of the 30th IEEE Computer Security Foundations Symposium (CSF'17), 2017, pp. 68-82.
[2]. K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. Research Report RR-9040, Inria, 2017.
[3]. K. Bhargavan, B. Blanchet, and N. Kobeissi. Verified Models and Reference Implementations for the TLS 1.3 Standard Candidate. In Proc. of the IEEE Symposium on Security and Privacy (S&P'17), pp. 483-503.
[4]. S. Meier, B. Schmidt et al. The TAMARIN prover for the symbolic analysis of security protocols. In Proc. of the 25th International Conference on Computer Aided Verification, 2013. pp. 696-701.
[5]. S. Meier. Advancing automated security protocol verification. Doctoral Thesis. ETH Zurich, 2013, 214 p.
[6]. B. Schmidt. Formal analysis of key exchange protocols and physical protocols. Doctoral Thesis. ETH Zurich, 2012, 199 p.
[7]. B. Blanchet. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. Foundations and Trends in Privacy and Security, vol. 1, no. 1-2, 2016, pp 1-135.
[8]. B. Blanchet. Automatic Verification of Correspondences for Security Protocols. Journal of Computer Security, vol. 17, issue 4, 2009, pp 363-434.
[9]. B. Blanchet. Automatic Verification of Security Protocols in the Symbolic Model: the Verifier ProVerif. Lecture Notes in Computer Science, vol. 8604, 2012, pp. 54-87.
[10]. B. Blanchet. CryptoVerif: A Computationally Sound Mechanized Prover for Cryptographic Protocols. In Proc. of the Dagstuhl Seminar on Formal Protocol Verification Applied, 2007.
[11]. T. Elgamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Transactions on Information Theory, vol. 31, no. 4, 1985, pp. 469-472, 1985.
[12]. E. Vinarskii. URL: https://github.com/vinevg1996/proverif_code_optimisation, accessed 24.10.2021.

Информация об авторах / Information about authors

Евгений Максимович ВИНАРСКИЙ – аспирант факультета ФКН ВШЭ, старший лаборант ИСП РАН. Научные интересы: конечные автоматы, верификация.

Evgenii Maksimovich VINARSKII – PhD student, Department of Computer Science, Higher School of Economics. Research interests: finite state machines, software testing and verification.

Алексей Васильевич ДЕМАКОВ – кандидат физико-математических наук, ведущий научный сотрудник. Научные интересы: графы программ, тестирование.

Alexey Vasilyevich DEMAKOV – PhD in Physics and Mathematics, Leading Researcher. Research interests: program graphs, software testing and verification.

DOI: 10.15514/ISPRAS-2021-33(5)-7

CC BY

Формальный язык первичных спецификаций криптографических протоколов

С.Е. Прокопьев, ORCID: 0000-0002-4410-2565 <sepr@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25
АО «НПК «Криптонит»,
115114, Москва, Шлюзовая набережная, 4 (БЦ «Россо Рива»)

Аннотация. Применение формальных методов в разработке реализаций сетевых протоколов способствует повышению качества этих реализаций. Наибольший эффект даст изложение на формальном языке первичных спецификаций протоколов, публикуемых в документах RFC. В статье предлагается формальный язык описания криптографических протоколов, нацеленный на более полное, по сравнению с существующими языками первичных спецификаций, выполнение следующих требований: лаконичность, декларативность, выразительность, однозначность, исполнимость, возможность автоматического извлечения из спецификаций наборов тестов высокого качества. Основа языка – вычислитель, специально разработанный для предметной области криптографических протоколов, – т. н. C2-машина. В статье при изложении принципов построения и функционирования C2-машины используется метод последовательного уточнения модели: сначала вводится недетерминированный автомат, на котором в наиболее общем виде задается операционный контракт протокола, затем вводится машина C0, на которой демонстрируется предлагаемый подход к специфицированию недетерминизма протоколов, и далее – машина C1, на которой в формализованном виде задается семантика базовых инструкций C2-машины. Статья иллюстрирована примерами для протокола TLS.

Ключевые слова: криптографические протоколы; формальные спецификации; C2-машина

Для цитирования: Прокопьев С.Е. Формальный язык первичных спецификаций криптографических протоколов. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 117-136. DOI: 10.15514/ISPRAS–2021–33(5)–7

Благодарности: Исследование поддержано грантом Минобрнауки России № 075-15-2020-788.

A formal language for primary specifications of the cryptographic protocols

S.E. Prokopev, ORCID: 0000-0002-4410-2565 <sepr@ispras.ru>

Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
JSRPC «NPK Kryptonite»,
4, Shluzovaya emb., Moscow, 115114, Russia

Abstract. Use of the formal methods in the development of the protocol implementations improves the quality of these implementations. The greatest benefit would result from the formalizing of the primary specifications usually contained in the RFC documents. This paper proposes a formal language for primary specifications of the cryptographic protocols, which aims at fulfilling (in a higher degree than in the existing approaches) the conditions required from primary specifications: they have to be concise, declarative, expressive, unambiguous

and executable; in addition, the tools supporting the language have to provide the possibility of automatic deriving of the high quality test suites from the specifications. The proposed language is based on a machine (dubbed the C2-machine) specifically designed for the domain of the cryptographic protocols. Protocol specification is defined as a program of the C2-machine. This program consists of two parts: the definition of the protocol packets and the definition of the non-deterministic behavior of the protocol parties. One execution of the program simulates one run of the protocol. All the traces, which can be potentially produced by such execution, by definition, comprise all conformant traces of the protocol; in other words, the program of the C2-machine defines the operational contract of the protocol. In the paper, to make the design and operational principles of the C2-machine easier to understand, two abstractions of the C2-machine are presented: C0-machine and C1-machine. C0-machine is used to explain the approach taken in expressing non-determinism of the protocols. The abstraction level of the C1-machine (which is a refinement of C0-machine) is enough to define the semantics of the basic C2-machine instructions. To enhance the readability of the programs and to reach the level of declarativeness and conciseness of the formalized notations used in some of conventional RFCs (e.g. TLS RFCs), C2-machine implements some syntactic tricks on top of the basic instructions. To use C2-specifications in the black-box testing, the special form of the C2-machine (C2-machine with excluded role) is presented. Throughout the paper the proposed concepts are illustrated by examples from the TLS protocol.

Keywords: cryptographic protocols; formal specifications; C2-machine

For citation: Prokopev S.E. A formal language for primary specifications of the cryptographic protocols. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 117-136 (in Russian). DOI: 10.15514/ISPRAS-2021-33(5)-7

Acknowledgements. The work is supported by a grant from the Ministry of Education and Science of Russia.

1. Введение

Перед тем, как пользователи получат в свое распоряжение реализацию криптографического протокола, специалисты проводят ряд обязательных исследований, включая анализ криптографических свойств протокола в разных моделях (в стандартной модели, модели Долева-Яо и др.) и анализ корректности программной реализации протокола с использованием статических и динамических методов.

Формальные методы позволяют повысить качество исследований, для некоторых из них позволяют задать метрики оценки достигнутого уровня качества, а также упрощают задачу верификации полученных результатов. При этом использование формальных методов не всегда означает удорожание исследований, так как удачная формализация может обеспечить возможность их автоматизации и, как следствие, снизить трудоемкость их проведения.

На текущий момент формальные методы используются в разработке программных реализаций [2, 3], в тестировании [4, 1], в анализе криптографических свойств протоколов [5, 6].

Во всех перечисленных случаях по первичным описаниям (спецификациям) протоколов, изложенным на естественном языке (например, в документах RFC), вручную разрабатываются вторичные формальные спецификации (к которым мы относим и исходные тексты реализаций).

Можно попытаться уменьшить объем ручного труда, обеспечив автоматическую трансляцию вторичных спецификаций из одной нотации в другую. Например, спецификации в нотации ESPI (пригодной для анализа инструментом ProVerif) можно пытаться конвертировать в исходные тексты на языке Java (см. [7]), или наоборот, исходные тексты на языке C пытаться конвертировать в спецификации для ProVerif (см. [8]).

Хотя вторичные спецификации, полученные автоматически из других вторичных спецификаций, могут решать свою задачу менее эффективно, чем написанные вручную, на практике фактор минимальной трудоемкости их получения (усиливаемый постоянными корректировками первичных спецификаций в ходе жизненного цикла разработки протоколов), этот недостаток может перевесить. Как отмечено в [9], *«...a developing*

environment where several tools can be used for different purposes on the base of the same specification model can be much more convenient than having different tools, even if more efficient, working on input models with their own different languages».

Помимо автоматической трансляции спецификаций из одной нотации в другую, есть и путь встраивания методов анализа, работающих в своих моделях со своей специфической нотацией, в другие модели. Например, в работе [10] метод верификации протоколов в модели Долева-Яо на основе дизъюнктов Хорна встроен во фреймворк Java Cryptography Architecture.

Важнейшим вопросом является то, насколько точно вторичная спецификация отражает требования, предъявленные к протоколу в первичной спецификации, то есть насколько она полна и корректна. Возникает желание уменьшить «семантический разрыв» между первичной неформальной спецификацией и вторичной формальной спецификацией путем повышения уровня формализации первичной спецификации. В первичных спецификациях многих протоколов такие попытки делаются.

Например, формализованная запись форматов сообщений в RFC 8446 (TLS версии 1.3) делает спецификацию протокола в этой части более строгой (уменьшает неоднозначность). Однако этот формализм не исполним, то есть он не поддержан инструментом трансляции в некоторый «полезный» язык, например, в промышленный язык программирования, на котором реализуется протокол, или в нотацию, принятую в некоторой системе тестирования. Пример исполнимого формализма – первичная спецификация протокола Signal, содержащая вставки кода на языке Python. Эти вставки можно без изменений перенести в исходные тексты реализации, при этом они достаточно хорошо читаемы ввиду лаконичности синтаксиса этого языка. Другой пример: в работе [11] для первичных спецификаций предложена формальная нотация для описания управляющей логики протокола, поддержанная инструментом трансляции в код на C++ и Java.

Сформулируем следующие требования к «идеальному» языку первичных спецификаций криптографических протоколов:

1) хорошая читаемость (лаконичность и декларативность);
2) высокая выразительность: часть спецификации, выраженная на нем, должна охватывать на порядок больше требований к протоколу, чем часть, выраженная на естественном языке;
3) однозначность;
4) возможность генерации из спецификации произвольного числа контрольных примеров с параметрами, интересующими исследователя;
5) возможность автоматической трансляции спецификаций в реализации, пригодные для встречной работы с промышленными реализациями протоколов;
6) возможность автоматического извлечения из спецификаций наборов тестов (как тестов на конформность, так и фаззинг-тестов), существенно превышающих по качеству наборы, которые можно сформировать вручную. При этом должна выдаваться адекватная оценка качества проведенного тестирования;
7) возможность автоматической трансляции спецификаций в языки специализированных инструментов анализа криптографических свойств протоколов (ProVerif, Tamarin, CryptoVerif и др.).

В настоящей работе представлена формальная нотация, нацеленная на более полное, чем в существующих подходах, выполнение предъявленных выше требований. Нотация основана на вычислителе (так называемой *C2-машине),* специально разработанном для предметной области промышленных криптографических протоколов, таких как TLS, IKEv2 и др.

C2-машина исполняет программы, содержащие *гардированные команды* (guarded command programs, [12]). В таких программах существуют точки выполнения, в которых дальнейшее

вычисление задается наборами пар *guard/update*, где *guard* – предикат (гард), задающий условия, при которых действие *update* может быть выполнено. В случае, когда одновременно истинны несколько гардов, недетерминированно выбирается один из них. Детали исполнения программ С2-машины, касающиеся «гардирования», можно найти в разд. 2 и 3 настоящей статьи.

Тексты программ для С2-машины и предлагается использовать в качестве первичных спецификаций протоколов.

Каждый запуск спецификации на С2-машине дает трассу сообщений протокола. Так же, как в подходе NCT [13], протокол определяется множеством своих конформных трасс, которое, в свою очередь, задается как множество всех трасс сообщений, которые потенциально может произвести машина, выполняя спецификацию. В целом идея использования специализированных машин при моделировании криптографическими протоколов, не нова (примеры: машины CrAM [14] и CVM [8]).

В настоящей статье изложение принципов построения С2-машины проводится методом последовательного уточнения модели (model refinement). Сначала в разд. 2 представлен простой недетерминированный автомат *A* (первое приближение С2-машины), в контексте которого рассматривается понятие *операционного контракта* протокола. В разд. 3 рассмотрен предлагаемый подход к захвату недетерминизма протоколов с помощью *последовательностей* и *гардирования* значений полей сообщений протоколов и представлено второе приближение С2-машины – машина *C0*. В разд. 4 рассмотрены *инструкции*, *процедуры*, *выражения* и *программы* С2-машины и представлено ее третье приближение – машина *C1*. Разд. 5 посвящен способам представления процедур С2-машины в лаконичном и декларативном виде, разд. 6 – тестированию реализаций протоколов в режиме «черного ящика» с помощью так называемой *С2-машины с исключенной ролью*.

2. Операционный контракт протокола

Рассмотрим недетерминированный автомат *A*, определение которого адаптировано к предметной области протоколов передачи сообщений:

A = (State, Role, Packet, ProtocolSpec, calcpacket, s0), где

State – множество состояний автомата, *Role* – множество ролей протокола, *Packet* – множество пакетов протокола, *ProtocolSpec* – множество спецификаций протоколов, *s0 :: State* – начальное состояние автомата,

calcpacket :: (ProtocolSpec, State, Role, Role) →: (State, Packet)∪{stop, fail} – недетерминированная функция вычисления пакета протокола и перехода в следующее состояние; *stop, fail* – особые значения, доопределяющие функцию *calcpacket*.

Примечание: двойное двоеточие :: обозначает принадлежность элемента к множеству; запись (X,...Y) – декартово произведение множеств X,...Y; стрелочка с двоеточием "→:" – недетерминированную функцию.

Равенство *calcpacket (protSpec, si, src, dst) = (sj, pkt)* интерпретируется следующим образом: автомат, работая по спецификации *protSpec*, находясь в состоянии *si* и получая в качестве входного символа пару *(src, dst)*, переходит в состояние *sj* и выдает выходной символ *pkt*.

Алгоритм работы автомата *A* (для некоторого значения *protSpec*):

0. в момент запуска автомат находится в состоянии *s0*;

1. из множества *(Role,Role)* недетерминированно выбирается *(src,dst)*;

2. если *calcpacket (protSpec, si, src, dst) :: {stop, fail}*, то конец работы;

3. если *calcpacket (protSpec, si, src, dst) = (sj, pkt)*, то возврат на шаг 1.

Трассой запуска автомата A называется список троек *(src,dst,pkt)*, снимаемых внешним наблюдателем на шаге 3. В общем случае автомат может не останавливаться и производить бесконечные трассы.

Ввиду недетерминизма шага 1 (*недетерминизма чередования*, interleaving choice) и недетерминизма функции *calcpacket* (*недетерминизма спецификации*, specification choice), разные запуски автомата, как правило, будут производить разные трассы. Таким образом, автомат задает язык, словами которого являются все трассы, которые он потенциально может произвести (трассы, завершившиеся *fail*, в язык не включаются).

Если язык, производимый автоматом *A*, совпадает со множеством конформных трасс моделируемого протокола, то говорят (см., например, [15]), что этот автомат задает *операционный контракт* протокола.

Автомат *A* является первым приближением C2-машины.

3. Специфицирование недетерминизма протоколов

3.1 Последовательности сессий, пакетов и фрагментов пакетов

Ключевым понятием предлагаемого подхода к моделированию протоколов являются *последовательности*. Предполагается, что в любом протоколе всегда присутствуют две последовательности: *_sessions* – последовательность сессий протокола (это последовательность всех пакетов из трассы автомата *A* из раздела 1) и *_packets* – последовательность пакетов, принадлежащих одной сессии протокола.

Обычно последовательности образуют иерархию. Например, элемент последовательности *_sessions* содержит элементы последовательности *_packets*. В каждом пакете протокола может быть своя иерархия последовательностей фрагментов.

Пример: иерархия (неполная) последовательностей протокола TLS:

_sessions → _packets
_packets → _records
_records → _handshMess
_handshMess → _clientSuites
_handshMess → _clientHelloExts
_clientHelloExts → _clntSuppGroups
_clientHelloExts → _clntPointFormats
_clientHelloExts → _clntSigAlgs
_clientHelloExts → _clntSuppVersions
_clientHelloExts → _clntKeyShares
_handshMess → _serverHelloExts
_handshMess → _certRequestExts
_certRequestExts → _certRequestSigAlgs
_handshMess → _certificates
_certificates → _certificateExts

Каждому фрагменту пакета, включая пакет целиком, можно сопоставить список пар («идентификатор последовательности», «номер элемента последовательности»), отражающий место этого фрагмента в иерархии последовательностей протокола. Будем называть этот список *sequence-индексом* (или просто, *индексом*) фрагмента пакета.

Пример: в протоколе TLS фрагмент пакета, содержащий четвертый по порядку элемент расширения *ServerHelloExtensions* сообщения *ServerHello* может иметь следующий sequence-индекс:

[(_sessions,1), (_packets,2), (_records,1), (_handshMess,1), (_serverHelloExts,4)].

Разные части одной последовательности могут быть «раскиданы» по разным пакетам и по разным местам пакетов. Например, в протоколе TLS версии 1.3 в первых двух пакетах первой сессии могут быть фрагменты со следующими индексами:

пакет 1:

[(_sessions, 1), (_packets, 1), (_records, 1), (_handshMess, 1)] – фрагмент пакета, в котором размещено сообщение *ClientHello* подпротокола TLSHandshakeMessages подпротокола TLSRecords;

пакет 2:

[(_sessions, 1), (_packets, 2), (_records, 1), (_handshMess, 1)] – фрагмент пакета, в котором размещено сообщение *ServerHello* подпротокола TLSHandshakeMessages подпротокола TLSRecords;

[(_sessions, 1), (_packets, 2), (_records, 2)] – фрагмент пакета, в котором размещено сообщение *ChangeCipherSpec* подпротокола TLSRecords;

[(_sessions, 1), (_packets, 2), (_records, 3), (_handshMess, 1)] – фрагмент пакета, в котором размещено сообщение *EncryptedExtensions* подпротокола TLSHandshakeMessages подпротокола TLSRecords;

[(_sessions, 1), (_packets, 2), (_records, 3), (_handshMess, 2)] – -//- CertificateRequest -//-;

[(_sessions, 1), (_packets, 2), (_records, 3), (_handshMess,3)] – -//-Certificate -//-;

[(_sessions, 1), (_packets, 2), (_records, 3), (_handshMess, 4)] – -//- CertificateVerify -//-;

[(_sessions, 1), (_packets, 2), (_records, 3), (_handshMess, 5)] – -//- Finished -//-.

В данном примере последовательность *_handshMess* разбита на три части: первая часть (состоящая из одного элемента) размещена в первом пакете, вторая (состоящая из одного элемента) и третья (состоящая из пяти элементов) – во втором.

3.2 Второе приближение C2-машины – машина *C0*

Машина *C0* – это абстракция C2-машины, достаточная для формализованного изложения предлагаемого способа специфицирования недетерминизма протоколов.

Ниже используется псевдокод, содержащий следующие интуитивно понятные конструкции:

if … then (do) … (else (do) …)	*return …*	*… && …*
case … do … case … do …	*let … = (do) …*	

Используются следующие операции со списками и другие соглашения:

[] пустой список,
a:as добавление в начало списка *as* элемента *a*,
as ++ bs конкатенация списков *as* и *bs*,
null as если список *as* пустой, то истина, иначе ложь,
length as длина списка *as*,
take n as список из первых *n* элементов списка *as*,
drop n as список *as* без первых *n* элементов,
reverse as перестановка элементов списка *as* в обратном порядке,
init as список *as* без последнего элемента,
last as последний элемент списка *as*,
fst ab левый элемент пары *ab*,
snd ab правый элемент пары *ab*,
empty пустая частичная функция,
Maybe A множество *{Just x | x :: A} ∪ {Nothing}*,
A \ B разность множеств *A* и *B*,
a :← A элемент *a* недетерминированно выбран из множества *A*,
a ← f *a* принимает значение, возвращаемое функцией *f*,
v := x обновление переменной *v* состояния машины значением *x*,
[X] множество списков конечной длины, составленных из элементов множества *X*, включая пустой список,
Map X Y множество всех частичных функций из *X* в *Y*,
Dom f область определения частичной функции *f*,
insert a b f добавление в частичную функцию *f* пары *(a, b)* (с замещением пары вида *(a, *)*, если таковая в *f* была),

f a b c f :: (A, B, C) → Y; параметры функций (в данном случае параметрами функции *f* являются *a :: A, b :: B и c :: C*) обычно будут записываться без круглых скобок и запятых – через пробелы,

N1 натуральные числа, начиная с единицы,

Bool *{True, False}*.

C0 = (State, Role, SeqID, FieldName, FieldValue, protSpec).

Состояние машины *C0* – это три переменных, значениями которых являются частичные функции:

State = {packetRoles :: Map (N1, N1) (Role, Role),
fieldValues :: Map (FieldName, SeqIndex) FieldValue,
sequenceInfo :: Map SeqID (Map SeqIndex N1)},

где *SeqID* – множество идентификаторов последовательностей, *FieldName* и *FieldValue* – множества имен и значений полей пакетов протокола, *SeqIndex = [(SeqID, N1)]* – множество всех sequence-индексов.

Для *seqid :: Dom sequenceInfo* частичная функция *sequenceInfo seqid* состоит из множества пар *(index, num)*, где *index :: SeqIndex* – это индекс фрагмента пакета, внутри которого создана часть последовательности с идентификатором *seqid*, а *num :: N1* – число элементов в этой части.

Спецификация протокола разбита на два компонента:

protSpec = {packetDef :: PacketDef, controlDef :: ControlDef}, где

PacketDef = (State, SeqID, SeqIndex) → Fragment – функция вычисления фрагментов пакетов протокола,

Fragment = Field FieldName | Sequence SeqID | B [Fragment] – (заданное в виде алгебраического типа данных) множество фрагментов пакетов (то есть фрагмент – это либо именованное поле, либо последовательность фрагментов, либо конкатенация фрагментов),

ControlDef = SeqID → ([(GuardFun, [Assignment])], ReadyFun) – функция, для каждого индекса задающая, во-первых, варианты возможных значений именованных полей фрагмента пакета с этим индексом вместе с *гардами* этих вариантов (то есть условиями, при которых они допустимы):

Assignment = (FieldName, [FieldValue]),

GuardFun = (State, SeqIndex) → Bool;

во-вторых, условие, при котором возможен выход из последовательности:

ReadyFun = (State, SeqIndex) → Bool.

Недетерминизм спецификации (specification choice) «рассредоточен» по трем местам: 1) выбор «продолжать или не продолжать последовательность» – при условии, что выход возможен, 2) выбор гарда *guard :: GuardFun* из числа тех, что обратились в истину, 3) выбор значений именованных полей из числа возможных вариантов, задаваемых списком *assignments :: [Assignment]*, стоящим в паре с этим *guard*. Ниже в псевдокоде Листинга 1 эти места выделены полужирным шрифтом.

```
runC0 = run_session 1
run_session s = do
    do_sequence _packets 1 [(_sessions,s)]
    b :← Bool
    if b then run_session (s+1) else stop

do_sequence seqid i index = do
  b ← proceed_or_drop seqid i index
  if not b then return else do
      if seqid == _packets then do
          (src,dst) :← (Role,Role)
          packetRoles := insert (s,i) (src,dst) packetRoles
```

```
      new_seqelement seqid i index
      do_sequence seqid (i+1) index
proceed_or_drop seqid i index = do
    let checkready = snd (controlDef seqid)
    let b = checkready (index ++ [(seqid,i)])
    if not b then return True else do
        c :← Bool
        if c then return True
             else return False
new_seqelement seqid i index = do
    let index' = index ++ [(seqid,i)]
    let guardedassignments = fst (controlDef seqid)
    truelists ← select_truepairs guardedassignments []
    assignment :← truelists
    assign_fieldvalues assignment
    let frag = packetDef seqid index'
    do_fragment frag
    sequenceInfo := insert (seqid,index) i sequenceInfo
select_truepairs xs acc =
  if xs == [] then return acc else do
    let (guard,assignments):xs' = xs
    if guard index'
        then select_truepairs xs' (acc ++ [assignments])
        else select_truepairs xs' acc
assign_fieldvalues xs =
  if xs == [] then return else do
      let (fieldname,variants):xs' = xs
      vnt :← variants
      fieldValues := insert (fieldname,index') vnt fieldValues
      assign_fieldvalues xs'
do_fragment frag = do
  case frag == Field fieldname do
      if (fieldname,index') :: Dom fieldValues
          then return else fail
  case frag == Sequence seqid' do
      do_sequence seqid' 1 index'
  case frag == B frags do
      do_concat frags
do_concat xs =
  if xs == [] then return else do
      let frag':xs' = xs
      do_fragment frag'
      do_concat_of xs'
```

Листинг 1. Алгоритм работы машины C0
Listing 1. Algorithm of the machine C0

4. Дальнейшие детали построения C2-машины

4.1 Инструкции и процедуры

Процедура C2-машины – это последовательность *инструкций*, исполняемых на стеке: аргументы инструкций берутся из стека и результаты выполнения инструкций кладутся в стек. Элементами стека являются байт-строки переменной конечной длины.

При исполнении инструкций используется и обновляется внутреннее состояние С2-машины. Кроме того, используется информация, снимаемая со встроенных ДСЧ и часов реального времени.

Результирующим набором байт-строковых значений инструкции (процедуры) называется содержимое стека после завершения исполнения инструкции (процедуры). Если этот набор состоит из одного элемента, то используется термин *байт-строковое значение* инструкции (процедуры).

Примеры инструкций С2-машины (с неформальным и упрощенным пояснением семантики):

C [0x15, 0x21] –положить в стек байт-строку *[0x15, 0x21]*;

V_ "ChosenSuite" [(_sessions, 1)] – если машине известен фрагмент данных с именем *"ChosenSuite"*, вычисленный в первой сессии протокола, то положить в стек этот фрагмент данных;

Random (Plain 4) Clnt (V_ "Key" [(_sessions, 3)]) – если сейчас идет третья сессия протокола и отправитель текущего формируемого пакета – роль *Clnt*, и если фрагмент данных с именем *"Key"* в третьей сессии еще не был вычислен, то снять с ДСЧ четыре случайных байта и положить их в стек, при этом считать эти байты значением фрагмента данных *"Key"* из третьей сессии;

*Length_* – взять из стека крайний элемент и положить обратно длину этого элемента в байтах (в формате big endian без лидирующих нулевых байтов);

I2Bytes_ 32 – дополнить крайний элемент стека слева лидирующими нулевыми байтами до длины 32;

*Hash_* – взять из стека два крайних элемента (идентификатор алгоритма хэширования и текст), вычислить и положить в стек хэш;

Encrypt_ 4 – взять из стека четыре крайних элемента (идентификатор алгоритма шифрования, открытый текст, ключ и инициализирующий вектор), вычислить на этих параметрах шифртекст и положить его в стек;

B_ 3 – взять из стека три крайних элемента и положить обратно их конкатенацию;

*Not_* – взять из стека крайний элемент, если он равен *[0x01]*, то положить в стек *[0x00]*, иначе положить *[0x01]*;

And_ 2 – взять из стека два крайних элемента, если они оба равны [0x01], то положить в стек *[0x01]*, иначе положить *[0x00]*;

Src Serv – если отправитель текущего пакета – роль *Serv*, то положить в стек *[0x01]*, иначе положить *[0x00]*;

Equals_ 2 – взять из стека два крайних значения и сравнить их байт-строковые значения; если равны, то положить в стек *[0x01]*, иначе - положить *[0x00]*;

P instrs – выполнить процедуру instrs.

Машине не всегда хватает информации, хранящейся во внутреннем состоянии, чтобы вычислить байт-строковое значение инструкции. Такое событие является нормальным, и существует специальная инструкция, заданная на этом событии:

Known instr – если байт-строковое значение инструкции *instr* может быть вычислено, то положить в стек *[0x01]*, иначе положить *[0x00]*.

Примечание: запись [a,b...], где a, b и т.д. – элементы некоторого множества, обозначает список из этих элементов.

Язык инструкций С2-машины не поддерживает циклы или рекурсии. Есть поддержка ветвлений посредством инструкции *Select instr cases*.

Пример:

Select (V_ "ChosenSuite" [(_sessions,1)]) [([0x13,0x01], ins1), ([0x13,0x02], ins2), ...] – если байт-строковое значение инструкции *V_ "ChosenSuite" [(_sessions,1)]* равно *[0x13,0x01]*, то выполняется инструкция *ins1*, если *[0x13,0x02]*, то *ins2* и т.д.

Последовательности фрагментов внутри пакетов создаются с помощью инструкции *Sequence seqid seqelem*, где *seqid* – идентификатор последовательности, а *seqelem* – процедура вычисления очередного элемента последовательности.

Ввиду наличия таких «макроинструкций», как *P instrs*, *Select instr cases* и др., термины «инструкция» и «процедура» C2-машины являются взаимозаменяемыми.

4.2 Выражения C2-машины

Контекст выполнения процедуры – это sequence-индекс фрагмента пакета, вычисляемого этой процедурой. В зависимости от контекста, одна и та же процедура C2-машины может иметь разные байт-строковые значения.

Выражения – это формулы, конструируемые C2-машиной в ходе выполнения процедур, с помощью которых она запоминает результаты вычислений с учетом контекста выполнения.

Далее *Word8* – множество целых чисел в диапазоне *0..255*, *String* – множество символьных строк.

Выражения и вычисленные байт-строковые значения, сопоставленные им, хранятся в следующих переменных состояния C2-машины:

expressions :: Map ExprDesc (CoreInstr, [ExprDesc]), где *CoreInstr* – множество базовых (не зависящих от контекста) инструкций C2-машины, *ExprDesc* – множество дескрипторов выражений;

values :: Map ExprDesc [Word8].

Таким образом, выражение – это пара *(coreinstr, ds)*, где *coreinstr* – базовая инструкция, а *ds* – список дескрипторов выражений параметров.

Примеры выражений:

(V_ "ChosenSuite" [(_sessions, 1), (_packets, 2)], [])

(C [0x15, 0x21], [])

(Encrypt_ 4, [34, 53, 57, 60])

Пример инструкции машины, зависящей от контекста: *Vi str*, где *str :: String*. Встретив инструкцию *Vi str* в процедуре, выполняемой в контексте *ix*, машина заменяет *Vi str* на *V_ str ix*.

В C2-машину в неизменном виде переносится хранилище *sequenceInfo*, заданное нами выше для машины *C0*.

Механизм построения выражений и сохранения их вместе с байт-строковыми значениями обеспечивает C2-машине следующие свойства:

1) машина никогда не дублирует вычисления: сначала она строит выражение, соответствующее процедуре, потом проверяет, известно ли уже байт-строковое значение этого выражения; если да, то выполнение процедуры на этом заканчивается;
2) одинаковые процедуры в одинаковом контексте выполнения дают один и тот же результат, несмотря на случайность генерируемых ключей;
3) появляется возможность иметь следующие важные инструкции (с учетом информации о последовательностях, накопленной в *sequenceInfo*):

   LastIn seqid instr filtercondition – найти последний по порядку индекс, оканчивающийся (в левой части пары) на *seqid*, такой что в этом контексте инструкция *filtercondition*, выступающая в роли предиката, истинна, и вычислить в этом контексте инструкцию *instr (примечание: если байт-строковое значение инструкции равно [0x01], то считается, что эта инструкция обращается в истину, иначе – в ложь)*;

AllIn seqid instr filtercondition – записать по порядку в список все индексы, оканчивающиеся на *seqid*, такие что в их контексте инструкция-предикат *filtercondition* истинна, и вычислить в контексте каждого из них инструкцию *instr*; результирующие байт-строковые значения положить в стек согласно порядку отобранных sequence-индексов.

В C2-машине в отдельной переменной состояния *curstep :: (N1, N1)* хранится пара *(s, p)*, где *s* – номер текущей сессии, *p* – номер текущего пакета. От значения *curstep* зависит результат выполнения инструкций *InCurrSess* и *InCurrPack* (ниже считаем, что начальный отрезок текущего контекста имеет вид *[(_sessions, s'), (_packets, p'), …]*):

InCurrSess если *s' == s*, то результат *[0x01]*, иначе *[0x00*]

InCurrPack если *p' == p*, то результат *[0x01]*, иначе *[0x00*]

4.3 Программа C2-машины (спецификация протокола)

Двухкомпонентная спецификация, определенная для машины *C0*, в C2-машине детализируется следующим образом:

packetDef :: Instr;

controlDef :: Map SeqID [((Instr, [(Instr, Instr)]), Instr)].

Здесь *Instr* – это множество всех инструкций C2-машины.

В структуре *((guard, assignments), ready) = controlDef seqid* компоненты *guard* и *ready* – это инструкции-предикаты. Элемент *assignments* – это список, состоящий из пар вида *(instr, values)*, где *instr* – инструкция с (пока что) неизвестным байт-строковым значением (обычно это инструкция *Vi str*), *values* – инструкция, результирующий набор байт-строковых значений которой составляет множество возможных значений для *instr*.

Пример: элемент списка *assignments*, определенный как

```
(P [LastIn _handshMess (P [Vi "HandshType", C [0x01], Equals_ 2]) (Src Clnt),
    Known (LastIn _handshMess (P [Vi "HandshType", C [0x02], Equals_ 2]) (Src Serv)),
    Not_,
    And_ 2],
 [(Vi "HandshType",  C [0x02]),
  (Vi "ChosenSuite",  P [C [0x13, 0x01], C [0x13, 0x02]])]),
```

означает следующее: если в последнем (на текущий момент) элементе последовательности *_handshMess* из пакета, отправленного ролью *Clnt*, фрагмент данных *"HandshType"* имел значение *[0x01]*, и ранее в *_handshMess* не существовало элемента из пакета, отправленного ролью *Serv*, в котором фрагмент данных *"HandshType"* имел значение *[0x02]*, то в текущем формируемом элементе последовательности *_handshMess* присвоить фрагменту данных *"HandshType"* значение *[0x02]*, а фрагменту данных *"ChosenSuite"* присвоить значение, недетерминированным образом выбранное из двух вариантов – *[0x13, 0x01]* и *[0x13, 0x02]*.

Как было показано на уровне абстракции машины *C0*, C2-машина совершает недетерминированный выбор (в части specification choice) тогда и только тогда, когда наступает момент формирования очередного элемента последовательности. Резюмируем поведение машины в этот момент для инструкции *Sequence seqid seqelem*:

1) перед формированием очередного элемента последовательности последовательно перебираются все пары *(guard, assignments)*, относящиеся к *seqid*, и в этих парах вычисляются гарды; далее из числа тех гардов, что обратились в истину, недетерминированно выбирается один;
2) обрабатывается список *assignments*, стоящий в паре с выбранным гардом (здесь также совершается недетерминированный выбор значений); если спецификация корректна, то после обработки этого списка процедура *seqelem* становится детерминированно

вычислимой: всем нужным инструкциям, не имевшим байт-строковых значений, эти значения назначены;

3) выполняется процедура *seqelem*, вычисляющая очередной элемент последовательности *seqid*.

4.4 Третье приближение С2-машины – машина *C1*

Особенности функционирования С2-машины, рассмотренные в предыдущих параграфах раздела, опишем в формализованном виде в виде машины *C1*.

C1 = (State, Instr, N1, Word8, String, Bool, SeqID, argsnum, calcfun, protSpec)
State = {curstep :: (N1,N1),
packetRoles :: Map (N1,N1) (Role, Role),
expressions :: Map Desc (CoreInstr, [ExprDesc]),
values :: Map Desc BS,
sequenceInfo :: Map SeqID (Map SeqIndex N1)}
SeqIndex = [(SeqID, N1)]
Instr ::= CoreInstr | Vi String | V1 String | V2 String | Known Instr | P Instrs |
Src Role | Dst Role | InCurrSess | InCurrPack |
Sequence SeqID Instr | Select Instr [([Word8], Instr)] | B [Instr]
CoreInstr ::= C [Word8] | V_ String SeqIndex | Encrypt_ N1 | Hash_ | Length_ |
B_ N1 | Known_ SeqIndex | Equals_ N1 | Not_ | And_ N1 | ...
SeqID ::= _sessions | _packets | _records | ...
argsnum :: CoreInstr → N1
calcfun :: (CoreInstr, [[Word8]]) → [Word8]
protSpec :: (Instr, Map SeqID [(Instr, [(Instr, Instr)])])

Определения вспомогательных функций процедуры функционирования машины *C1*:

dochoices :: (SeqID, SeqIndex) → Bool
symexecute :: ([Instr], SeqIndex, [ExprDesc]) → [ExprDesc]
calcvalue :: ExprDesc → [Word8]

Реализация вспомогательных функций процедуры функционирования машины *C1* и Алгоритм работы машины *C1* показаны в Листингах 2 и 3 соответственно.

```
dochoices seqid ix = do
  let filtguards ac xs =
    if null xs then return ac
      else do
        let (guard,assignments):xs' = xs
        [d] ← symexecute [B [guard]] ix []
        bs ← calcvalue d
        if bs == [0x01]
          then filtguards (ac ++ [assignments]) xs'
          else filtguards ac xs'
  let doassignments xs =
    if null xs then do
          let info = if seqid :: Dom sequenceInfo
            then sequenceInfo(seqid) else empty
          let info' = insert (init ix) (snd (last ix)) info
          sequenceInfo := insert seqid info' sequenceInfo
          return True
      else do
        let (ins,values):xs' = xs
        [d] ← symexecute [B [ins]] ix []
```

```
        if d :: Dom values then fail
          else do
            ds ← symexecute [values] ix []
            d' :← ds
            mbs ← calcvalue d'
            if mbs == Just bs
              then do
                  values := insert d bs values
                  doassignments xs'
              else fail
  let controlDef = snd protSpec
  updlist ← filtguards [] (controlDef(seqid))
  if null updlist then return False
    else do assignments' :← updlist
                doassignments assignments'
findcase bs cases =
  if null cases then fail else do
    let (cs,ins1):cases' = cases
    if cs == bs then symexecute [B [ins1]] ix []
                     else findcase cases'
doexpress coreins ix stk = do
  let push d stk = return (d:stk)
  let pops n stk =
      if length stk < n then fail
                                  else return (take n stk, drop n stk)
  (ds, stk') ← pops (argsnum coreins) stk
  let expr = (coreins, reverse ds)
  if d :: Dom expressions && expressions(d) == expr
    then push d stk'
    else do let e :: DescExpr \ Dom expressions
                expressions := insert e expr expressions
                push e stk'
symexecute instrs ix stk = do
  if null instrs then return stk else do
    let chkbool expr = if expr then C [0x01] else C [0x00]
    let [(_sessions,s),(_packets,p)] = take 2 ix
    let ins:inss = instrs
    case ins :: CoreInstr do
        stk' ← doexpress ins ix stk
        symexecute inss ix stk'
    case ins == P inss' do (symexecute (inss' ++ inss) ix stk)
    case ins == Src r do chkbool (fst (packetRoles(s,p)) == r)
    case ins == Dst r do chkbool (snd (packetRoles(s,p)) == r)
    case ins == InCurrSess do chkbool (fst curstep == s)
    case ins == InCurrPack do chkbool (snd curstep == p)
    case ins == B inss' do
        ds ← symexecute inss' ix []
        d ← symexecute [B_ (length ds)] ix []
        symexecute inss ix (d:ds ++ stk)
    case ins == Vi str do (symexecute [V_ str ix] ix stk)
    case ins == V1 str do (symexecute [V_ str (take 1 ix)] ix stk)
    case ins == V2 str do (symexecute [V_ str (take 2 ix)] ix stk)
    case ins == Known ins' do
        symexecute [B [ins'], Known_ ix] ix stk
    case ins == Select ins' cases do
        [d] ← symexecute [B [ins']] ix []
        mbs ← calcvalue d
```

```
        case mbs == Just bs do [e] ← findcase bs cases
                                                symexecute inss ix (e:stk)
        case mbs == Nothing do fail
    case ins == Sequence seqid ins1 do
      let addelem i stk' = do
        let ix' = ix ++ [(seqid,i)]
        b ← dochoices seqid ix'
        if b then do [d] ← symexecute [B [ins1]] ix'
                           addelem (i+1) (d:stk')
             else symexecute inss ix stk
      addelem 1 []
calcvalue d =
  let mapMcalcdesc zs = do
      let calc ac xs =
        if null xs then return (Just ac)
          else do let x:xs' == xs
                  if calcdesc x == Just y
                      then calc (ac ++ [y]) xs'
                      else return Nothing
     calc [] zs
  if d :: Dom values then return (Just (values(d)))
    else do let (coreins,ds) = calcexpress (expressions(d))
                if mapMcalcdesc ds == Just bss
                    then do let bs = calcfun coreins bss
                                values := insert d bs values
                                return (Just bs)
                    else return Nothing
```

Листинг 2. Реализация вспомогательных функций процедуры функционирования машины C1
Listing 2. Implementation of the auxiliary functions of the procedure for the functioning of the C1 machine

```
     runC1 = do
               let packetDef = fst protSpec
            let run (s, p) = do
                     let chksess = do b' :← Bool
                                 if b' then run (s, p+1) else run (s+1, p)
                curstep := (s, p)
                (src, dst) :← (Role, Role)
                packetRoles := insert (s, p) (src, dst) packetRoles
                let ix = [(_sessions, s), (_packets, p)]
                b ← dochoices _packets ix
                if b then do [d] ← symexecute [B [packetDef]] ix
                                   mbs ← calcvalue d
                                 if mbs == Just bs then chksess else fail
                      else run (s+1, p)
            packetRoles := empty
            run (1,1)
```

Листинг 3. Алгоритм работы машины C1
Listing 3. Algorithm of the C1 machine

Последовательность троек *(src, dst, bs)*, снимаемых внешним наблюдателем на каждой итерации функции *run*, называется *трассой запуска машины C1*.

5. Декларативное представление процедур C2-машины

Так как C2-машина является стековой машиной, процедуры вычисления формул представляют собой запись этих формул в обратной «польской» нотации. Например, формулу *EncryptAES128 (plaintext ||HashSHA256 (plaintext), key, iv)*, где *plaintext, key, iv* –

некоторые байт-строки, можно вычислить следующей процедурой C2-машины: *[C iv, C key, C plaintext, C sha256oid, Hash_, C plaintext, B_ 2, C aes128oid, Encrypt_ 4]*, где *sha256oid, aes128oid* – это байт-строки-идентификаторы алгоритмов.

Если добавить в язык инструкций C2-машины «синтаксический сахар», в частности, записывая формулы в привычном инфиксном виде, то процедуры C2-машины приобретают удобный для восприятия декларативный вид. Например, описанную выше процедуру можно переписать следующим образом:

Encrypt (C aes128oid) [text, C key, C iv] where

text = B [C plaintext, Hash (C sha256oid) (C plaintext)]

Существуют плохо читаемые процедуры, часто встречающиеся в разных протоколах. Их можно вынести в библиотеки стандартных процедур. Например, процедура *withLen n instr* приписывает слева к результирующей байт-строке инструкции *instr* длину этой байт-строки, уложенную в *n* байтов в порядке big endian:

withLen n instr = B [B [instr], Length_, I2Bytes_ n, instr]

Весь «синтаксический сахар» языка C2-машины, ввиду ограничений на объем текста, в настоящей статье не приводится. Конечный результат применения приемов по улучшению читаемости показан в Листинге 4.

```
packetDef = Sequence _records tlsRecord
  where
  tlsRecord =
   B [Vi "ContentType" # OfLen 1,
        Vi "Version"         # OfLen 2,
        withLen 2
         (Select (Vi "ContentType")
           [Case [0x14] changeCipherSpecMsg,
            Case [0x15] alertMsg,
            Case [0x16] handshakeMsgs,
            Case [0x17] encryptedMsg])]
        where
        changeCipherSpecMsg = C [0x01]
        handshakeMsgs =
          Sequence _handshMess handshMsg
            where
            handshMsg = B
             B [Vi "HandshType" # OfLen 1,
                  withLen 3 (Vi "HandshakeMsg" ##
                    Select (Vi "HandshType")
                       [Case [0x01] clientHello,
                        Case [0x02] serverHello,
                        Case [0x05] endOfEarlyData,
                        Case [0x08] encrypExtensions,
                        Case [0x0d] certificateRequest,
                        Case [0x0b] certificate,
                        Case [0x0f] certificateVerify,
                        Case [0x14] finished,
                        Case [0x04] newSessionTicket,
                        Case [0x18] keyUpdate])]

clientHello =
 B [C [0x03,0x03],
      random Clnt,
      withLen 1 clientSessionID,
      withLen 2 clientSuites,
      withLen 1 (C [0x00]),
      withLen 2
```

```
(Sequence _clientHelloExts extension)]
 where
 extension =
  B [Vi "ClientExtensionID" # OfLen 2,
       withLen 2 (Vi "ClientHelloExtension" ##
          Select (Vi "ClientExtensionID")
             [Case [0x00,0x00] serverName,
              Case [0x00,0x0a] groups,
              Case [0x00,0x0b] pointFormats,
              Case [0x00,0x23] sessionTicket,
              Case [0x00,0x16] encryptThenMAC,
              Case [0x00,0x17] extendedMaster,
              Case [0x00,0x0d] supportedSigAlgs,
              Case [0x00,0x2b] supportedVersions,
              Case [0x00,0x31] postHandshAuth,
              Case [0x00,0x33] keyShare,
              Case [0x00,0x2a] earlyData,
              Case [0x00,0x2d] pskKeyExchMode,
              Case [0x00,0x29] preSharedKey]]
         where
         serverName = withLen 2
             (B [Vi "NameType" # OfLen 1,
                 …
```

Листинг 4. Начальная часть спецификации пакета протокола TLS версии 1.3
Listing 4. An initial part of the specification for a TLS protocol package version 1.3

Такая нотация, являясь по-настоящему формальной (исполнимой на некоторой машине) по восприятию не сильно отличается от нотации формализованных вставок, используемых в документах RFC 5246 (TLS версии 1.2) и RFC 8446 (TLS версии 1.3), – тот же декларативный и лаконичный стиль описания форматов сообщений и определений функций.

Пример дальнейшей формализации RFC 8446: тексту на естественном языке *"Many of the cryptographic computations in TLS make use of a transcript hash. This value is computed by hashing the concatenation of each included handshake message, including the handshake message header carrying the handshake message type and length fields, but not including record layer headers. I.e., Transcript-Hash(M1, M2, ... Mn) = Hash(M1 || M2 || ... || Mn).»* будет соответствовать следующая формальная запись на языке С2-машины:

```
Hash hashOID (AllIn  _handshMess
                     (B  [Vi "HandshType",  withLen 3 (Vi "HandshakeMsg")])
                     filtercondition)
```

где *filtercondition* – это инструкция-предикат, которая, в свою очередь, формально описывает условия отбора сообщений *M1, M2, ..., Mn*, неформально заданные в RFC 8446.

6. Тестирование «черного ящика» и криптографическая машина с исключенной ролью

Спецификация протокола задана на одной машине, совместно используемой всеми ролями протокола. Запуская спецификацию на С2-машине, разработчики промышленных реализаций протокола могут генерировать эталонные трассы с интересующими их параметрами и использовать их в качестве контрольных примеров.

Если же рассматривать задачу тестирования «черного ящика», то тестируемая реализация протокола – это отдельный объект. Тестирующей системе не известны секретные ключи объекта тестирования и результаты разрешения недетерминизма протокола на стороне объекта тестирования. Как следствие, предварительная генерация из спецификации некоторого множества трасс в целях проверки конформности «черного ящика» не имеет смысла. Тесты должны генерироваться динамически с учетом реакции «черного ящика».

C2-машина с исключенной ролью role (далее – C2ER-машина) – это вариант C2-машины, который работает в паре с «черным ящиком», реализующим протокол на стороне роли *role*.

C2ER-машина должна уметь решать две задачи: 1) проверять конформность пакетов, поступающих от «черного ящика» (message verification); 2) по пакетам, поступающим от «черного ящика», актуализировать свое состояние так, чтобы сохранялась конформность последующего поведения C2ER-машины (state updating).

Процедура функционирования *runC1ER*, описывающая C2ER-машину во втором приближении (Листинг 5), отличается от процедуры *runC1* наличием двух дополнительных функций – *receivefrom* (получение пакета извне) и *verifyupdate* (верификация пакета и актуализация состояния C2ER-машины; в случае успеха функция возвращает *True*, иначе *False*).

```
receivefrom :: Role → Maybe [Word8]
verifyupdate :: (ExprDesc, [Word8]) → Bool
runC1ER role = do
          let packetDef = fst protSpec
    let run (s, p) = do
               let chksess = do b' ←: Bool
                                if b' then run (s, p+1) else run (s+1, p)
          curstep := (s, p)
          (src, dst) :← (Role, Role)
          packetRoles := insert (s, p) (src, dst) packetRoles
               let ix = [(_sessions, s), (_packets, p)]
          if src == role
              then do mbs ← receivefrom role
              if mbs == Just bs
                       then do [d] ← symexecute [B [packetDef]] ix
                                          b ← verifyupdate d bs
                                          if b then chksess else fail
                                  else stop
              else do b ← dochoices _packets ix
                   if b then do [d] ← symexecute [B [packetDef]] ix
                                                 mbs ← calcvalue d
                                  if mbs == Just bs then chksess else fail
                                    else run (s+1, p)
    packetRoles := empty
    run (1,1)
```

Листинг 5. Процедура функционирования runC1ER
Listing 5. RunC1ER Functional Procedure

Реализация функции *verifyupdate* – непростая задача. Например, в работе [16] продемонстрирована нетривиальность задачи применения диссекторов общего назначения (на примере фреймворка Python Scapy) для разбора пакетов криптографических протоколов (на примере QUIC).

C2ER-машину будем называть *корректной проекцией* C2-машины, если выполняется следующее условие: для любой спецификации протокола множество трасс, производимых C2ER-машиной, является подмножеством множества трасс, производимых C2-машиной.

7. Заключение

Для проверки работоспособности предложенного подхода автором разработана реализация C2-машины с исключенной ролью на языке Haskell и спецификации протокола TLS (версий 1.2 и 1.3). Спецификация существенной части протокола TLS 1.3 (в частности, поддерживается двусторонняя аутентификация и восстановление сессий с использованием тикетов) на языке C2-машины занимает около 800 строк (часть *packetDef* – 500 строк и часть *controlDef* – 300 строк). Еще 200 строк (обработка сертификатов и др. общеупотребимые

процедуры) вынесены из *packetDef* в библиотеку, общую для всех протоколов. В роли «черного ящика» выступал программный пакет openssl.

Применительно к задаче тестирования «черного ящика» реализация машины умеет делать полный обход всех возможных траекторий протокола, зависящих от тестирующей стороны, последовательно перебирая различные варианты разрешения ситуаций недетерминированного выбора (в предположении, что объект тестирования детерминирован и для него возможен *надежный рестарт*, reliable reset).

На данном этапе реализации подхода ограничение перебора предполагается осуществлять путем ручного сокращения недетерминизма спецификации. Реализация эффективных стратегий обхода на полноценной спецификации – тема дальнейших исследований. Цель – на базе формализма С2-машины задать метрики качества тестирования криптопротоколов, схожие с метриками, принятыми в стандартах тестирования программ в авионике. Возможные подходы к решению этой задачи рассмотрены, например, в работе [17] (в контексте *машины с абстрактными состояниями* (ASM) [18]). В следующей работе также планируется представить результаты моделирования на языке С2-машины репрезентативного, с точки зрения обоснования выразительности предложенного языка, набора промышленных криптографических протоколов (IPSec (IKEv2), QUIC и др.).

Дополнительно авторская реализация С2-машины с исключенной ролью включает реализацию алгоритма Д. Англуин (Dana Angluin) [19], с помощью которого полученное дерево обхода сворачивается в более компактное (удобное для визуального анализа) представление в виде конечного автомата, а также транслятор трасс запуска С2-машины в спецификации в нотации анализатора протоколов ProVerif.

Литература

[1] А.В. Никешин, Н.В. Пакулин, В.З. Шнитман. Тестирование реализаций клиента протокола TLS. Труды ИСП РАН, том 27, вып.2, 2015 г., стр. 145-160 / A. V. Nikeshin, N. V. Pakulin, V. Z. Shnitman. TLS clients testing. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 2, 2015, pp. 145-160 (in Russian), DOI: 10.15514/ISPRAS-2015-27(2)-9.

[2] K. Bhargavan, A. Delignat-Lavaud et al. Implementing and Proving the TLS 1.3 Record Layer. In Proc. of the 38th IEEE Symposium on Security and Privacy, 2017, pp.463-482.

[3] J. Goubault-Larrecq and F. Parrenne. Cryptographic protocol analysis on real C code. Lecture Notes in Computer Science, vol. 3385, 2005, pp. 363-79.

[4] J. Bozic, L. Marsso et al. A formal TLS handshake model in LNT. In Proc. of the Third Workshop on Models for Formal Analysis of Real Systems and Sixth International Workshop on Verification and Program Transformation, 2018, pp. 1-40.

[5] K. Bhargavan, B. Blanchet, N. Kobeissi. Verified models and reference implementations for the TLS 1.3 standard candidate. In Proc. of the IEEE Symposium on Security and Privacy; 2017, pp.483-502.

[6] J. Zhang, L. Yang et al. Formal analysis of 5G EAP-TLS authentication protocol using ProVerif. IEEE Access, vol. 8, 2020, pp. 23674-23688.

[7] M. Backes, A. Busenius, C. Hriţcu. On the Development and Formalization of an Extensible Code Generator for Real Life Security Protocols. Lecture Notes in Computer Science, vol. 7226, 2-12, pp. 371-387.

[8] M. Aizatulin, A. Gordon, J. Jurjens. Extracting and verifying cryptographic models from C protocol code by symbolic execution. In Proc. of the 18th ACM Conference on Computer and Communications Security (CCS'11), 2011, pp. 331-340.

[9] P. Arcaini, A. Gargantini, E. Riccobene. AsmetaSMV: a way to link high-level ASM models to low-level NuSMV specifications. Lecture Notes in Computer Science, vol. 5977, 2010, pp. 61-74.

[10] J. Jürjens. Automated Security Verification for Crypto Protocol Implementations: Verifying the Jessie Project. Electronic Notes in Theoretical Computer Science, vol. 250, issue 1, 2009, pp. 123-136.

[11] I. Abdullah, D. Menascé. Protocol Specification and Automatic Implementation Using XML and CBSE. In Proc. of the International Conference on Communications, Internet and Information Technology. (CIIT2003), 2003, pp. 1-7.

[12] E.W. Dijkstra. Guarded commands, non-determinacy and a calculus for the derivation of programs. Communications of the ACM, vol. 18, issue 8, 1975, pp. 453-457.
[13] K.L. McMillan and L.D. Zuck. Formal specification and testing of QUIC. In Proc. of ACM Special Interest Group on Data Communication (SIGCOMM'19), 2019, 14 p.
[14] D. Rozenzweig, D. Runje. The Cryptographic Abstract Machine. Lecture Notes in Computer Science, vol. 3052, 2004, pp. 202-217.
[15] M. Veanes, C. Campbell et al. Model-Based Testing of Object-Oriented Reactive Systems with Spec Explorer. Lecture Notes in Computer Science, 2008, vol. 4949, pp. 39-76.
[16] E. Gagliardi, O. Levillain. Analysis of QUIC session establishment and its implementations. In Proc. of the IFIP International Conference on Information Security Theory and Practice, 2019, pp.169-184.
[17] A. Gargantini, E. Riccobene. ASM-based Testing: Coverage Criteria and Automatic Test Sequence Generation. Journal of Universal Computer Science, vol. 7, no. 11, 2001, pp. 1050-1067.
[18] Y. Gurevich. Sequential Abstract State Machines Capture Sequential Algorithms. ACM Transactions on Computational Logic, vol. 1, issue 1, 2000, pp. 77-111.
[19] D. Angluin. Learning Regular Sets from Queries and Counterexamples. Information and Computation, vol. 75, issue 2, 1987, pp. 87-106.

Информация об авторах / Information about authors

Сергей Евгеньевич ПРОКОПЬЕВ – кандидат технических наук, старший научный сотрудник отдела технологий программирования ИСП РАН, руководитель направления сетевых протоколов безопасности лаборатории сетевой и информационной безопасности АО НПК «Криптонит».

Sergey Evgenievich PROKOPEV – PhD, research associate in the Department of Programming Technologies of the ISP RAS, senior researcher in the area of security protocols in the Laboratory of the Network Security of JSRPC «NPK Kryptonite».

DOI: 10.15514/ISPRAS-2021-33(5)-8

CC BY

Разработка компилятора для стековой процессорной архитектуры TF16 на основе LLVM

Л.В. Скворцов, ORCID: 0000-0002-1580-1244 <lvs@ispras.ru>
Р.В. Баев, ORCID: 0000-0002-7999-7952 <baev@ispras.ru>
К.Ю. Долгорукова, ORCID: 0000-0002-8220-8328 <unerkannt@ispras.ru>
Е.Ю. Шарыгин, ORCID: 0000-0002-4042-3302 <eush@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация: Разработка для стековых процессорных архитектур обычно ведётся с использованием устаревших низкоуровневых языков программирования или языка ассемблер. Поэтому актуальна задача поддержки языков программирования высокого уровня для таких архитектур. В этой работе мы рассматриваем процесс разработки и реализации на базе инфраструктуры LLVM/Clang полноценной системы программирования для языка Си для стековой архитектуры TF16. Использование именно LLVM в качестве базовой системы программирования обусловлено большими возможностями адаптации дополнительных компонентов системы программирования, например, таких как дизассемблер, компоновщик и отладчик для использования с новыми архитектурами. Нами были разработаны две версии компилятора. В первой версии компилятора архитектура TF16 рассматривалась как классическая регистровая архитектура, и сгенерированный код не использовал стековые возможности. Эта версия была относительно проста в разработке и служила точкой сравнения для второй версии компилятора. Во второй версии компилятора был разработан и реализован платформо-независимый алгоритм планирования команд с учётом особенностей стековых архитектур. При сравнении двух версий версия компилятора с поддержкой стековых возможностей генерирует код, который в среднем на 35.7% быстрее по времени выполнения и на 50.8% меньше по размеру, чем код, генерируемый версией компилятора без поддержки стековых возможностей. Разработанный алгоритм позволяет реализовать в компиляторе LLVM поддержку других стековых процессорных архитектур.

Ключевые слова: стековый процессор; компилятор; LLVM

Для цитирования: Скворцов Л.В., Баев Р.В., Долгорукова К.Ю., Шарыгин Е.Ю. Разработка компилятора для стековой процессорной архитектуры TF16 на основе LLVM. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 137-154. DOI: 10.15514/ISPRAS–2021–33(5)–8

Developing an LLVM-based compiler for stack based TF16 processor architecture

L.V. Skvortsov, ORCID: 0000-0002-1580-1244 <lvs@ispras.ru>
R.V. Baev, ORCID: 0000-0002-7999-7952 <baev@ispras.ru>
K.Y. Dolgorukova, ORCID: 0000-0002-8220-8328 <unerkannt@ispras.ru>
E.Y. Sharygin, ORCID: 0000-0002-4042-3302 <eush@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract: Development for stack-based architectures is usually done using legacy low level languages or assembly code, so there exists a problem of a high level programming language support for such architectures. In this paper we describe the development process of an LLVM/Clang-based C compiler for stack-based TF16

processor architecture. LLVM was used due to adaptation possibilities of its components for new architectures, such as disassembler, linker and debugger. Two compiler versions were developed. The first version generated code without using stack capabilities of TF16, treating it instead as a register-based architecture. This version was relatively easy to develop and it provided us a comparison point for the second one. In the second version we have implemented a platform independent stack scheduling algorithm that allowed us to generate code that makes use of the stack capabilities of the CPU. When comparing the two versions, a version that utilized stack capabilities generated code that was on average 35.7% faster and 50.8% smaller than the original version. The developed stack scheduling algorithm also allows to support other stack based architectures in LLVM toolchain.

Keywords: stack processor; compiler; LLVM

For citation: Skvortsov L.V., Baev R.V., Dolgorukova K.Y., Sharygin E.Y. Developing an LLVM-based compiler for stack based TF16 processor architecture. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 137-154 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–8

1. Введение

Несмотря на то, что в настоящее время среди процессоров для встраиваемых устройств доминируют различные варианты RISC-архитектур, процессоры со стековой архитектурой всё ещё находят применение. Помимо вопросов совместимости и переиспользования проверенного временем кода в составе сложных систем, это в том числе связано с тем, что стековые процессоры удовлетворяют основным требованиям встраиваемых устройств, таким как надёжность работы, размер и вес микроконтроллера и энергопотребление [1].

Зачастую для стековых архитектур нет современных средств для разработки программного обеспечения, а для программирования используется ассемблер либо применяются специализированные стековые языки.

Одной из таких архитектур является TF16 [2]. Программирование для микроконтроллеров этой архитектуры выполняется исключительно на языке программирования FORTH, для которого реализован транслятор в машинные коды TF16.

В современном мире язык FORTH [3] имеет очень слабое распространение, поэтому актуальна задача создания системы программирования для широко распространенных языков высокого уровня для архитектуры TF16.

Целью данной работы является разработка и реализация на базе инфраструктуры LLVM/Clang [4] полноценной системы программирования для языка Си и платформы TF16. Использование именно LLVM в качестве базовой системы программирования обусловлено большими возможностями адаптации дополнительных компонентов системы программирования, например, таких как дизассемблер, компоновщик, и отладчик для использования с новыми архитектурами.

Существует несколько реализаций компилятора языка Си для стековых архитектур. Так, в работе [5] описывается модуль для GCC [6], оптимизирующий код под стековую архитектуру в рамках одной функции. В работе [7] этот подход расширяется до межпроцедурного уровня. Также существует реализация глобального планирования инструкций стека для компилятора LCC [8]. Однако все эти реализации были разработаны под чистые стековые архитектуры, т.е. архитектуры без регистров общего назначения, не являющихся частью стековой модели. TF16 не является такой архитектурой, поэтому для неё такой подход не годится.

Дальнейшее изложение построено следующим образом. В разд. 2 описываются особенности архитектуры TF16. Разд. 3 описывает реализацию базовой поддержки этой архитектуры в компиляторе LLVM и первого соглашения о вызовах. В разд. 4 обсуждается разработка поддержки для использования стековой модели вычислений вместо регистровой, что включает в себя создание стекового планировщика, упорядочивающего команды из промежуточного внутреннего представления DAG, которое описывает вычисления в виде дерева зависимостей по данным, и реализацию второго соглашения о вызовах. В разд. 5 рассматриваются реализации эмулятора и системы регрессионного тестирования для

платформы TF16. В Разд. 6 приведены результаты тестирования реализованного компилятора для различных наборов тестов. Разд. 7 содержит заключение.

2. Архитектура TF16

TF16 – 16-битная архитектура. Микроконтроллеры TF16 используют 16-битные регистры. При этом единицей любого обращения к памяти является 16-битная ячейка, то есть адрес обращения будет четным. Если передать нечетный адрес – младший бит будет проигнорирован. Для того, чтобы адресуемое пространство не было ограничено 64 Кб, используются четыре 6-битных сегментных регистра. Таким образом, с помощью сегментной адресации можно адресовать пространство объемом 4 Мб.

По классификации Купмана (Philip J. Koopman) [1], архитектура TF16 ближе всего к модели множества стеков с малым размером буфера и инструкциями с одним операндом (MS1). Это означает наличие нескольких стеков (в случае TF16 – двух), расположенных в общей памяти, элементы которых при необходимости можно адресовать напрямую, используя их адрес в общей памяти. Также это означает, что инструкции в архитектуре TF16 явно принимают один аргумент и неявно используют как второй аргумент элемент на вершине стека. Следует заметить, что не все инструкции TF16 попадают под это правило. Подробнее инструкции рассматриваются в соответствующем разделе.

2.1 Регистры процессора TF16

При работе стекового процессора TF16 используется два стека – стек данных и стек адресов возврата. Для указания вершин этих стеков используются регистры PSP и RSP соответственно; оба регистра содержат по 16 разрядов. В регистре хранится именно адрес верхнего элемента стека, а не указатель на следующую свободную ячейку. В стек данных помимо информации из памяти логически встраиваются регистры T и N, описанные ниже. В стеке возвратов такой надстройки нет, верхний элемент будет находиться по адресу [RSP]. Направление роста для обоих стеков определяется младшим битом регистра-указателя. При установленном младшем бите (значение 1) соответствующий стек «растет» в направлении уменьшения адресов.

Сегментные регистры CS и SS имеют особое назначение – первый из них указывает на сегмент кода, откуда производится выборка команд; второй регистр – SS – указывает на стековый сегмент. Остальные два сегментных регистра DS и ES могут произвольным образом использоваться для адресации данных в любых сегментах.

Регистр T является 16-разрядным аккумулятором, а также виртуальной вершиной стека данных. Каждая инструкция производит чтение или запись этого регистра. Регистр N также 16-разрядный, и является расширением аккумулятора, или вторым сверху элементом на стеке данных. Последующие элементы стека хранятся в памяти.

Существует еще 4 регистра, которые можно назвать регистрами общего назначения. Они называются A1, A2, A3 и A4 и также содержат 16 бит. Они могут использоваться как базовые при обращении к памяти.

Для организации некоторых видов циклов есть специальный 16-разрядный регистр счетчика цикла C.

Существует также одноразрядный регистр флага CARRY, который обычно устанавливается в 1 при наличии переполнения во время выполнения последней арифметической команды.

В качестве счётчика команд используется 15-разрядный регистр PC. Адресом текущей команды является значение регистра PC, логически сдвинутое влево на один разряд.

Дополнительные регистры, отвечающие за работу системы прерываний, мы не будем рассматривать в рамках данной работы.

Процессор TF16 также поддерживает добавление к нему периферийных устройств. В этой работе мы будем рассматривать только одно такое устройство – сопроцессор для вещественных вычислений. В этом сопроцессоре содержатся 8 16-битных регистров общего назначения R1-R8, объединённых в 32-битные регистры OP1-OP4, где R1 << 16 | R2 = OP1, R3 << 16 | R4 = OP2 и т.д. На регистрах OP1-OP4 следует располагать 32-битные числа с плавающей запятой.

2.2 Система команд

Команды процессора TF16, как и у других стековых процессоров, очень плотно упакованы, и за одну команду позволяют выполнить несколько непротиворечивых действий одновременно. Существуют 16-битные и 32-битные коды команд. Последние занимают два последовательных 16-битных слова, и второе слово является 16-разрядным непосредственным операндом.

К некоторым командам можно добавить суффиксы, что добавляет количество эффектов от выполнения этих команд. Список суффиксов с кратким описанием их работы будет приведён ниже.

Все команды процессора можно разделить на несколько классов.

- Команды работы со стеком.
  - Команда PPUSH добавляет на стек данных значение, находящееся в регистре T. В терминах процессора TF16 это означает, что произойдёт инкремент (или декремент) регистра PSP, затем значение, находящееся в регистре N, будет записано в память по адресу [PSP], а в регистр N будет записано значение, находящееся в регистре T.
  - Команда PPOP снимает со стека данных верхнее значение. В терминах процессора TF16 это означает, что значение регистра N будет записано в регистр T, значение, находящееся в памяти по адресу [PSP], будет записано в регистр N, и произойдёт декремент (или инкремент) регистра PSP.
  - Для работы со стеком возвратов используются команды RINC, RDEC, RMOVA, RLOAD, RPUSH и RPOP. Команды RINC и RDEC осуществляют инкремент и декремент значения регистра RSP в зависимости от значения младшего бита этого регистра. Команды RMOVA и RLOAD осуществляют чтение и запись значения на вершине стека возвратов соответственно. В первом случае значение с вершины стека возвратов будет записано в регистр T, во втором значение в регистре T будет записано на вершину стека возвратов. Команды RPUSH и RPOP эквивалентны парам команд RINC+RLOAD и RMOVA+RDEC соответственно.
- Команды чтения и записи регистров.
  - Команда LOAD записывает значение регистра T в регистр, переданных в качестве явного операнда. К этой команде можно добавить суффиксы POP, POPNT, PUSH, RET.
  - Команда MOVA записывает значение регистра, переданного в качестве явного операнда, в регистр T. К этой команде можно добавить суффиксы POP, PUSH и RET. Также в качестве операнда можно передать непосредственное целое значение. При этом если непосредственный операнд находится в диапазоне от 0 до 255, такую команду можно закодировать двумя способами, как 16-битную с 8-битным непосредственным операндом, так и как 32-битную с 16-битным операндом.
  - Команда XCHG меняет местами значение регистра, переданного в качестве явного операнда, и значение регистра T. К этой команде можно добавить суффикс(ы POP, PUSH,) RET.
  - Все три команды принимают в качестве явного операнда любой из регистров A1-A4,

N, C, CS, SS, DS, ES, PSP, RSP.

- Арифметические и логические команды
  - ADD, ADDC, SUB, SUBB, SBB, SBBB, AND, OR и XOR.
  - Одним из операндов в этих командах всегда выступает регистр T, а в качестве второго может использоваться как любой из регистров набора A1-A4, N, C, так и непосредственное значение. При этом если непосредственный операнд находится в диапазоне от 0 до 255, такую команду можно закодировать двумя способами, как 16-битную с 8-битным непосредственным операндом, так и как 32-битную с 16-битным операндом. Операции сложения и вычитания всегда устанавливают флаг переполнения CARRY согласно обычным правилам работы с беззнаковыми вычислениями. Команда ADDC отличается от ADD тем, что при вычислении суммы дополнительно прибавит значение CARRY. Аналогично SUBB отличается от SUB, значение CARRY будет дополнительным вычитаемым. В командах SBB и SBBB реализовано вычитание со сменой операндов – в них регистр T является вычитаемым. Результат вычислений всегда будет расположен в регистре T.
  - Если в качестве явного операнда используется регистр, то к этим командам можно добавить один из суффиксов POP, RUSH, RET или REG!.
- Команды перехода.
  - Команда NOP переходит на следующую команду без побочных эффектов. К этой команде можно добавить суффикс POP. Результатом получившейся команды NOP POP будет сброс второго элемента (регистра N) со стека данных с сохранением вершины стека (регистра T) без изменений.
  - Команды BR, BP осуществляют безусловный переход. Операндом является знаковое смещение относительно текущего значения регистра PC. Поскольку обе команды 16-разрядные, смещение должно умещаться в 8 бит. В случае команды BP также со стека данных сбрасывается верхний элемент.
  - Команда JUMP осуществляет безусловный переход по передаваемому в качестве операнда абсолютному 16-битному адресу.
  - Команды BZ, BNZ, BLZ, BGE, BZP, BNZP, BLZP, BGEP, BC, BNC, BCP, BNCP осуществляют условный переход аналогично командам BR и BP. Для последних четырёх команд условие перехода зависит от значения регистра CARRY, для остальных - от значения регистра T.
  - Команда CYCL осуществляет условный переход аналогично команде BR. При переходе команда также понижает значение счётчика цикла (регистр C).
  - Команды CALL и RETURN осуществляют вызов функций и возврат из функций соответственно. В терминах процессора TF16 это означает, что команда CALL принимает в качестве операнда 16-разрядный адрес перехода аналогично команде JUMP и кладёт на вершину стека возвратов адрес возврата. Команда RETURN снимает с вершины стека возвратов адрес возврата и осуществляет безусловный переход по этому адресу.
- Команды сдвига.
  - DSRL, DSRA, DSLL, DSRC, DSLC – команды двойного сдвига. Осуществляют сдвиг 32-битного целого числа, находящегося в регистрах N и T, где в регистре T находятся верхние 16 бит числа, в регистре N – нижние 16 бит числа. К этим командам можно добавить суффикс RET.
  - SRL, SRA, SLL, SRC, SLC – команды одинарного сдвига. Осуществляют сдвиг 16-битного целого числа, находящегося в регистре T. Также существует команда SWAB

– циклический сдвиг целого числа в регистре T на 8 разрядов (можно рассматривать как перестановку верхних и нижних 8 байт числа). К этим командам можно добавить суффиксы PUSH, POP и RET.

- Все команды сдвига, кроме команды SWAB, осуществляют сдвиг на один разряд и не принимают явных операндов.

- Команды работы с памятью.
  - Эти команды принимают 2-3 явных операнда. Команды строятся следующим образом:
  - Ключевое слово LOAD или MOVA – запись в памяти и чтение из памяти соответственно
  - Регистр общего назначения A1-A4 или ключевое слово MEM. В первом случае запись будет означать, что в качестве адреса используется значение, находящееся в регистре общего назначения. При этом к регистру можно добавить прединкремент, преддекремент, постинкремент или постдекремент. Во втором случае в качестве адреса используется значение, находящееся на регистре T.
  - Сегментный регистр CS, SS, DS или ES.
  - Ключ -n или -t. Этот ключ указывает на то, из какого регистра берётся значение для записи в память, либо в какой регистр будет загружено значение из памяти. Используются только регистры N и T.

Как было отмечено выше, к некоторым командам можно добавить следующие суффиксы:

- Суффикс POP. После вычисления результата второй элемент стека данных, находящийся в регистре N, будет сброшен со стека.
- Суффикс POPNT. Используется только с командой LOAD. Аналогично паре команд LOAD <REG> + PPOP.
- Суффикс PUSH. Помимо вычисления результата перед записью в регистр производится команда PPUSH.
- Суффикс RET. После вычисления результата и записи в регистр будет выполнен возврат из текущей функции.
- Суффикс REG!. Старое значение регистра T будет записано в регистр, использованный в качестве явного операнда инструкции.

Помимо этого, несмотря на традиционный для стековых машин отказ от использования конвейера команд, в TF16 он присутствует. Наиболее очевидно это проявляется в поведении конструкции LOAD CS POPNT; RETURN. Будучи выполненной последовательно, первая команда в этой конструкции записывает новое значение в сегментный регистр, означающий сегмент кода, затем новая команда должна быть считана уже из нового сегмента. Однако, в реальности команда RETURN будет считана ещё со старым значением сегментного регистра CS, а переход по адресу возврата уже будет использовать новое значение. В разд. 4 будет показано, как именно используется эта особенность.

3. Нестековый компилятор

На первом этапе разработки стояла задача реализации базовой версии компилятора, используя встроенные возможности LLVM. Поскольку LLVM поддерживает только классические регистровые архитектуры, то на этом этапе процессор рассматривался как регистровый процессор с регистрами общего назначения A1-A4 и C. Кроме того, регистры T и N не рассматривались как вершина стека, поэтому в рамках этого раздела вершиной стека будет называться элемент по адресу [RSP]. Регистр N тоже считался регистром общего назначения.

Рассмотрим общую схему работы LLVM на примере функции foo, осуществляющей сложение двух 32-битных чисел и возврат результата. Функция указана на рис. 1.

```
int32_t foo(int32_t x, int32_t y) { return x+y; }
```

Рис. 1. Тестовая функция
Fig. 1. Test function

Рассматривать будем оптимизированный код на уровне -O1. LLVM работает с кодом на языке LLVM IR, для получения которого из кода на языке C используется фронтэнд Clang.

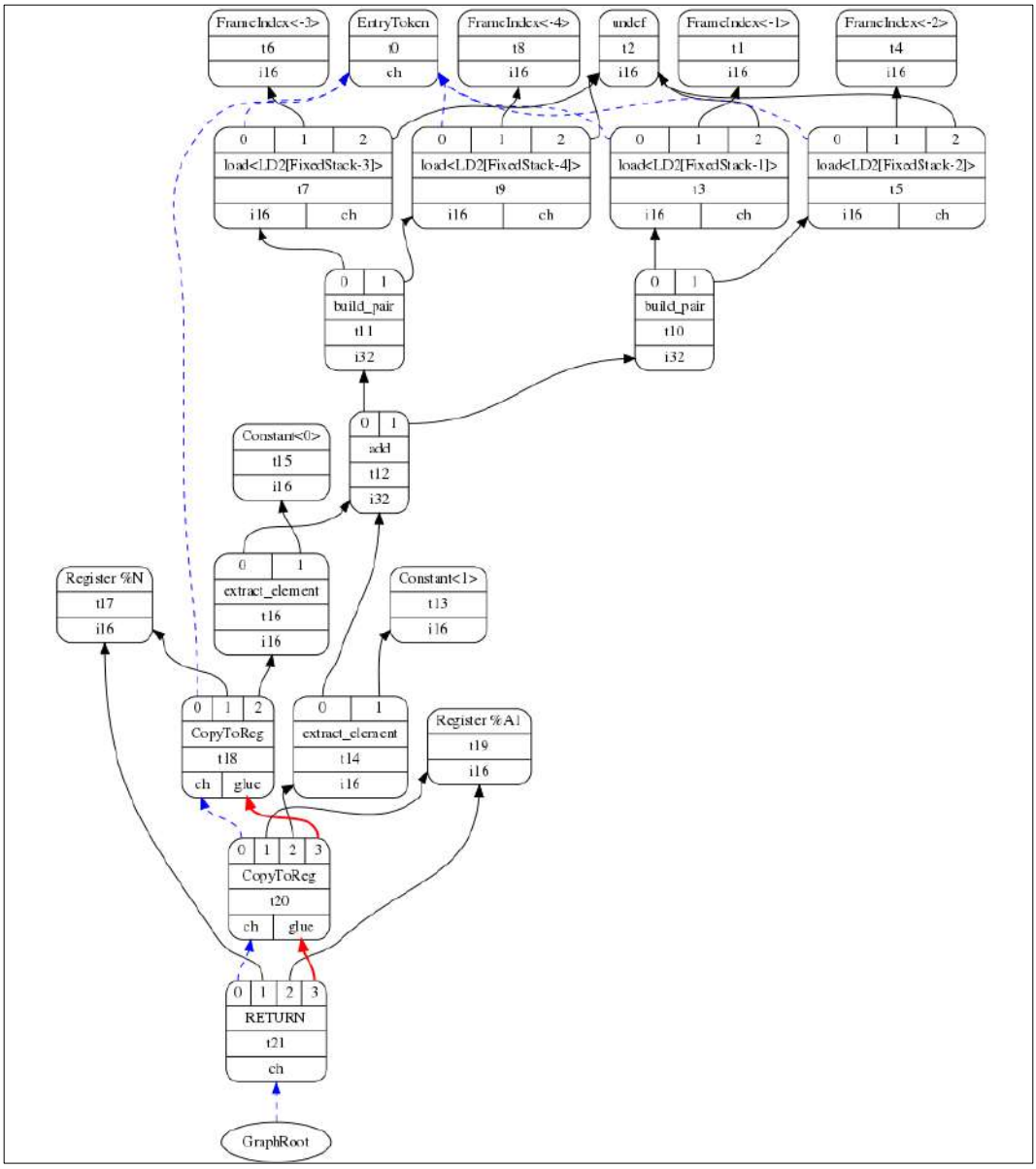


Рис. 2. Граф внутреннего представления для функции foo
Fig. 2. DAG for "foo" function

По этому коду строится машинно-независимое внутреннее представление в виде направленного ациклического графа (DAG). Начальный граф для функции foo представлен на рис. 2. Каждый узел графа представляет из себя структуру, включающую в себя операцию, выполняемую узлом, значения результата и указатели на операнды. На рисунке указание на операнды обозначено чёрной стрелкой. Результатом работы инструкции могут быть как целочисленные и вещественные значения, так и специальные значения для реализации так называемых «сцепленных» инструкций. На рис.2 сцепленные инструкции соединяются синей пунктирной стрелкой. Если у инструкции есть некоторый побочный эффект (например, чтение/запись), то она обязана принимать цепную связь в качестве первого операнда и возвращать в качестве последнего результата. Помимо цепной связи порядок выполнения инструкции может быть указан через связь, называемую склейкой или склеенными инструкциями. Такая связь означает, что между двумя склеенными инструкциями не может быть запланирована на выполнение никакая другая инструкция. На рисунке эта связь указана красной стрелкой.

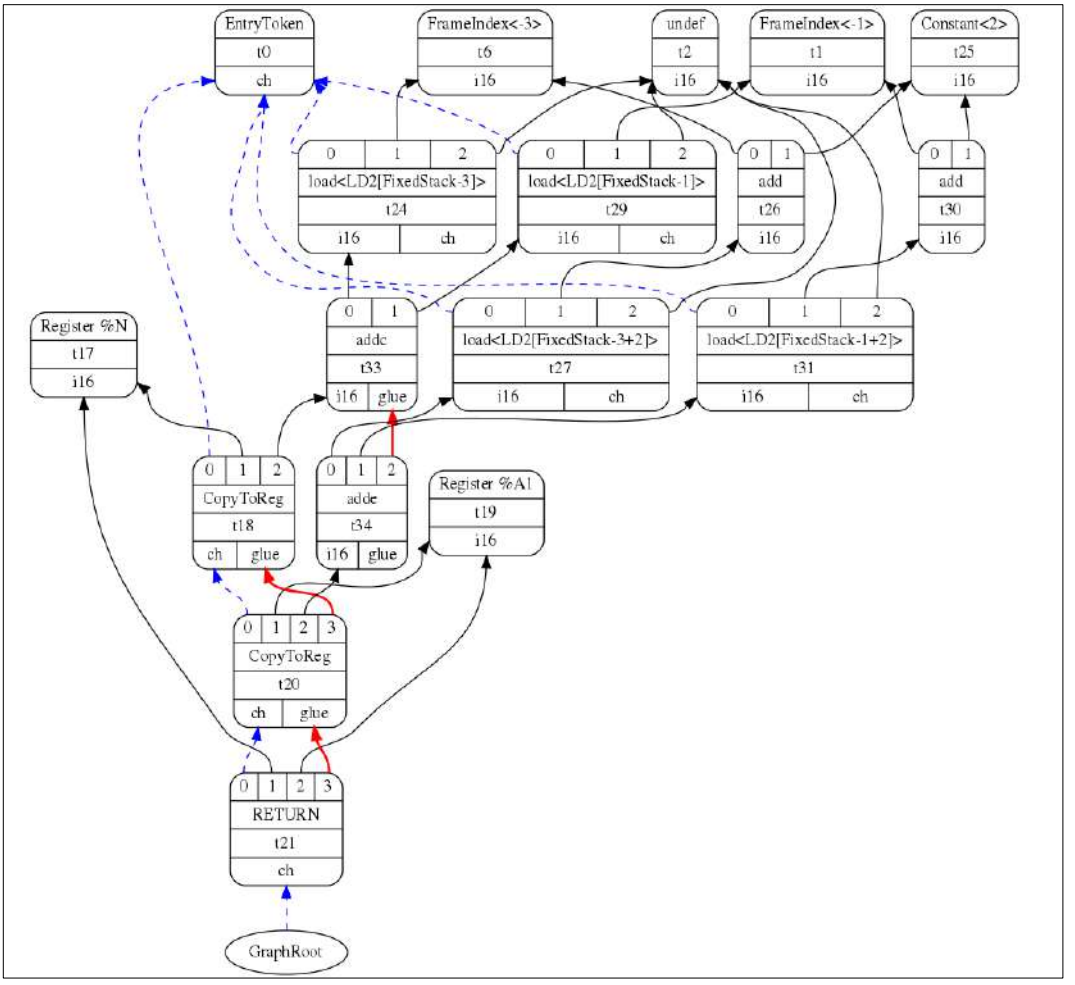


Рис. 3. Оптимизированный граф внутреннего представления для функции foo
Fig. 3. Optimized DAG for "foo" function

Начальное представление проходит через несколько преобразований с целью оптимизации и легализации под выбранную архитектуру. Легализация означает, что в представлении не должно использоваться типов данных и операций, которые не поддерживаются выбранной

архитектурой. Так, легализованный и оптимизированный граф для функции foo для архитектуры TF16 указан на рис. 3. На этом шаге преобразований стояла одна из основных задач для поддержки архитектуры. Для легализации графа необходимо, чтобы каждой инструкции соответствовало некоторое преобразование в последовательность инструкций, использующих только поддерживаемые платформой операции и типы данных. Часть таких преобразований доступны по умолчанию, например, на рис. 4 видно, что 32-битная инструкция add преобразована в последовательность инструкций addc и adde, осуществляющих сложение с установкой переполнения и добавлением переполнения соответственно. Однако, некоторые инструкции преобразуются слишком неоптимально для 16-битной платформы (например, select), или преобразование такой инструкции не предусмотрено в LLVM (32-битные сдвиги). Поэтому для поддержки TF16 были реализованы недостающие преобразования.

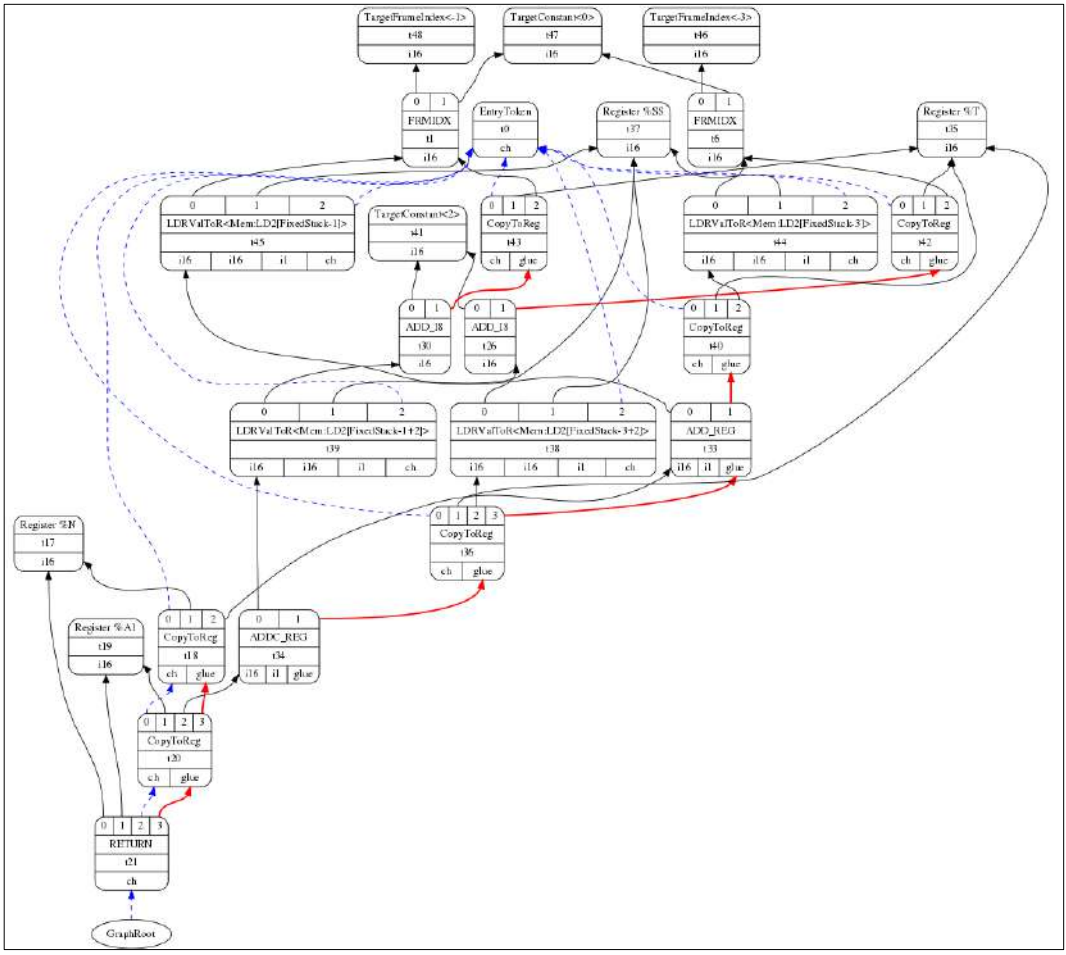


Рис. 4. Граф с инструкциями целевой архитектуры для функции foo
Fig. 4. Target Instruction DAG for "foo" function

На основе легализированного DAG происходит выбор инструкций целевой архитектуры и планирование порядка их выполнения. Пример DAG с выбранным инструкциями для TF16 показан на рис. 4.

Запланированный DAG преобразуется в последовательность машинных инструкций и псевдоинструкций – конструкций, выполняющих некоторое вычисление, которые позже заменяются на последовательность команд процессора. На этапе выбора машинных

инструкций происходит большинство оптимизаций, распределение регистров и раскрытие псевдоинструкций. В конце работы этапа остаётся представление, в котором каждой машинной инструкции соответствует команда процессора. Из этого представления генерируется исполняемый код.

Для хотя бы частичной поддержки возможностей инструкций, выполняющих несколько действий одновременно, был разработан оптимизирующий проход слияния инструкций. В этом проходе машинные инструкции заменяются на аналогичные по результату выполнения, но занимающие меньше памяти или процессорного времени.

Стоит обратить отдельное внимание на поддержку вызовов функций. На первом этапе было принято решение использовать стек данных для передачи параметров функции и регистры N, A1-A3 — для возврата значений из функции. Параметры располагались на стеке в порядке, при котором в момент вызова функции на вершине стека находился первый параметр. Возвращаемое значение было расположено по принципу, напоминающему little endian, т.е. в N лежала самая младшая часть, в A1 – старшая часть 32-битного числа, в A2-A3 – старшая половина 64-битного числа. Фрейм функции создавался на стеке данных, а обращение к элементам фрейма происходило относительно регистра A4.

Для линковки исполняемых файлов также была реализована поддержка архитектуры TF16 в линкере LLD. Для корректного распределения кода и данных в памяти используется специальный линкер-скрипт, а для начальной настройки сегментных регистров – начальный файл begin.s.

4. Стековый компилятор

В процессе реализации нестековой версии компилятора почти не были использованы стековые возможности процессора. Между тем, при правильной организации расположения данных на стеке использование этих возможностей могло бы дать более эффективный код. В связи с этим вторым этапом стал переход от регистровой модели к стековой модели.

4.1. Планирование команд

Основной задачей на данном этапе являлась разработка алгоритма планирования команд для стековой архитектуры. Было необходимо выстроить порядок команд таким образом, чтобы минимизировать количество обращений к данным вне стека во время работы программы. В классических стековых архитектурах планирование команд, по сути, заменяет собой этап распределения регистров.

Для архитектуры TF16, имеющей регистры общего назначения, а также отдельный регистровый файл для команд с плавающей точкой, используется гибридный подход. Распределение регистров общего назначения происходит на этапе выбора машинных инструкций, а планировщик старается как можно больше данных уложить в стек. Как пример будем рассматривать указанную выше функцию сложения foo. На рис. 5 указан начальный граф для функции в стековой версии компилятора.

Можно заметить, что этот граф слегка отличается от начального графа в нестековой версии компилятора. Это отличие связано с изменённой конвенцией вызовов функций. Как и в нестековой версии, аргументы вызова функции передаются через стек данных, причём первый аргумент находится на вершине стека. Однако, в отличие от нестековой версии компилятора, возвращаемые значения тоже передаются через стек данных, а не через регистры. На вершине стека при этом находится старшая часть возвращаемого значения. Фрейм функции создаётся на стеке возвратов, а обращение к элементам фрейма идёт относительно регистра RSP. В случае использования массивов переменной длины эта функция переходит регистру A4. Также, вместо явных загрузок значений из фрейма функции используются загрузки из виртуальных регистров. В стековой модели эти загрузки будут преобразованы в инструкции взятия значения со стека.

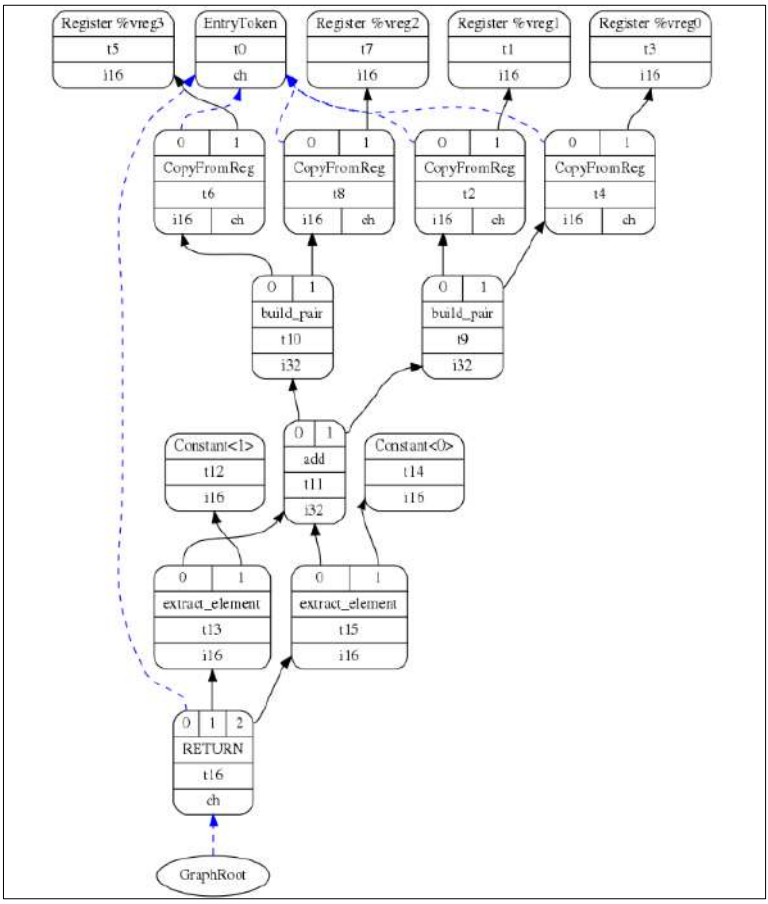


Рис. 5. Граф внутреннего представления для функции foo в стековой версии компилятора
Fig. 5. DAG for "foo" function in the stack scheduling compiler version

Граф после этапа легализации указан на рис. 6. Помимо уже обозначенных выше отличий можно заметить отсутствие вычислений смещения относительно индекса фрейма для получения верхней и нижней половин 32-битного числа.

Граф после этапа выбора инструкций представлен на рис. 7. Поскольку на этом этапе стековая и нестековая версии компилятора перестают быть похожими, проводить прямое сравнение графов не имеет смысла.

Спецификация архитектуры TF16 предполагает фиксированный порядок расположения операндов на стеке при вызове, а также возвращаемые значения в определённых регистрах, и эти перемещения мы также отражаем при создании DAG на этапе понижения внутреннего представления (lowering) как зависимости chain, если действия обязательно должны быть выполнены до или после инструкции, но неважно, насколько рано, или glue, когда действия должны быть выполнены строго перед или после инструкции.

Разработанный алгоритм планирования команд работает в два шага. На первом шаге происходит планирование команд в DAG-представлении. На втором шаге происходит генерация машинных инструкций на основе запланированного списка команд.

Схематично работа основной функции первого шага изображена на рис. 8. Первый шаг обходит DAG в глубину, начиная с корневого узла графа (последняя инструкция, имеющая побочный эффект). Для рассматриваемого примера это команда возврата из функции. На каждой итерации рассматривается вся последовательность инструкций, имеющих с

инструкцией входного узла зависимость glue. Такие “склеенные” цепочки не могут быть разъединены, поэтому обрабатываются как единое целое. Так, команда AddePseudo будет рассматриваться вместе с командой AddcPseudo. Если таких связей нет, то последовательность будет состоять только из инструкции входного узла.

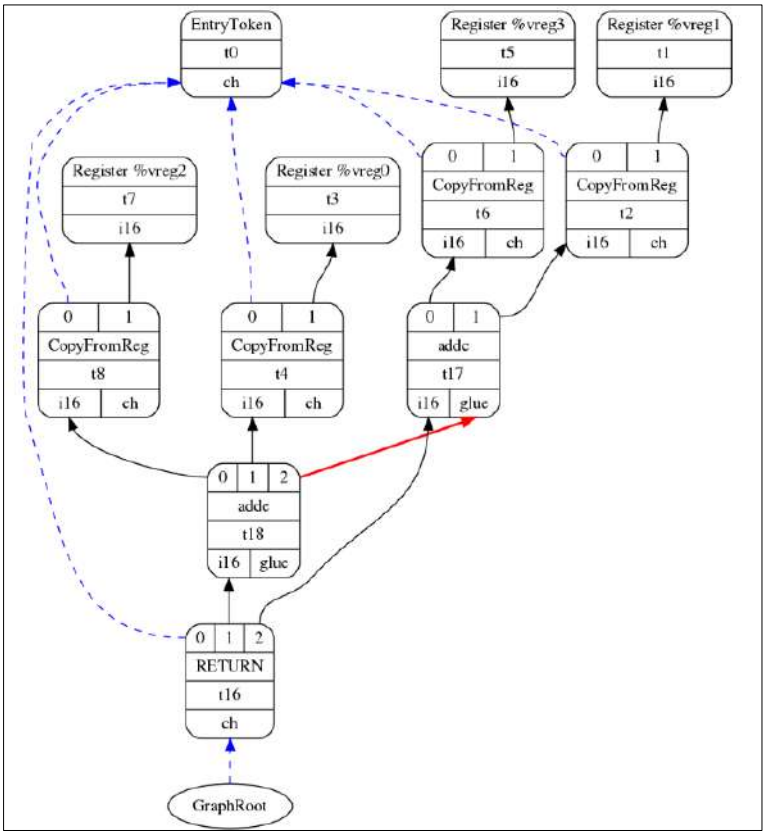


Рис. 6. Оптимизированный граф внутреннего представления для функции foo в стековой версии компилятора

Fig. 6. Optimized DAG for “foo” function in the stack scheduling compiler version

Сначала для текущей последовательности рассматриваются узлы, связанные с её элементами зависимостью chain. Для этих узлов проверяется, имеет ли смысл планировать их рано. Раннее планирование используется для минимизации возможного использования регистров для временного хранения результата. Критерием раннего планирования является отсутствие узлов, связанных chain зависимостью с рассматриваемым узлом и использующих значения результата этого узла – то есть цепочка является независимой от рассматриваемого узла и может быть выполнена отдельно.

Если обнаружились узлы, для которых раннее планирование имеет смысл, то они планируются. Здесь и далее под «планируется» имеется в виду рекурсивный вызов основной функции алгоритма для планируемого узла.

Затем для каждого элемента последовательности планируются его операнды. Последовательность при этом обходится с конца в обратном порядке. Также в обратном порядке обходятся операнды каждой инструкции в последовательности.

Затем планируются узлы, связанные с элементами последовательности через chain значения, для которых не имело смысла раннее планирование.

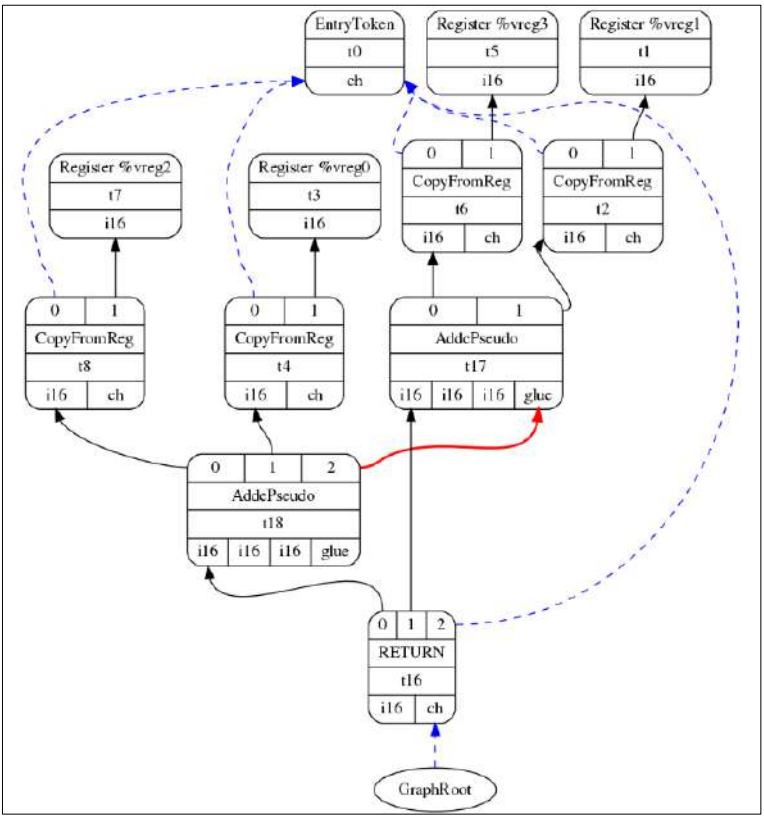


Рис. 7. Граф с инструкциями целевой архитектуры для функции foo в стековой версии компилятора
Fig. 7. Target Instruction DAG for "foo" function in the stack scheduling compiler version.

Наконец, элементы рассматриваемой последовательности добавляются в запланированную последовательность. Входная последовательность здесь рассматривается в прямом порядке. Таким образом получается порядок инструкций, при котором для каждой инструкции на момент начала её выполнения её операнды будут находиться на вершине стека. Вместе с этим рассматриваются результирующие значения инструкций. Если одно из результирующих значений инструкции используется больше, чем один раз, то оно сохраняется в виртуальный регистр. Если результирующее значение инструкции не используется, то оно сбрасывается со стека. Также при необходимости в запланированную последовательность вставляются инструкции, дублирующие результат на стеке или меняющие порядок результатов на стеке – например, если результат является операндом другой инструкции, но вычисляется не последним, то есть не остаётся на вершине стека. Подобные манипуляции со стеком на этапе планирования предоставляют гибкость для последующих оптимизаций, позволяя оставлять стековые операции, где это возможно, и убирать сохранения в регистры и загрузки из них.

Вторым шагом алгоритма является выбор машинных инструкций на основе запланированной последовательности. Этот шаг в целом прямолинеен. Если текущий элемент последовательности соответствует машинной инструкции, то строится соответствующая машинная инструкция и её операнды. Если нет, то генерируются специальные инструкции в зависимости от типа элемента. Такие конструкции генерируются для копирований из регистра, копирований в регистр, ассемблерных вставок, границ интервалов жизни, а также команд сброса со стека, дублирования стека, и смены порядка элементов на вершине стека.

Следует отметить, что разработанный алгоритм работает на высоком уровне абстракции, поэтому подходит для реализации генерации стекового кода не только для архитектуры TF16, но и для других стековых архитектур.

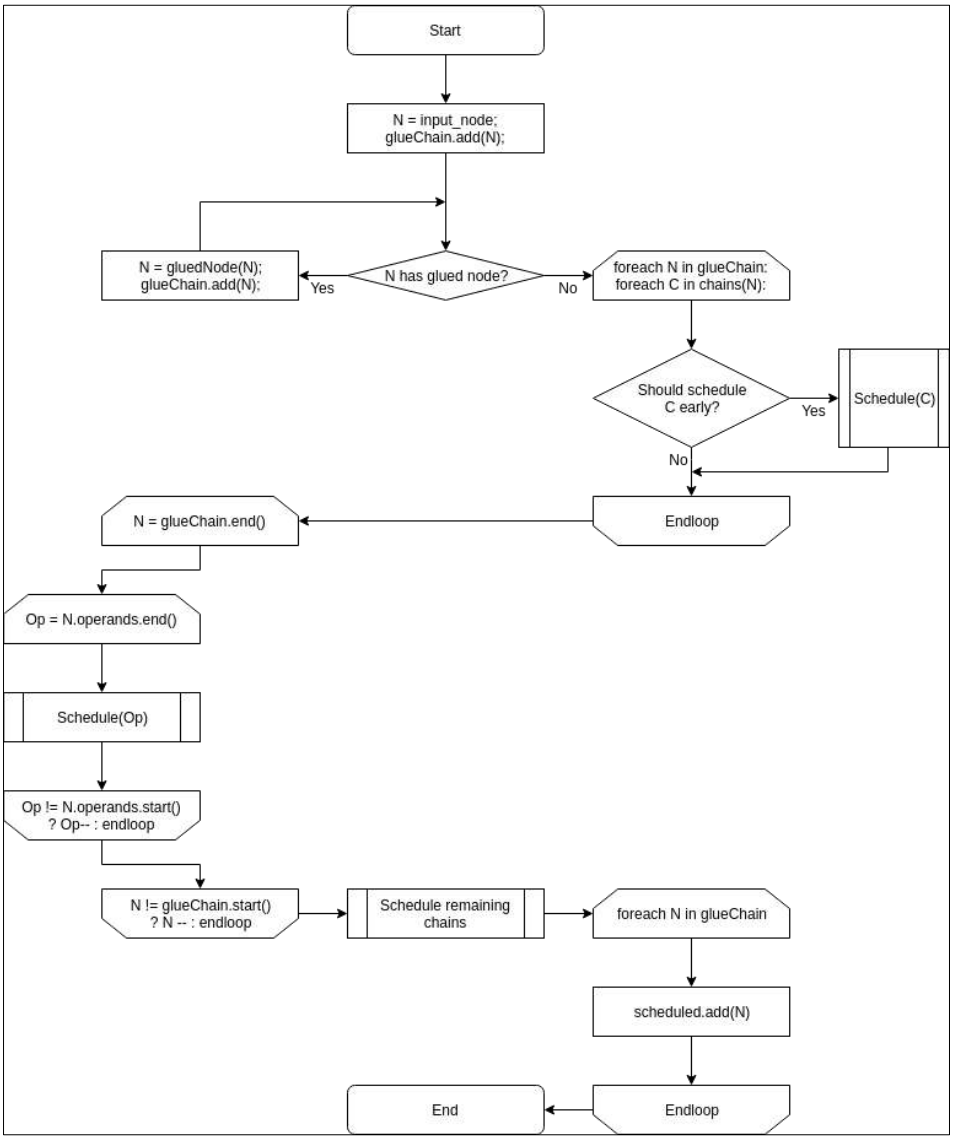


Рис. 8. Схема работы алгоритма планирования команд в стековой версии компилятора
Fig. 8. Stack scheduling algorithm scheme for stack scheduling compiler version

4.2. Адаптация под стековое выполнение

Помимо создания алгоритма планирования команд, для создания стекового компилятора также было необходимо адаптировать некоторые моменты в генерации кода. Так, в стековом компиляторе при раскрытии псевдоинструкций используется правило сохранения баланса стека. Поскольку предполагается, что машинные инструкции принимают аргументы со стека данных и сохраняют результат на стеке данных, то каждая псевдоинструкция должна раскрываться таким образом, чтобы было верно равенство $PSP_{pre} - OpSize + ResSize \equiv PSP_{post}$, где OpSize – размер операндов псевдоинструкции на стеке, ResSize – размер

результата псевдоинструкции на стеке, а PSP_{pre} и PSP_{post} – значения регистра PSP до выполнения псевдоинструкции и после соответственно.

Побочным эффектом необходимости сохранять баланс стека стало увеличение количества инструкций, делающих что-либо со стеком, причём зачастую аннулирующих результаты друг друга. Поскольку операции со стеком относительно дорогие – на работу с памятью уходит как минимум на один такт процессорного времени больше, чем на вычисления на регистрах - для снижения действий со стеком был доработан оптимизирующий проход слияния инструкций.

4.3. Адресные пространства

Одним из побочных эффектов перехода на стековую модель выполнения стало повышенное использование общей памяти для работы программ. Процессор TF16 поддерживает сегментную адресацию памяти, поэтому необходимо было реализовать генерацию кода для работы с несколькими сегментами.

На раннем этапе разработки было принято соглашение, что по умолчанию код будет располагаться в сегменте 0, а данные – в сегменте 1. Через линкер-скрипт и начальный файл begin.s можно было расположить данные и код в других сегментах.

Поддержка адресации через сегментные регистры была основана на системе адресных пространств в Clang и LLVM. Следует отметить, что разработка велась в версии LLVM 4.0.1, в которой поддержка адресных пространств была реализована не до конца, поэтому в процессе разработки поддержка адресных пространств была обратно портирована из более поздней версии LLVM.

Для архитектуры TF16 были введены несколько ключевых слов, указывающих на то, в каком сегменте будет находиться глобальный объект.

Также была разработана поддержка так называемых дальних вызовов, т.е. вызовов функций, находящихся в другом сегменте. Для этого была реализована вспомогательная функция дальнего вызова, принимающая номер сегмента и адрес вызываемой функции, сохраняющая текущий номер сегмента на стеке возвратов и делающая непрямой дальний вызов через конструкцию LOAD CS POPNT; RETURN, особенность работы которой описывается в разд. 2. На возврате из дальней функции используется аналогичная конструкция.

5. Разработка дополнительных инструментов

С целью отладки генерируемого кода на раннем этапе разработки на основе существующего в LLVM дизассемблера был разработан программный эмулятор для процессора TF16. Для этого в дизассемблере была реализована поддержка расшифровки машинных кодов процессора. Эмулятор выполняет программу по инструкциям и возвращает значение на вершине стека в момент завершения функции main в качестве кода возврата. Эмулятор также считает время выполнения программы в тактах процессора и размер исполняемого кода программы.

С использованием программного эмулятора и входящего в состав LLVM инструмента тестирования lit была разработана система для регрессионного тестирования разрабатываемого компилятора.

6. Тестирование

Поскольку под архитектуру TF16 не существовало других компиляторов языка Си, провести сравнение результатов возможно только для нестековой и стековой версий разработанного нами компилятора. Следует отметить, что во время разработки стекового компилятора нестековая версия поддерживалась путём обратного портирования из стековой версии новых разработок, не завязанных на стековой модели выполнения программ. Однако при разработке

не стояла задача обратного портирования всех новых разработок, поэтому в стековой версии возможно присутствие оптимизаций, которые можно было бы обратно портировать в нестековую версию, но по тем или иным соображениям этого сделано не было.

Также следует отметить несколько моментов, связанных с системой тестирования.

1) В нестековом компиляторе не поддерживаются уровни оптимизации выше, чем O1.
2) Поскольку для возвращения результата тесты полагаются исключительно на код возврата, при включении уровня оптимизации O1 часто возникает ситуация, когда код функции main был оптимизирован в подобие return 0. Тесты, попадающие под этот случай, объединены в общий тест return-0.c.
3) Наборы поддерживаемых библиотечных функций сильно отличаются между версиями, как отличаются и их реализации как на языке ассемблера, так и на языке Си, поэтому их сравнение сложно назвать корректным.

В связи с этим для демонстрации были выбраны два набора тестов, взятых из существующих Си компиляторов. Первый – набор тестов из компилятора scc [9]. Второй – тесты компилятора fcc [10]. Всего было использовано 191 синтетических тестов. Сравнение проводилось только на уровнях оптимизации O0 и O1. Для нестековой версии компилятора приводятся значения двух вариантов направления роста стеков: данные вверх и возвраты вниз, и данные вниз и возвраты вверх. Единицей измерения производительности служит количество тактов процессора, измеренное в программном эмуляторе, единицей измерения объёма – число 16-битных слов.

По результатам в среднем размер кода по сравнению с нестековой версией уменьшается на 50.8%. Заметным исключением оказывается случай return 0, но это связано с тем, что значение в нестековой версии возвращается через регистр N, а значит код можно объединить в конструкцию MOVA 0; LOAD N RET. По времени выполнение среднее улучшение составляет 35.7%. Однако в отдельных тестах наблюдалось и ухудшение до 10%. В основном это связано с дороговизной стековых операций на процессоре TF16.

7. Заключение

В данной работе были рассмотрены задачи, связанные с разработкой компилятора языка Си для стековой архитектуры TF16 на основе компилятора LLVM, а также особенности реализации генерации кода для стековой архитектуры в рамках компиляторной инфраструктуры, предназначенной прежде всего для RISC-подобных архитектур. Было проведено сравнение эффективности кода, сгенерированного без учёта особенностей стековой архитектуры, с кодом, сгенерированным с учётом особенностей стековой архитектуры . Результаты сравнения показывают, что при учёте особенностей стековой архитектуры разработанная компиляторная инфраструктура генерирует более оптимальный код как по размеру, так и по времени выполнения.

В процессе работы над компилятором был разработан платформо-независимый алгоритм планирования команд с учётом особенностей стековых архитектур для компилятора LLVM, позволяющий реализовать поддержку в этом компиляторе других стековых архитектур.

Список литературы / References

[1] Koopman P. Stack computers: the new wave. Halsted Press, 1989. 231 p.
[2] Каршенбойм И. Стековые процессоры, или новое — это хорошо забытое новое. Компоненты и технологии, no. 2, 2004 г., стр. 130-134 / Karshenboim J. Stack processors, or new is the well-forgotten new. Components and technologies, no. 2, 2004, pp. 139-134 (In Russian).
[3] Rather E, Colburn D, Moore C. The evolution of Forth. ACM SIGPLAN Notices, volю 28, то. 3, 1993, pp. 177-199.
[4] LLVM Compiler Infrastructure. Available at: https://llvm.org, accessed 10.09.2021.

[5] Koopman P. A preliminary exploration of optimized stack code generation. Journal of Forth Application and Research, vol. 6, no. 3, 1994, pp. 241-251.
[6] GCC, the GNU Compiler Collection. Available at https://gcc.gnu.org, accessed 10.09.2021.
[7] Bailey C. Inter-boundary scheduling of stack operands: A preliminary study. In Proc. of the EuroForth Conference, 2000, pp 3-11.
[8] Shannon M. A C Compiler for Stack Machines. MS Thesis. University of York, 2006, 116 p.
[9] SCC, Simple C Compiler. Available at https://www.simple-cc.org, accessed 10.09.2021
[10] FCC, Fedjmike's C Compiler. Available at https://github.com/Fedjmike/fcc, accessed 10.09.2021

Информация об авторах / Information about authors

Леонид Владленович СКВОРЦОВ – стажёр-исследователь отдела компиляторных технологий. Научные интересы: компиляторные технологии, оптимизации.

Leonid Vladlenovich SKVORTSOV – Researcher in Compiler Technology department. Research interests: compiler technologies, optimizations.

Роман Вячеславович БАЕВ – старший лаборант отдела компиляторных технологий. Научные интересы: компиляторные технологии, оптимизации.

Roman Vyacheslavovich BAEV – Researcher in Compiler Technology department. Research interests: compiler technologies, optimizations.

Ксения Юрьевна ДОЛГОРУКОВА – младший научный сотрудник отдела компиляторных технологий. Научные интересы: компиляторные технологии, оптимизации.

Ksenia Yurievna DOLGORUKOVA – Researcher in Compiler Technology department. Research interests: compiler technologies, optimizations.

Евгений Юрьевич ШАРЫГИН – стажёр-исследователь отдела компиляторных технологий. Научные интересы: компиляторные технологии, оптимизации.

Eugene Yurievich SHARYGIN – Researcher in Compiler Technology department. Research interests: compiler technologies, optimizations.

DOI: 10.15514/ISPRAS-2021-33(5)-9

CC BY

Динамический анализ IoT-систем на основе полносистемной эмуляции в QEMU

Р.Д. Коваленко, ORCID: 0000-0002-2225-5560 <r.kovalenko@ispras.ru>
А.Н. Макаров, ORCID: 0000-0003-2237-3396 <alex1118@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. Стремительное развитие Интернета-вещей (IoT) требует развития методов и средств анализа подобных устройств. Существенная часть таких устройств работает под управлением операционных систем (ОС) семейства Linux. Непосредственное применение существующих средств для анализа программного обеспечения (ПО) данного класса устройств не всегда возможно. В процессе исследования встраиваемых ОС Linux был создан инструмент ELF (Embedded Linux Fuzz), который и представлен в данной работе. В статье рассматривается анализ систем, построенных исключительно на базе ядер Linux. Среда ELF предназначена для динамического анализа устройств на основе полносистемной эмуляции в QEMU. В основу ELF были положены следующие аспекты: выполнение тестирования и анализа ПО реальных устройств в среде, максимально приближенной к их «родной» среде выполнения; интеграция с существующими средствами фаззинга; возможность проведения распределенного анализа.

Ключевые слова: фаззинг; тестирование; IoT-системы; Linux

Для цитирования: Коваленко Р.Д., Макаров А.Н. Динамический анализ IoT-систем на основе полносистемной эмуляции в QEMU. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 155-166. DOI: 10.15514/ISPRAS–2021–33(5)–9

Dynamic analysis of IoT systems based on full-system emulation in QEMU

R.D. Kovalenko, ORCID: 0000-0002-2225-5560 <r.kovalenko@ispras.ru>
A.N. Makarov, ORCID: 0000-0003-2237-3396 <alex1118@ispras.ru>

Ivannikov Institute for System Programming of the RAS,
25, Alexander Solzhenitsyn st., Moscow,109004, Russia

Abstract. The sweeping evolution of the Internet of Things (IoT) requires the development of methods and tools for analyzing such devices. A significant part of similar devices run under operating systems (OS) of the Linux family. Direct application of existing tools for analyzing software (SW) of this class of devices is not always possible. In the process of researching embedded Linux OS, the ELF (embedded linux fuzz) tool was created, which is presented in this work. The article deals with the analysis of systems built exclusively on the basis of Linux kernels. ELF environment is designed for dynamic analysis of devices based on full-system emulation in QEMU. ELF was based on the following aspects: performing software testing and analysis of real devices in an environment as close as possible to their «native» execution environment; integration with existing fuzzing tools; the ability to conduct distributed analysis.

Keywords: fuzzing; testing; IoT-systems; Linux

For citation: Kovalenko R.D., Makarov A.N. Dynamic analysis of IoT systems based on full-system emulation in QEMU. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 155-166 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–9

1. Введение

В процессе проведения динамического анализа ПО IoT-устройств обычно возникает ряд проблем, которые прежде всего связаны с особенностями реализации и функционирования таких устройств. Во-первых, аппаратура и архитектура Интернета-вещей разнообразны (ARM, MIPS, x86 и т.д.) и зависят от реализации конкретного IoT-устройства, из-за чего анализ ПО данного класса может быть существенно затруднен. Во-вторых, для прикладного ПО IoT-систем характерно отсутствие исходных кодов, что само по себе не является уникальной особенностью IoT-систем, однако создает некоторые дополнительные трудности при выполнении их анализа, в том числе динамического. Анализ бинарного кода внутри десктопных систем, как правило, не вызывает никаких затруднений на самом раннем его этапе (дизассемблирование, запуск и т.д.), поскольку объект анализа полностью доступен. В случае же IoT-систем, процесс анализ обычно начинается с получения доступа к бинарным кодам объектов анализа. В данной статье мы не будем останавливаться подробно на этом шаге, поскольку распаковка образов IoT-устройств является отдельной темой для исследования. Отметим только тот факт, что в простом случае можно воспользоваться средством binwalk [1] или аналогичными, а также обратиться на сайт производителя, который может предоставлять обновления для своих IoT-устройств, а в некоторых случаях даже исходные коды для отдельных частей их систем.

Запуск бинарного кода IoT-устройства вне предполагаемой производителем среды выполнения может быть либо вовсе невозможен, либо затруднен, в силу чего необходимо будет провести ряд действий по настройке среды выполнения и самого ПО. Это, в свою очередь, может привести к тому, что поведение объекта анализа будет существенно отличаться от предполагаемого в оригинальном окружении. Такое положение вещей существенно ограничивает применение динамических методов анализа, в том числе фаззинга. Использование эмулятора QEMU позволяет только частично решить указанную проблему.

Часто для организации динамического анализа бинарного кода необходимо внесение дополнительных инструментов в среду выполнения (отладчики, профилировщики и т.д.). Добавление подобного инструментального кода внутрь среды выполнения IoT-устройства не всегда возможно, а в ряде случаев сопряжено со множеством трудностей. Во-первых, объём памяти IoT-устройства может быть сильно ограничен. Во-вторых, производитель IoT-устройства может использовать файловые системы (ФС), в которых на операции записи в ФС накладываются дополнительные ограничения. В-третьих, для того, чтобы пересобрать образ IoT-устройства, может потребоваться дополнительный анализ реализации загрузчиков, кода проверки целостности и т.п [2]. При этом даже в тех случаях, когда удалось поместить, например, отладчик, внутрь IoT-устройства, его использование может быть затруднено по самым различным причинам (например, ограниченность функциональности – в ядре недоступен системный вызов prtace).

Одним из эффективных подходов для организации проведения динамического анализа бинарного кода (и, в частности, фаззинга) является применение средств инструментального анализа бинарного кода (Pin Tool [3], DynamoRio [4]). Их использование внутри IoT-устройства не всегда представляется возможным, поскольку используемое окружение внутри среды, в которой функционирует код IoT-устройства, не позволяет этого сделать. В частности, внутри большинства IoT-устройств работает ОС на базе старых ядер Linux (линейки вторых и третьих версий), что также накладывает дополнительные ограничения на инструменты анализа таких систем.

Основываясь на опыте анализа IoT-систем, можно выделить два основных подхода к построению программной среды для анализа кода ядра, сервисов и пользовательских утилит, функционирующих внутри IoT-устройства. Суть первого подхода (назовем его внешним) заключается в том, что конкретные сервисы и/или утилиты извлекаются из IoT-устройства и запускаются во внешнем окружении, близком к оригинальному. Далее к ним применяются различные технологии анализа (фаззинг, символьная интерпретация, и т.д.). Второй подход (внутренний), наоборот, заключается в том, что к функционирующим в оригинальной программной среде IoT-устройства сервисам и утилитам применяются в каком-либо виде технологии анализа.

Оба подхода обладают рядом достоинств и недостатков. Самым существенным недостатком внешнего подхода является тот факт, что зачастую IoT-устройство обладает своим специфическим кодом ядра (как само ядро, так и различные модули ядра), от особенностей реализации которого сильно зависит поведение сервисов. Это делает слишком трудоемким процесс переноса анализируемых сервисов в инструментальную программную среду (даже в том случае, когда удалось подобрать полностью или частично подходящее по параметрам стандартное ядро Linux). Также зачастую сервисы, функционирующие внутри IoT-устройства, сильно связаны друг с другом, в том числе общими данными, которые неочевидным образом инициализируются в ходе трудоемкого для анализа процесса загрузки устройства. Поэтому извлечь из IoT-устройства и заставить штатным образом функционировать какой-либо один интересующий сервис (особенно сетевой) не всегда представляется возможным.

Основным преимуществом внешнего подхода является тот факт, что если удалось построить инструментальную программную среду для корректного функционирования интересующих сервисов, то дальнейший процесс их анализа (в том числе фаззинга) упрощается. Появляется возможность запускать в этой же среде различные продвинутые средства анализа, осуществлять фаззинг с учетом покрытия кода, отслеживать аварийные завершения, выходы за границы буферов памяти и т.д. Однако даже в случае успешного запуска сервисов во внешней среде, существуют ограничения, обусловленные свойствами анализируемого ПО. Часто в IoT-системах используют старые версии ядер Linux и библиотек. Инструментальные средства для анализа ПО разрабатываются с учетом работы на современных версиях ОС. Поэтому в полученном внешнем окружении такие средства почти наверняка не заработают «из коробки», их необходимо будет дополнительно настраивать, возможно компилировать специально созданными или подобранными тулчейнами.

Основным преимуществом внутреннего подхода к анализу IoT-систем является сохранение штатной среды функционирования анализируемого программного кода. Это практически исключает появление ложных ошибок (например, вследствие того, что какие-либо данные неправильно инициализированы, либо отсутствуют зависимости и т.д.), в отличие от использования внешнего подхода к анализу. Однако при этом необходимо иметь возможность запуска своего кода внутри IoT-устройства, что не всегда возможно без проведения дополнительных исследований, что является недостатком данного подхода. Еще одной трудностью, с которой можно столкнуться при использовании внутреннего подхода к анализу IoT-систем, является наличие естественных ограничений на выполняемый код, что связано с особенностями реализации того или иного IoT-устройства – платформозависимость, зависимость от версий используемых библиотек, ядра и т.д. Подобные ограничения, как уже упоминалось выше, могут возникнуть и в случае применения внешнего подхода.

Исходя из перечисленных достоинств и недостатков описанных подходов к анализу IoT-устройств, становится ясно, что в большинстве случаев целесообразно применять комбинированный подход, который заключается в том, что если в IoT-устройстве есть некоторый бинарный код, который обладает небольшим количеством внешних зависимостей, и его можно оттуда относительно просто извлечь и запустить, то анализ такого

кода можно осуществлять вне IoT-устройства (например, различные библиотеки, используемые сервисами внутри устройства, несложные утилиты, и т.д.). Анализ же сложных модулей, таких как например сетевые сервисы, целесообразно осуществлять внутри IoT-устройства.

Также в качестве отдельного обширного и довольно перспективного направления стоит выделить применение технологии виртуализации к фаззингу IoT-систем.

Данная работа состоит из следующих разделов: (2) приведен обзор существующих методов и средств для анализа IoT-устройств, в том числе на основе полносистемной эмуляции; (3) приведена схема работы инструмента ELF, его основные компоненты; (4) приведены результаты измерений на синтетических тестах, а также ряд сравнений с существующими инструментами; (5) в заключение предлагаются дополнительные новшества и улучшения в работе, которые появятся в будущих версиях инструмента ELF.

2. Существующие методы и средства

Обобщим основные подходы к динамическому анализу IoT-систем, классифицируя их по методу запуска встраиваемого кода.

1) Аппаратная отладка с использованием интерфейса JTAG или подобных средств [5]. Данный метод может применяться для анализа только части кода IoT-устройства, но трудно реализуем для анализа кода пользовательского режима. Метод аппаратной отладки зависит от IoT-устройства. Он, как правило, выполняется вручную, его трудно автоматизировать и масштабировать.

2) Технология эмуляции выполнения кода пользовательского режима (QEMU-usermode) [6] является простым и быстрым методом динамического анализа пользовательского кода, позволяющим запускать бинарный код, предназначенный для различных аппаратных архитектур (arm, mips и т.д). Суть использования данного подхода заключается в извлечении usermode-среды (бинарный код, файлы настроек и т.п.) из IoT-устройства и применения chroot, что позволяет решить проблему отсутствия файлов динамических библиотек и прочих ссылок во время выполнения. Кроме того, возможен запуск gdb для отладки кода. Однако серьезное ограничение накладывает сам механизм работы эмуляции пользовательского режима. Как только в коде встречается системный вызов, он переадресуется хостовой системе, а та, в свою очередь, не всегда может выполнить его корректно, поскольку для обработки системного вызова может требоваться специфичная аппаратура или проприетарные модули ядра, отсутствующие в ядре хостовой системы.

3) Полносистемная эмуляция в одной из сред виртуализации, например, в системном режиме QEMU. Если удается выполнить эмуляцию IoT-устройства подобным образом, то на основе данного подхода можно получить хорошие результаты. Например, такой инструмент, как FIRMADYNE [7] основан на полносистемной эмуляции.

Ключевое слово здесь – «если». Полносистемная эмуляция IoT-устройства не может быть выполнена для произвольного устройства, из-за их большого разнообразия. Современные среды виртуализации (vmware, qemu, virtualbox и т. д.) поддерживают лишь типовой набор виртуальных устройств для каждой архитектуры. Использование полносистемной эмуляции также не означает, что эмулируется абсолютно весь код IoT-устройства. Например, код загрузчика может быть заменен, если это не принципиально для проведения исследования.

4) Дополненная (augmented) эмуляция – это технология совмещения эмуляции пользовательского режима и полносистемной эмуляции. Основным преимуществом данного подхода является повышение скорости работы динамического анализа. Например, FIRM-AFL основана на этом методе [8]. Однако такой подход сам по себе не устраняет недостатков полносистемной эмуляции, он не может решить проблему аппаратной зависимости. Кроме того, переключение и обеспечение синхронизации между пользовательским и системным режимами приводит к дополнительным издержкам.

5) Частичная эмуляция Qemu – это технология эмуляции оборудования, информация о котором передается Qemu в виде описания на скриптах lua. LuaQEMU [9] — это среда для прототипирования целевых систем. Интерпретатор LuaJIT встроен в QEMU и предоставляет доступ к внутреннему API QEMU для эмуляции оборудования. Основным недостатком применения указанного подхода к анализу реальных IoT-систем является трудоемкость прототипирования в общем случае, даже если представлена документация на микросхему устройства.

6) Анализ, основанный как на реальных IoT-устройствах, так и на технологии эмуляции. Данный подход направлен на решение проблемы точности выполнения кода IoT-системы, которая не может быть решена с помощью эмуляции, и проблем автоматического анализа, которые не могут быть решены с помощью аппаратной отладки. Аватар [10] и Аватар 2 [11] являются примерами этого подхода, пытающиеся перенаправить операции ввода-вывода на реальные устройства, если их невозможно выполнить в среде эмуляции. Однако Аватар и Аватар 2 обладают концептуальной проблемой, заключающейся в том, что многократные переключения между средой эмуляции и реальными устройствами негативно влияют на скорость работы при полномасштабном фаззинг-тестировании.

Из общедоступных известных современных реализаций, основанных на полносистемной или дополненной эмуляции, отметим следующие: инструмент FIRMADYNE [7], FIRM-AFL [8], TriforceAFL [12], IoTFuzzer [13], luaqemu [9]. Опишем немного более детально основные особенности некоторых из этих реализаций.

Инструмент Firmadyne [7] представляет собой фреймворк для автоматического тестирования заданных IoT-устройств на наличие в них известных уязвимостей. В его составе имеется таблица производителей устройств, с соответствующими URL-адресами, по которым с использованием некоторого автоматизированного инструмента происходит получение с сайтов производителей бинарных образов устройств, запуск их под управлением QEMU в режиме специальной полносистемной эмуляции и последующая автоматическая проверка на наличие известных уязвимостей с использованием Metasploit Framework[14]. В результате тестовых запусков фиксируются как удачные попытки эксплуатации уязвимостей, так и аварийные завершения внутренних процессов. При этом в базе данных данного инструмента для каждого из хотя бы раз запущенных IoT-устройств сохраняется информация для возможности выполнения последующих тестовых запусков уже без повторного выкачивания образа. Поддерживаются архитектуры ARM и MIPS. В данной схеме интерес представляет подход FIRMADYNE к полносистемной эмуляции IoT-устройства в QEMU. Образ каждого выкачанного устройства разворачивается путем запуска эмулятора со специально подготовленным заранее ядром Linux (для всех образов используется одно и то же ядро, собранное разными кросс-компиляторами под соответствующую архитектуру) и образом корневой файловой системы. В данное ядро на уровне исходных кодов встраивается драйвер, который выставляет заглушки на узлы дерева устройств, псевдо-файловой системы /proc и т.д., тем самым заставляя IoT-устройство видеть необходимые ему для работы элементы системы. В качестве содержимого корневой файловой системы выступает извлеченная в автоматическом режиме из образа IoT-устройства файловая система, в которую добавлена поддержка дополнительной консоли для трансляции управляющих команд, а также библиотека для управления эмуляцией nvram через упомянутый выше драйвер ядра. Более подробное описание и исходный код проекта FIRMADYNE приведены в [7].

Инструмент TriforceAFL [12] сам по себе не является ни фаззером, ни фреймворком для полносистемной эмуляции IoT-устройств, а представляет из себя механизм для фаззинга ядра Linux с использованием AFL. Данный механизм реализован путем добавления в код эмулятора QEMU функционала, реализующего, во-первых, несуществующие инструкции для архитектур x86 и ARM с целью передачи через них мутированных данных от AFL внутрь гостевой системы на системные вызовы, во-вторых, fork-сервер внутри QEMU, и в-третьих, сбор карты покрытия AFL внутри QEMU. Поддерживается только версия QEMU 2.3. Для

организации фаззинга ядра Linux с использованием TriforceAFL необходимо на целевом ядре запустить корневую ФС, содержащую в качестве процесса init так называемый user-mode драйвер фаззинга. Этот драйвер содержит несуществующие инструкции процессора, выполнение которых приводит к определенной последовательности действий внутри IoT-устройства и QEMU: запуск fork-сервера; получение мутированных данных от фаззера; начало и завершение выполнения целевого кода для фаззинга. Этот код представляет собой обертку, распределяющую мутированные данные по параметрам целевого системного вызова. Примеры организации фаззинга с использованием TriforceAFL приведены в [12]. Также следует отметить существенную особенность TriforceAFL – фаззинг выполняется в RAM-памяти, на каждом тестовом запуске гостевой системе недоступен жесткий диск, что годится для фаззинга ядра, но может существенно повлиять на достоверность результатов фаззинга в случае использования указанного подхода к фаззингу user-mode кода.

Инструмент FIRM-AFL [8] предназначен непосредственно для фаззинга IoT-устройств. Новые подходы, используемые в данном инструменте, прежде всего направлены на оптимизацию по скорости процесса фаззинга. В качестве фаззера используется AFL. Полносистемная эмуляция IoT-устройств выполняется с использованием FIRMADYNE [7], за исключением того, что выбрана более новая версия ядра, на которой запускаются целевые устройства. Также, как и в случае TriforceAFL [12], используется встроенная в QEMU карта покрытия AFL, однако в отличие от него, используется не встроенный в QEMU fork-сервер, а механизм легковесных снимков (snapshots), при этом выбор момента сохранения состояния гостевой виртуальной машины, так же, как и другие вспомогательные моменты, выполняется с использованием DECAF [15] и встроенной в нее интроспекции. Одним из средств оптимизации по скорости является использование дополненной эмуляции, суть которой заключается в том, что в процессе инициализации целевого IoT-устройства выполняется полносистемная эмуляция, а после того, как гостевая система перешла в нужную точку, делается снимок ВМ, и фаззинг запускается в режиме user-mode эмуляции, поскольку выполняется тестирование прикладных программ. При этом в некотором файле хранится состояние полносистемного контекста, используемого целевым приложением, а системные вызовы транслируются в ядро хостовой системы. Такой подход, со слов авторов, позволяет значительно увеличить скорость фаззинга. В [8] указано, что данный проект поддерживает архитектуры MIPS и ARM, однако по факту анализ кода из открытого репозитория проекта показал, что в работоспособном состоянии находится только код для архитектуры MIPS, а код для ARM не тестировался. При этом анализ кода также показал, что реализация довольно сильно привязана к архитектуре, и добавление, например, поддержки архитектуры x86 представляется трудоемкой задачей.

3. Инструмент ELF

Инструмент ELF представляет собой инструментальную платформу на основе полносистемной эмуляции QEMU, обеспечивающую внесение различных инструментов анализа (в том числе фаззеров) в штатную среду функционирования программного кода IoT-устройств. Как правило, для запуска дистрибутива ОС Linux (в том числе и для IoT-систем) в среде виртуализации QEMU нужен образ ФС, ядро, образ начальной корневой ФС (initrd) и загрузчик. Для работы с ELF требуется предварительно выделить все эти компоненты из IoT-устройства. Условно такой подход можно назвать – «разделяй и властвуй», т.к. в этом случае дополнительный инструментальный код может быть добавлен в любой из указанных компонентов в зависимости от решаемой задачи.

В некоторых случаях удаётся восстановить версию и конфигурацию ядра IoT-системы. Тогда ядро может быть заново собрано из исходных кодов с добавлением различных модулей, полезных для отладки и анализа.

Если образ ФС позволяет вносить изменения, то очевидным образом, инструментальные средства добавляются на уровне ФС.

Однако в ходе проведенных исследований было установлено, что наиболее простым способом является внесение в целевую среду дополнительных инструментальных средств анализа через начальную корневую ФС. При таком способе инжектирования инструментальные средства остаются прозрачны для программного кода IoT-устройства и не влияют на его функциональность.

В состав ELF входит набор скриптов, обеспечивающих автоматическую генерацию начальной корневой ФС при каждом запуске QEMU в соответствии с запускаемым сценарием теста (выбор инструментальных средств, входные параметры и т. п.).

На рис. 1 приведена схема работы ELF и его основные компоненты.

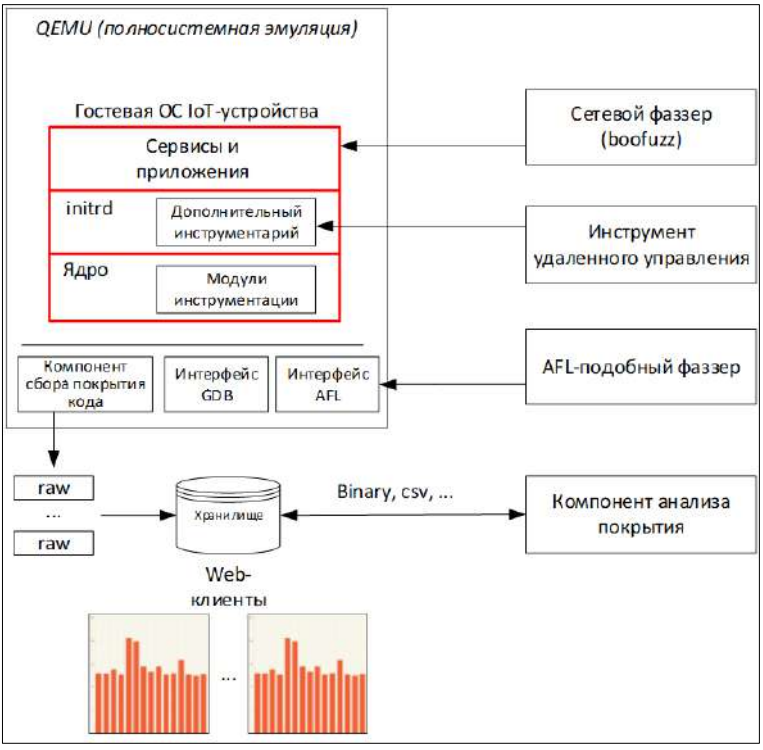


Рис. 1. Основные компоненты инструмента ELF
Fig. 1. The main components of the ELF tool

В настоящее время в ELF авторами реализована поддержка AFL-подобных [16] и сетевых фаззеров Sulley/Boofuzz [17, 18] для архитектур x86_64 и ARM. Для использования режима отладки QEMU может быть запущена со встроенным GDB-сервером. Данный режим используется для работы с такими фаззерами, как Sulley/Boofuzz.

Далее рассматриваются основные компоненты ELF.

а) *QEMU*. Основная часть ELF – это полносистемный эмулятор QEMU. В качестве базовой версии выбрана специализированная версия QEMU ИСП РАН [19]. В эту версию QEMU авторами внесены следующие дополнительные компоненты:

- интеграции с AFL фаззером (частично заимствованно из проекта TriforceAFL);
- сбора покрытия кода;
- работы со снимками.

В текущей версии ELF интегрирован только с AFL фаззером через интерфейс AFL и драйвера фаззинга, по аналогии как это сделано в TriforceAFL. Схематично взаимодействие AFL и QEMU в инструменте ELF показано на рис. 2.

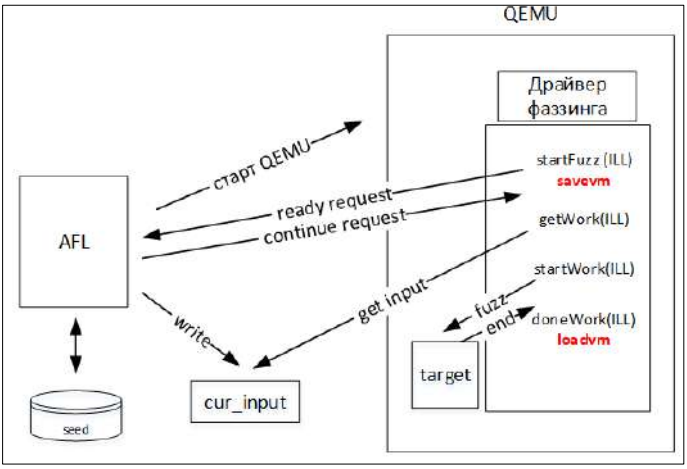


Рис. 2. Протокол взаимодействия AFL и QEMU
Fig, 2. Protocol of interaction AFL and QEMU

В драйвер фаззинга вставляется специальная несуществующая машинная инструкция, которая обрабатывается QEMU особым образом и сигнализирует в какой момент начинать фаззинг. Фаззер AFL запускается непосредственно на хосте, где и генерирует входные данные для исследуемого приложения. Эти данные транслируются в QEMU и уже в среде IoT-устройства передаются приложению. Таким образом, нет необходимости портировать фаззер для работы в среде IoT-устройства. Аналогичным способом можно адаптировать не только AFL, но и другие существующие фаззеры без необходимости выполнять их сборку кросс-компилятором под другую архитектуру.

В инструменте ELF авторами реализован собственный механизм подсчета покрытия кода, не связанный с каким-либо фаззером. Компонент сбора покрытия кода выполняет фиксацию адресов выполненных базовых блоков. Данный механизм, помимо традиционной оценки процесса фаззинга конкретным фаззером, может использоваться также для сравнения различных фаззеров, работающих на платформе ELF над одной и той же целью.

В отличие от TriforceAFL, в инструменте ELF в QEMU для фаззинга применяется не fork-сервер, а механизм снимков (vm-snaphots). Несмотря на то, что последний является более медленным, использование fork-сервера предполагает тестирование исключительно в ram-памяти после порождения дочернего процесса. Особенности реализации fork-сервера в QEMU таковы, что подобный подход влечет за собой отсутствие фиксации тех ошибок, на возникновение которых влияют данные и код считываемый с жесткого диска ВМ. Такое поведение негативно влияет и на качество проводимого фаззинга.

б) *Компонент анализа покрытия*. В ходе работы ELF на хостовой машине сохраняется информация о полученном покрытии, а именно: 1) виртуальные адреса всех блоков кода (как для ядра, так и пользовательского режима), 2) карта покрытия AFL (для совместимости с этим фаззером), 3) в том случае если возможно перекомпилировать ядро (с опцией GCOV_KERNEL), то сохраняется покрытие ядра в формате gcov.

Компонент анализа покрытия агрегирует указанные виды собираемого покрытия, обрабатывает и отображает. Для пользователя покрытие представляется в базе данных Ida Pro для определенного модуля или через Web-браузер для всей системы в целом с разбиением по коду пользовательского режима и коду ядра (рис. 3).

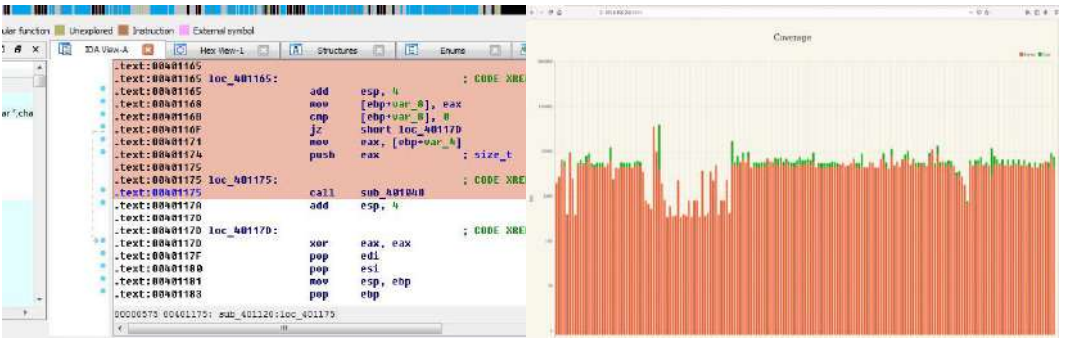


Рис. 3. Отображение покрытия кода в инструменте ELF: в базе данных Ida Pro и в Web-интерфейсе для ядра и пользовательского режима

Fig. 3. Displaying code coverage in the ELF tool: in the Ida Pro database and in the web interface for kernel and user mode

в) *Дополнительный инструментарий*. Данный компонент ELF включает в себя дополнительные usermode-модули, разработанные авторами для управления и тестирования:

- conroot – это инструмент, который обеспечивает удаленное управление посредством shell-команд, позволяя выполнять команды в ОС IoT-устройства или запускать приложения из набора дополнительного инструментария;
- драйверы для фаззинга – дополнительные usermode-библиотеки, которые определяют цель фаззинга и позволяют проводить тестирование не только отдельных утилит, но и библиотек или драйверов в ОС IoT-устройства;
- набор утилит для работы с покрытием;
- тестовые утилиты – набор приложений с внесенным ошибками для самотестирования.

г) *Отладочные инструменты*. Для анализа ОС IoT-устройства добавляются такие утилиты, как gdb, gdbserver, strace и т.п. Кроме того, если есть необходимость, то возможно включить в сборку специальный собранный busybox, расширяющий штатный функционал IoT-устройства.

д) *Интерфейсом инструмента ELF* для взаимодействия с пользователем является набор скриптов, который обеспечивают не только запуск тестов, различных фаззеров, но и выполняет при необходимости пересборку всех компонент в зависимости от сценария тестирования. Тем самым исключаются ошибки в работе ELF, связанные с рассогласованностью отдельных компонентов и модулей.

4. Синтетические тесты

В данном разделе приводятся результаты работы инструмента ELF на различных синтетических тестах. Для проверки и контроля работоспособности инструмента ELF в его состав входит три группы синтетических тестов:

- тесты на основе Juliet v1.3 [20];
- тесты от Google на основе fuzzer-test-suite [21];
- сетевой сервер vulnserver, основанный на проверочном тесте для Boofuzz [22].

Для первого и второго набора тестов созданы драйвера фаззинга. С их помощью осуществляется тестирование AFL-подобным фаззером. Третий тест может быть выполнен большинством сетевых фаззеров. В проводимом исследовании в качестве сетевого фаззера выбран Boofuzz.

Первый набор тестов из Juliet v1.3 сформирован из 105 тестов с различными ошибками переполнения буфера: CWE-121 (Stack Based Buffer Overflow), CWE-122 (Heap Based Buffer

Overflow), CWE-124 (Buffer Underwrite), CWE-369 (Divide by Zero) и CWE-476 (NULL Pointer Dereference). Данный набор тестов предназначен для проверки возможности успешного фаззинга и нахождения различного типа ошибок, регистрации аварийных завершений в среде исследуемого IoT-устройства. Для каждого теста в течение нескольких минут должно быть найдено хотя бы одно аварийное завершение. Успешное выполнение данного набора тестов демонстрирует, что стенд с исследуемым IoT-устройством корректно настроен и находится в работоспособном состоянии.

Второй набор тестов (fuzzer-test-suite) предназначен для определения скорости работы фаззеров и получаемого ими покрытия. Каждый тест представляет собой библиотеку, которая содержит известные ошибки, выявленные ранее. С помощью кросс-компилятора, библиотеки из состава fuzzer-test-suite собираются для работы в среде IoT-устройства. Данный набор тестов может также использоваться для сравнения работы различных фаззеров. В том числе, в ходе проведения исследований было выполнено сравнение двух версий инструмента ELF, реализующих разные подходы — с использованием fork-сервера и механизма снимков ВМ: 1) на основе QEMU версии 2.3 с fork-сервером (TriforceAFL) и 2) на основе QEMU версии 5.2 с механизмом снимков ВМ. Начальные входные данные (seeds) использовались одинаковые для каждой версии.

В табл. 1 представлены результаты фрагмента испытаний на 4-х библиотеках из состава fuzzer-test-suite. В столбцах, отмеченных «v1», представлен результат работы ELF с Qemu 2.3 и fork-сервером (TriforceAFL), в столбцах с отметкой «v2» — результат работы ELF с Qemu 5.2 и механизмом снимков, а в столбцах с отметкой «v2-ram» — результат работы ELF с Qemu 5.2 и механизмом снимков с использованием RAM-диска.

Табл. 1. Сравнение производительности AFL фаззеров для Qemu с Fork-сервером
Table 1. Comparison of performance of AFL fuzzers for Qemu with Fork-server

	Время работы (ч.)			Число запусков (млн.)			Средняя скорость (запусков/ с.)			Покрытие (AFL, total paths)		Ошибки	
	v1	v2	v2-ram	v1	v2	v2-ram	v1	v2	v2-ram	v1	v2	v1	v2
libjpeg	113	113	75	16.4	2.71	2.49	40.3	6.7	9.2	701	**1359**	-	-
libc-ares	113	113	75	18.0	2.89	2.62	44.2	7.1	9.7	739	**779**	-	-
liblcms	113	113	75	3.94	0.7	0.78	9.7	1.7	2.9	212	**605**	-	+
libproj4	113	113	75	15.9	2.77	2.57	39	6.8	9.5	1629	**1817**	-	-

В результате эксперимента был получен следующий результат – при б**о**льшем числе запусков (скорость работы QEMU с fork-сервером приблизительно в 6 раз выше, чем у QEMU с механизмом снимков ВМ) ELF-версия с fork-сервером (от TriforceAFL) показывает значительно худший результат по покрытию. Заметим, что в столбце «Покрытие (AFL, total paths)» приведены результаты учёта работы всего пользовательского кода в системе, включая все фоновые процессы. Исследование показало, что при порождении дочернего процесса QEMU 2.3 (TriforceAFL) с использованием вызова fork теряется связь с виртуальным жестким диском. Такое поведение существенно влияет на логику работы процессов, использующих механизмы файлового ввода/вывода. В ряде случаев это приводит к тому, что версия ELF с QEMU 2.3 (TriforceAFL) пропускает аварийные завершения, которые были зафиксированы версией ELF с QEMU 5.2 (механизм снимков), поскольку часть кода становится недостижима. Таким образом, при тестировании библиотек в ELF с QEMU 5.2 (механизм снимков) покрытие увеличилось в случае libjpeg на 94%, libc-ares – на 5%, liblcms – на 185%, libproj4 – на 12 %.

Кроме того, версия ELF с механизмом снимков ВМ была адаптирована к работе с RAM-дисками, что увеличило скорость работы в среднем в 1.5 раза (см. столбцы v2-ram в табл. 1) и позволило получить те же результаты за 75 ч.

Третий набор тестов – это сетевой сервер vulnserver, в котором реализуется HTTP-подобный протокол. В обработчиках запросов сервера намеренно заложены ошибки. Запуск теста позволяет проверить работу сетевого стека протокола IoT-устройства. Регистрация аварийных завершений обеспечивается через интерфейс gdb QEMU.

5. Заключение

Приведенная схема использования инструмента ELF позволяет упростить процесс подготовки программного кода IoT-устройства к динамическому анализу, в том числе с применением различных существующих фаззеров.

Полученные замеры скорости работы и покрытия уже в текущей версии позволяют проводить фаззинг за приемлемое время, при этом в процессе фаззинга обеспечивается стабильная работа IoT-устройства и всего комплекса в целом. Также в инструменте ELF, в отличие от TriforceAFL, решена задача синхронизации данных по жесткому диску во время фаззинга.

С целью усовершенствования инструмента ELF планируется рассмотреть возможность добавления следующего функционала:

- расширение номенклатуры поддерживаемых фаззеров;
- добавление возможности масштабирования;
- использование декомпилятора для представления покрытия в формате lcov, даже если исходные коды отсутствуют;
- инжектирование средств бинарной динамической инструментации.

В первую очередь, планируется добавить фаззер libFuzzer для тестирования кода библиотек, входящих в состав IoT-устройства, а также один из фаззеров ядра ОС Linux.

Текущая версия позволяет запускать несколько копий инструмента ELF для проведения фаззинга. Однако, настройка каждого экземпляра тестируемой ВМ, их клонирования, указание заданий для фаззинга (цели, начальные данные) — всё это выполняется вручную. Для удобства использования и быстрого масштабирования указанные задачи в следующей версии ELF будут выполняться автоматизировано.

Несмотря на то, что настоящая версия ELF поддерживает интеграцию с базой данных Ida Pro, очевидно, что для анализа более удобен формат lcov, в котором покрытие предоставляется в виде исходных кодов с указанием блоков языка C, выполненных в ходе фаззинга. В следующие версии ELF планируется добавить разметку кода на уровне языка C. Поскольку ядро ОС Linux, многие сервисы и приложения пользовательского режима в IoT-устройствах написаны на языке C, то можно предполагать, что декомпилированный код будет функционально близок к оригинальному.

Инжектирование современных средств бинарной динамической инструментации позволит расширить класс фиксируемых ошибок на IoT-устройстве. Основная сложность заключается в том, что запуск таких средств анализа в среде IoT-устройств часто затруднителен в силу ограниченности функционала среды выполнения.

Список литературы / References

[1]. Binwalk. URL: https://github.com/ReFirmLabs/binwalk, accessed 17.10.2021.
[2]. C. Simmonds. Mastering Embedded Linux Programming. Second Edition. Packt Publishing, 2017, 983 p.
[3]. Pin - A Dynamic Binary Instrumentation Tool. URL: https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-dynamic-binary-instrumentation-tool.html, accessed 17.10.2021.
[4]. DynamoRIO. https://dynamorio.org/, accessed 17.10.2021.

[5]. M. Sharma, N. Agarwal, and S.R.N. Reddy. Design and development of daughter board for USB-UART communication between Raspberry Pi and PC. In Proc. of the International Conference on Computing, Communication & Automation, 2015, pp. 944-948
[6]. F. Bellard. QEMU, a fast and portable dynamic translator. In Proc. of the USENIX Annual Technical Conference, 2005, pp. 41-46.
[7]. D. D. Chen, M. Egele et al. Towards automated dynamic analysis for Linux-based embedded firmware. In Proc. of the Network and Distributed System Security Symposium *(*NDSS*)*, 2016, pp. 1-16.
[8]. Y. Zheng, A. Davanian et al. FIRMAFL: high-throughput greybox fuzzing of iot firmware via augmented process emulation. In Proc. of the 28th USENIX Security Symposium (USENIX Sec), 2019, pp. 1099-1114.
[9]. LuaQEMU. URL: http://github.com/Comsecuris/luaqemu, accessed 17.10.2021.
[10]. J. Zaddach, L. Bruno et al. Avatar: A framework to support dynamic security analysis of embedded systems firmwares. In Proc. of the Network and Distributed System Security Symposium *(*NDSS*)*, 2014, pp. 1-16.
[11]. M.Muench, D. Nisi et al. Avatar2: A Multi-Target Orchestration Platform. In Proc. of the Workshop on Binary Analysis Research, 2018, pp. 1-11.
[12]. TriforceAFL. URL: https://github.com/nccgroup/TriforceAFL, accessed 17.10.2021.
[13]. J. Chen, W. Diao et al. IoTFuzzer: Discovering memory corruptions in IoT through app-based fuzzing. In Proc. of the Network and Distributed System Security Symposium (NDSS), 2018, pp. 1-15.
[14]. Metasploit. URL: https://www.metasploit.com/, accessed 17.10.2021.
[15]. A. Henderson, LK. Yan et al. DECAF: A Platform-Neutral Whole-System Dynamic Binary Analysis Platform. IEEE Transactions on Software Engineering, vol. 43, issue 2, 2017, pp. 164-184.
[16]. M. Zalewski. American Fuzzy Lop. URL: http://lcamtuf.coredump.cx/afl/, accessed 17.10.2021.
[17]. J. Pereyda. Boofuzz Documentation, Release 0.4.0, 2021. URL: https://buildmedia.readthedocs.org/media/pdf/boofuzz/latest/boofuzz.pdf, accessed 17.10.2021.
[18]. Sulley. URL: https://github.com/OpenRCE/sulley, accessed 17.10.2021.
[19]. P. Dovgalyuk. Deterministic Replay of System's Execution with Multi-target QEMU Simulator for Dynamic Analysis and Reverse Debugging. In Proc. of the 16th European Conference on Software Maintenance and Reengineerin, 2012, pp. 553-556.
[20]. fuzzer-test-suite. https://github.com/google/fuzzer-test-suite, accessed 17.10.2021.
[21]. NIST Test Suites. https://samate.nist.gov/SRD/testsuite.php, accessed 17.10.2021.
[22]. CTP/OSCE Prep – Boofuzzing Vulnserver for EIP Overwrite. https://h0mbre.github.io/Boofuzz_to_EIP_Overwrite/, accessed 17.10.2021.

Информация об авторах / Information about authors

Роман Дмитриевич КОВАЛЕНКО – научный сотрудник отдела компиляторных технологий. Сфера научных интересов: информационные технологии, безопасность ПО, анализ бинарного кода, системное программирование, фаззинг, исследования программ.

Roman Dmitrievich KOVALENKO – Researcher, Compiler Technologies Department. Research interests: information technology, software security, binary code analysis, system programming, fuzzing, program exploration.

Алексей Николаевич МАКАРОВ – научный сотрудник отдела компиляторных технологий. Сфера научных интересов: информационные технологии, безопасность ПО, анализ бинарного кода, системное программирование, фаззинг, исследования программ.

Alexey Nikolaevich MAKAROV – Researcher, Compiler Technologies Department. Research interests: information technology, software security, binary code analysis, system programming, fuzzing, program exploration.

DOI: 10.15514/ISPRAS-2021-33(5)-10

CC BY

Using the functional programming library for solving numerical problems on graphics accelerators with CUDA technology

[1,2]*M.M. Krasnov, ORCID: 0000-0001-7988-6323 <kmm@kiam.ru>*
[1]*O.B. Feodoritova, ORCID: 0000-0002-2792-9376 <feodor@kiam.ru>*

[1]*Keldysh Institute of Applied Mathematics of Russian Academy of Sciences,*
4, Miusskaya sq., Moscow, 125047, Russia
[2] *Moscow Institute of Physics and Technology,*
9, Insitutsky per., Dolgoprudny, 141701, Russia

Abstract. Modern graphics accelerators (GPUs) can significantly speed up the execution of numerical tasks. However, porting programs to graphics accelerators is not an easy task. Sometimes the transfer of programs to such accelerators is carried out by almost completely rewriting them (for example, when using the OpenCL technology). This raises the daunting task of maintaining two independent source codes. However, CUDA graphics accelerators, thanks to technology developed by NVIDIA, allow you to have a single source code for both conventional processors (CPUs) and CUDA. The machine code generated when compiling this single text depends on which compiler it is compiled with (the usual one, such as gcc, icc and msvc, or the compiler for CUDA, nvcc). However, in this single source code, you need to somehow tell the compiler which parts of this code to parallelize on shared memory. For the CPU, this is usually done using OpenMP and special pragmas to the compiler. For CUDA, parallelization is done in a completely different way. The use of the functional programming library developed by the authors allows you to hide the use of one or another parallelization mechanism on shared memory within the library and make the user source code completely independent of the computing device used (CPU or CUDA). This article shows how this can be done.

Keywords: C ++; functional programming library; CUDA; OpenMP; OpenCL; OpenACC

For citation: Krasnov M.M., Feodoritova O.B. Using the functional programming library for solving numerical problems on graphics accelerators with CUDA technology. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 167-180. DOI: 10.15514/ISPRAS–2021–33(5)–10

Использование библиотеки функционального программирования для решения численных задач на графических ускорителях с технологией CUDA

[1,2]*М.М. Краснов, ORCID: 0000-0001-7988-6323 <kmm@kiam.ru>*
[1]*О.Б. Феодоритова, ORCID: 0000-0002-2792-9376 <feodor@kiam.ru>*
[1]*Институт прикладной математики им. М.В. Келдыша РАН,*
125047, Москва, Миусская пл., д.4
[2] *Московский физико-технический институт,*
141701, Россия, Долгопрудный, Институтский пер., д. 9

Аннотация. Современные графические ускорители (GPU) позволяют существенно ускорить выполнение численных задач. Однако перенос программ на графические ускорители является непростой задачей. Иногда перенос программ на такие ускорители осуществляется путём практически полного их переписывания (например, при использовании технологии OpenCL). При этом возникает

непростая задача поддержки двух независимых исходных кодов. Однако, графические ускорители CUDA, благодаря разработанной компанией NVIDIA технологии, позволяют иметь единый исходный код как для обычных процессоров (CPU), так и для CUDA. Машинный код, генерируемый при компиляции этого единого текста, зависит от того, каким компилятором он компилируется (обычным, таким, как gcc, icc и msvc, или компилятором для CUDA, nvcc). Однако, в этом едином исходном коде нужно каким-то образом указать компилятору, какие части этого кода нужно распараллеливать на общей памяти. Для CPU это обычно делается с помощью OpenMP и специальных прагм компилятору. Для CUDA распараллеливание делается совершенно по-другому. Применение разработанной авторами библиотеки функционального программирования позволяет скрыть использование того или иного механизма распараллеливания на общей памяти внутри библиотеки и сделать пользовательский исходный код полностью независимым от используемого вычислительного устройства (CPU или CUDA). В настоящей статье показывается, как это можно сделать.

Ключевые слова: C++; библиотека функционального программирования; CUDA; OpenMP; OpenCL

Для цитирования: Краснов М.М., Феодоритова О.Б. Использование библиотеки функционального программирования для решения численных задач на графических ускорителях с технологией CUDA. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 167-180 (на английском языке). DOI: 10.15514/ISPRAS–2021–33(5)–10

1. Introduction

In recent years, graphics accelerators (GPUs), used as computing devices for numerical calculations, have become increasingly widespread. Such accelerators are installed on many computing clusters, in particular, in the TOP500 list of the most productive supercomputers from June 2021, in the top ten, six use graphics accelerators from NVIDIA [1]. The speed of numerical calculations on such accelerators can be many times higher than on a CPU (according to the authors' experience, the acceleration can reach 10-20 times). Therefore, the transfer of programs implementing numerical methods to graphics accelerators is an extremely urgent task.

However, porting an existing program to a GPU is not an easy task. Perhaps the ideal option is to write the program right away so that it can execute on any computer. In any case, the first question that arises is what kind of GPU technology to use? At the moment there are at least three main technologies - OpenCL (open standard for heterogeneous systems) [2], OpenACC [3] and CUDA - developed by NVIDIA for its graphics accelerators [4]. Each of these technologies has advantages and disadvantages. The main advantage of OpenCL is its open standard. A program using OpenCL will run on any computing device that supports this standard, including NVIDIA and AMD GPUs, Intel Xeon Phi processors with Intel MIC technology, and even conventional CPUs. The main disadvantage of this technology is that the source code of the program appears in two copies: for the CPU, which is compiled by a regular compiler and is part of the main program, and the text for OpenCL is in separate files, and when changing algorithms, you must make changes in both places. The advantages and disadvantages of CUDA technology are a mirror image of the disadvantages and advantages of OpenCL. CUDA only runs on NVIDIA GPUs. On the other hand, in CUDA we have a single source code that is precompiled as a part of the main program (including the code that will be executed on the GPU). The main disadvantage of the OpenACC technology is that it is not yet widespread enough. Compilers that supports this technology are not installed on all clusters with graphics accelerators.

We choose CUDA technology. Our main argument is that in (our) real life we are faced exclusively with devices from NVIDIA. GPUs from AMD and Intel Xeon Phi processors are quite exotic, and although we have met them, they are really irrelevant. Therefore, the disadvantages of CUDA is not a disadvantage for us, but its advantages remains.

The next problem is that parallelization on shared memory on the CPU and on CUDA is done in completely different ways. If we want to get a single text that should be compiled for both CPU and CUDA, then in those places where parallelization should be, we will have to write different code (for example, using the #ifdef construction), which is inconvenient. Then the idea arose to use a

library of functional programming for the C++ language previously written by one of the authors of the article [5]. When using this library, all the specifics of the computing device (CPU or CUDA) can be placed inside the library, and the user source code will remain platform independent.

This article consists of three main parts: a short introduction to functional programming (as long as needed to understand the rest of the text), a short description of the funcprog functional programming library, and a description of the use of this library for solving numerical problems. A short working example is also given.

2. A brief introduction to functional programming

In functional programming, the central object is (as the name suggests) a function. Functions are full participants in the computational process, the same as numbers are in ordinary calculations. This means that a function can be passed as a parameter to another function and can be returned as a result of a function. A function can be calculated, just as a number can be calculated in normal calculations. A simple example is a composition of two one-place functions, which returns a new one-place function that calls both functions in sequence. In specialized functional programming languages (such as Haskell), such capabilities are built into the language, while the implementation of function composition in C++ is a non-trivial task that requires special tweaks. Examples will be given in the Haskell language, since this language allows you to write many things as concisely and at the same time clearly.

Functional programming has a number of features compared to imperative programming that can be summarized in several principles. The first and main mandatory principle, already mentioned above, is that a function is a full-fledged participant in the computational process and can be either passed as a parameter or returned as a result of a function. Among others we can mention function purity, immutable variables and lazy calculations. Let us shortly describe some other principles (important for our discussion).

Currying. Named after the American mathematician and logician Haskell Curry (the programming language Haskell is also named after him). The principle of currying is that if the parameters of a function are not fully specified, an error does not occur, but instead a function is generated with a smaller number of parameters (equal to the number of missing parameters). It is implemented in many modern functional programming languages (in particular, in the Haskell language). Currying allows a function with multiple arguments to be treated as a collection of functions with one argument.

η-reduction (eta reduction, or η-transformation). Suppose we want to write a function with one parameter, a list of numbers, that returns a list of the sines of these numbers. The text of this function in Haskell is obvious:

```
mapsin lst = map sin lst
```

It follows from the currying principle that if we omit the second parameter when calling the map `sin lst` function, that is, we write just map sin, then we will get a function with one parameter that takes a list of numbers as this parameter and returns a list of sines of these numbers, that is, in fact `mapsin` function. That is, `mapsin` is equivalent to `map sin`. The principle of η-reduction says that in such cases, the last parameter (one or more) in the function definition can be omitted. The definition of the `mapsin` function can be written shorter: `mapsin = map sin`.

Functions composition. Function composition is so important in functional programming that Haskell makes it as simple as possible. Usually, one considers the composition of one-place functions (let's call them `f` and `g`), in Haskell it is written like this:

```
(f . g) x = f (g x)
```

3. Functors, Applicatives, and Monads

Functors. Suppose we have some container that stores a number of values, for example, a list or an object or the Maybe class. Now let us set the task: apply a regular one-place function (for example, sin) to the values in the container. You know how to do this with lists – use the map function. However, how do you do this with the Maybe type, and in general, how do you do it with data in an arbitrary container? The universal approach is to entrust this responsible business to the container itself. For this, Haskell defines a special `Functor` class, in which the `fmap` function is declared:

```
class Functor f where
  fmap :: (a -> b) -> f a -> f b
  (<$>) = fmap
```

The operator `<$>` is an infix synonym of the fmap function. This is a statement for applying a function to a functor. It is similar to the operator for applying a function to an ordinary value (`$`). The fmap function prototype can be written in another equivalent form (this follows from the right-associativity of the right arrow):

```
fmap :: (a -> b) -> (f a -> f b)
```

Thus, `fmap` can be thought of as a one-parameter function that takes a function that takes and returns normal values, and converts it into a function that takes and returns functors. Any data type can be declared a functor by implementing an instance of the `Functor` class and the `fmap` function for it. Any functor implementation must satisfy two functor laws:

```
1. fmap id = id                    -- 1st law
2. fmap (g . f) = fmap g . fmap f -- 2nd law
```

Here `id` is a function that returns its argument: `id x = x`. The first law says that applying the id function to a functor should not change the functor, just as applying this function to an ordinary value does not change it. The second law is the distribution law of the functor operation with respect to the composition of functions.

Applicatives. If the task is to apply a function with two arguments to two containers (for example, to sum two lists), then the functionality of the `Functor` class will not be enough. To solve this problem, another class is intended – an applicative functor (applicative). Here is the definition of the `Applicative` class:

```
class Functor f => Applicative f where
  pure  :: a->f a
  (<*>) :: f (a -> b) -> f a -> f b
  liftA2 :: (a->b->c)->f a->f b->f c
  liftA2 f x y = f <$> x <*> y
```

Thus, in each applicative, two main operations must be implemented: the pure function, which puts an ordinary value in the «pure» applicative, and the operator (`<*>`), which takes the function placed in the applicative as the first parameter, and the value, placed in the same applicative, as the second parameter and returning the result in the same applicative.

If we look at the prototype and implementation of the `liftA2` function, we can see that it passes a function with two parameters to the operator (`<$>`), which takes a function with one parameter. However, there is no contradiction here, since we can write a function with the prototype `a->b->c` as follows: `a->(b->c)`, that is, as a function with one parameter that returns a function. Then the operator (`<$>`) will just return to us the function `b->c`, placed in the functor, which is then passed to the operator (`<*>`). By analogy with the `fmap` function, we can write the prototype of the `liftA2` function like this:

```
liftA2 :: (a->b->c)->(f a->f b->f c)
```

that is, think of it as a function with one argument, taking a function that works with ordinary values and returns a function that works with functors, «lifting» a function with two arguments (hence its name) into the applicative. By analogy with the `liftA2` function, you can write a function that «lifts» a function with three (and any other number) arguments into the applicative:

```
liftA3 :: Applicative f =>
  (a->b->c->d)->f a->f b->f c->f d
liftA3 f x y z = f <$> x <*> y <*> z
```

Any implementation of the applicative must satisfy the applicative laws:

```
pure id <*> v = v                    -- Identity
pure f <*> pure x = pure (f x)  -- Homomorphism
u <*> pure y = pure ($ y) <*> u -- Interchange
-- Composition:
pure (.) <*> u <*> v <*> x = u <*> (v <*> x)
```

Monads. Let's write the «safe» functions `safe_sqrt` and `safe_log`:

```
safe_sqrt :: (Ord a, Floating a) => a -> Maybe a
safe_sqrt x=if x<0 then Nothing else Just(sqrt x)
safe_log :: (Ord a, Floating a) => a -> Maybe a
safe_log x=if x<=0 then Nothing else Just(log x)
```

These functions return the result of `Maybe` data type, which has two constructors, `Nothing` with no arguments and `Just` with exactly one argument. Returning `Nothing` means an error condition (the function's argument is out the scope of its definition) and `Just` – normal result. Now suppose we want to calculate the square root of the logarithm of a number. For both operations, we have «safe» functions. How do we now apply the `safe_sqrt` function to the result of the `safe_log` function? The functionality of functor and applicative is not enough for this. Monads serve this purpose. Monads can be viewed as a further continuation of the applicative; they are intended for building chains of monad calculations. Each monad has two main functions: `mreturn` (`return` in Haskell) and `mbind` (operator >>= in Haskell). The `mreturn` function is similar to the pure function for applicatives (in fact, for most monads, `mreturn` is defined as pure), and the `mbind` operation has the following definition:

```
(>>=) :: (Monad m) -> m a -> (a -> m b) -> m b
```

It takes as parameters a monad and a function that takes an ordinary (non-monad) value and returns a monad value (possibly of a different type, but in the same monad). We will call such functions monadic. The `safe_log` and `safe_sqrt` functions are examples of monadic functions.

The `mreturn` and `mbind` functions must obey three monad laws. In order to formulate them, we introduce the operation of monad composition (`mcompose` or the operator >=> («fish» operator) in the Haskell language). It is defined as follows:

```
  (>=>) :: f >=> g = \x -> (f x >>= g)
```

Both operands of a monad composition and its result are monadic functions. Consequently, monadic composition can be viewed as a group operation in the space of such functions. In terms of this group operation, monad laws are formulated as follows:

```
1.  mreturn >=> f == f
2.  f >=> mreturn == f
3.  (f >=> g) >=> h == f >=> (g >=> h)
```

In other words, the `mreturn` and `mbind` functions must be defined so that, firstly, the `mreturn` function is the unit element (left and right) of the monad composition (the first two laws), and, secondly, the monad composition must be associative (the third law). Now the square root of the logarithm can be calculated like this:

```
safe_log   5  >>= safe_sqrt -- Just 1.26863624
safe_log (-5) >>= safe_sqrt -- Nothing
safe_log 0.5  >>= safe_sqrt -- Nothing
```

If an error occurs somewhere in the chain of calculations, then a quick exit from the entire chain occurs, the rest of the functions are not actually calculated. Keep in mind that the `Maybe` data type is declared as a functor, an applicative and a monad.

4. Functional programming library

4.1 General description of the library

When implementing the functional programming library for the C++ language named funcprog, the task was to write a library with which one could write in C++ in a style close to the style of the Haskell language.

The important question is what is a function in terms of this library? In the original version of the library, a function was understood as an object of the `std::function` class. This option does not suit us now, since we want the function to be executed on a graphics accelerator, and an object of the `std::function` class can be executed only on the CPU (in particular, because its implementation uses virtual functions that are not portable to the GPU). It cannot be an ordinary function either, since it (its address) cannot be passed as a parameter from the CPU to CUDA. It was decided to consider any functional object (having a functional operator ()) as a function, in particular, it can be a C++ lambda expression. However, in order for this object to be passed to CUDA, this functional operator must be marked with the __device__ keyword. The disadvantage of this approach is that the metadata of the function includes not only its prototype (parameter types and return value), but also the implementation (object class) – this is the price for the possibility of porting to CUDA. This is how the function is implemented:

```
template<typename FuncType, typename FuncImpl> struct function2;

template<typename Ret, typename... Args, typename FuncImpl>
struct function2<Ret(Args...), FuncImpl> {
  using result_type = Ret;
  function2(FuncImpl const& impl) : impl(impl){}
  result_type operator()(Args... args) const
  { return impl(args...); }
private: FuncImpl const impl;2
};
```

You can see that, unlike the `std::function` class, two template parameters are passed to the `function2` class. To simplify the work with such functions, there is a helper function _ (underscore). Here's how it is defined:

```
template<typename FuncImpl, typename Ret,
  typename... Args>
struct function2_type_ {
  using type = function2<Ret(Args...), FuncImpl>;
};

template<class F>  // Common case for functors & lambdas
```

```
struct function2_type : function2_type<decltype(&F::operator())>{};

template<class Cls,typename Ret,typename... Args>
struct function2_type<Ret(Cls::*)(Args...)> :
  function2_type_<Cls, Ret, Args...>{};

template<class Cls,typename Ret,typename... Args>
struct function2_type<Ret(Cls::*)(Args...) const>
  : function2_type_<Cls, Ret, Args...>{};

template<typename F>
using function2_type_t =
  typename function2_type<F>::type;

template<typename F>
function2_type_t<F> _(F f){ return f; }
```

Let's give a short example of working with this library:

```
double d=(_([](double x){ return x*x; }) &
_([](double x){ return x+1; }))(5); //36
```

In this example, we create two functions (using the `_` function), compose them (using the `&` operator), and call the resulting composite function with the parameter 5. As a result, we get the number 36.

The funcprog library implements function currying quite fully. To do this, the library implements an operator for applying an argument to a function using the left shift operator <<. This creates a new function with one less parameter. In particular, if the original function had a single parameter, then a function with no parameters will be created.

4.2 Implementation of functors, applicatives and monads

The implementation of functors, applicatives and monads in the funcprog library is somewhat similar to the implementation of these concepts in the Haskell language. Any class can be declared as a functor, an applicative, or a monad. For this, it is enough to implement the specialization of classes `Functor`, `Applicative` and `Monad`, respectively, for this class. No changes are required to the class itself.

Any functor or monad is a type with one parameter. In C++, parameterized types are implemented using class templates. Let us look at the definition of a functor using the Maybe class as an example. It is defined like this:

```
template<typename A> struct Maybe;
```

A class `template` is not a type and cannot be passed as a parameter to a template of another class. Therefore, another class (without a template) is defined with the name `_Maybe` (with an underscore in front). This is a real class, it can be passed as a template parameter:

```
struct _Maybe {};
```

The `Maybe` class template inherits from this class:

```
template<typename A>
struct Maybe : std::optional<A>, _Maybe, {
  ...
};
```

The specializations of the `Functor`, `Applicative` and `Monad` classes are written for this `_Maybe` class:

```
template<> struct Functor<_Maybe>{
  ...
};
```

Inside the specialization of the `Functor` class, you need to define a static function `fmap`. The specializations of the `Applicative` and `Monad` classes are defined similarly. For the applicative, the methods are called pure and apply, and for the monad their names are `mreturn` and `mbind`. When implementing static methods of these classes, one should not forget about the functor, applicative and monad laws.

Note also that the funcprog library uses the division operator as a functor operator (`fmap`), and multiplication as an applicative operator (apply).

5. Using the library for numerical methods

When solving numerical problems (for example, problems of gas dynamics or heat conduction), a certain grid (regular or, more often, irregular) is constructed in the region to be solved. On some elements of this grid (for example, nodes or cells), a so-called grid function is created, in which some physical quantities in grid elements are stored. As a rule, there are two grid functions – at the previous time step and at the current one. The main cycle of the program runs over time; at each time step, the values at the current step are calculated based on the values at the previous step. This cycle is always performed sequentially. At the end of a step, the computed new values are copied from the grid function for the current step to the grid function for the previous step (sometimes these grid functions are simply swapped). Inside the body of the main loop, there is a loop (one or more) over the grid function, in which new values of physical variables are calculated for each value of the grid function index. If the method is explicit (in which the new values of the physical variables depend only on the old ones and do not depend on new values in the neighboring grid elements), then the values in different grid elements can be calculated independently of each other, in particular, these calculations can be carried out in parallel. Thus, inner loops can (and should) be parallelized on shared memory. When calculating on a CPU, parallelization of loops is done, as a rule, using OpenMP. When calculating on CUDA, the parallelization methods are different from OpenMP. To hide the parallelization method from the applied mathematician who implements the numerical method, it is proposed to use the funcprog functional programming library as described below.

5.1 Grid Expressions and Grid Functions

Let us introduce the concept of a grid expression. This is an object defined on all grid elements, that is, for any object that is a grid expression, you can find out what its value is for any grid index. A special case of a grid expression is a grid function, which simply stores its values in memory and returns them, if necessary. For grid expressions, a template is defined for the `grid_expression` class, from which all grid expression classes must inherit (in particular, the `grid_function` class also inherits from the `grid_expression` class). Thus, the phrase «an object is a grid expression» means that the class of this object is inherited from the `grid_expression` class. This inheritance uses the Curiously Recurring Template Pattern (CRTP) [6], in which the final class is passed to the base class as a template parameter. You can read about this pattern and other metaprogramming methods in books [7,8]. You can also read about expression templates in [9].

Any grid expression can be assigned to a grid function. This assignment operator iterates over all the indices of the grid function to which the grid expression is assigned. For each index it queries the grid expression for its value and assigns this value to the grid function at the given index. This assignment operator implies that values for different indices can be computed independently of each

other, and therefore can be computed in parallel. It is in this assignment operator that the very inner loop over the elements of the grid function is executed. This assignment operator, depending on which compiler the program is compiled with, chooses the method of parallelization of this loop. If it is a compiler for CUDA (the preprocessor variable `__CUDACC__` is defined), then parallelization is carried out using CUDA, otherwise – using OpenMP. Thus, the parallelization method is hidden from the application programmer inside this assignment operator. Let us show how this assignment operator is implemented for the CPU:

```
template<class GEXP>
void operator=(grid_expression(GEXP, typename GEXP::proxy_type) const&
gexp0){
  GEXP const& gexp = gexp0();
#pragma omp parallel for
  for(size_t i = 0; i < size(); ++i)
    (*this)[i] = gexp[i];
}
```

There is one more aspect to mention when talking about grid functions. A GPU can work only with its own memory, which means that when working on a GPU, the grid function must request memory for its data in the CUDA memory. There are no problems with this either. Grid functions are designed in such a way that, when compiled on CUDA, they request memory in CUDA, otherwise, in CPU memory.

5.2 Proxy objects

The `grid_expression` class template is defined as follows:

```
1. template<class E, class _Proxy = E>
2. struct grid_expression;
```

Here `E` is the final class and `_Proxy` is the proxy class. By default (if not specified), the proxy class is the same as the final class. It is needed to create a copy of an object. The only way to transfer parameters from the main processor memory to the CUDA memory is by value transfer (that is, a copy of the parameter is made). Transfer by address and reference is not possible. Only variables of simple types (numbers and pointers) and class objects containing only simple types can be passed by value. In addition, these objects must not contain virtual methods, and the methods called on them must be accessible from CUDA. Not all objects meet all these requirements. If such an object still needs to be transferred from the CPU to CUDA, then a proxy object can be created for it that meets the listed requirements and stores all the basic data from the main object. There is another reason why it is not always possible to store references and pointers to objects, even on the CPU. The fact is that in complex expressions temporary unnamed variables may appear that do not have permanent memory, and references and pointers to which cannot be saved. Such objects must be copied. You cannot always copy objects either, since there are «large» objects (for example, data vectors) that, when copied, make a copy of this data. The general rule is as follows. If the object is «small» and does not have virtual methods, then it can be copied, and the proxy class is not needed, otherwise such a class is needed.

5.3 Grid Expressions as Functors, Applicatives, and Monads

Grid expressions can be thought of as containers (this is especially true for grid functions). In funcprog library, the containers (e.g. lists and the `Maybe` class) are functors, applicatives, and monads. This makes it possible to apply ordinary functions to the values stored in them (a property of a functor). We will also make the grid expression a functor, applicative, and monad so that functions can be applied to grid expressions as well. To understand how this can be done, consider a typical loop that calculates the new value of the grid function from the old one:

```
for(size_t i = 0; i < N; ++i)
  f_new[i] = calculate(f_old[i]);
```

Here `calculate` is a function that calculates a new value in a cell from the old one. The old value in the cell is passed to it as a parameter. In the new approach, we want to be able to write in this case:

```
f_new = _(calculate) / f_old;
```

If the calculations require several more grid functions (let's call them `f2` and `f3`), then instead of

```
for(size_t i = 0; i < N; ++i)
  f_new[i] = calculate(f_old[i], f2[i], f3[i]);
```

we can write:

```
f_new = _(calculate) / f_old * f2 * f3;
```

that is, we applied the functor property for the first grid function, and the applicative for the subsequent ones. If we want to pass to the function some additional constant value (independent of the loop index), then we could write instead of:

```
for(size_t i = 0; i < N; ++i)
  f_new[i] = calculate(f_old[i], some_value);
```

something like:

```
f_new = _(calculate) / f_old * pure(some_value);
```

Thus, the result of applying a function to a grid expression (or to several grid expressions in the case of an applicative) must also be a grid expression, that is, it can be queried for a value by index (the `[]` operator must be implemented). Grid expressions, in addition to grid functions, are also the results of applying functions to grid expressions as to functors and applicatives. In addition, the sum and difference of two grid expressions, and the product and quotient of the grid expression and numbers, are also grid expressions.

Now let `f` be the function to be applied and `gexp` the grid expression.

Functor. For a functor, we give the following definition (in pseudo-Haskell):

```
(fmap f gexp)[i] = f gexp[i]
```

Theorem 1 (on the functor). The `fmap` function defined above is functorial.

Proof. Let us rewrite the first functor law completely (without η-reduction)

```
fmap id gexp = id gexp
```

or (by the definition of the `id` function):

```
fmap id gexp = gexp
```

Further,

```
(fmap id gexp)[i] = -- defenition of fmap
  id gexp[i] =      -- definition of id
  gexp[i]
```

that is, indeed, `fmap id gexp = gexp`. The first law is proven. Let us rewrite the second functor law without η-reduction:

```
fmap (g . f) gexp = (fmap g . fmap f) gexp
```

Then

```
(fmap (g . f) gexp)[i] = -- definition of fmap
  (g . f) gexp[i] = -- definition of function composition
  g (f gexp[i])
```

On the other side:

```
((fmap g . fmap f) gexp)[i] = -- definition of function composition
  (fmap g (fmap f gexp))[i] = -- def. of fmap
  g (fmap f gexp)[i] =        -- def. of fmap
  g (f gexp[i])
```

That is, indeed,

```
fmap (g . f) gexp = (fmap g . fmap f) gexp
```

The theorem is proved. The definition of the Functor is correct. □

Applicative. The `pure` function takes on some value and «brings» it into the applicative. In our case, makes a grid expression out of it. Let us define its operator `[]` so that it returns the same value for any index:

```
(pure val)[i] = val
```

The `apply` function (an analogue of the operator <*> in Haskell and the operator * in the funcprog library) in our case accepts two grid expressions: the first (let us call it `gexp_f`) returns functions, and the second (let's call it `gexp`) – some values (parameters of these functions). We define the grid expression of the apply function as follows:

```
(apply gexp_f gexp)[i] = gexp_f[i] gexp[i]
```

Theorem 2 (on the applicative). The `pure` and `apply` functions defined above satisfy applicative laws.

Proof. Let us rewrite the applicative laws using the apply function:

```
apply (pure id) eobj = eobj           -- Identity
apply (pure f) (pure x) = pure (f x) -- Homomorphism
apply u (pure y) = apply (pure ($ y)) u -- Interchange
apply (apply (apply (pure (.)) u) v) x = apply u (apply v x) --
Composition
```

The first law (Identity):

```
(apply (pure id) eobj)[i] = -- def of apply
  (pure id)[i] eobj[i] =    -- def of pure
  id eobj[i] =              -- def of id
  eobj[i]
```

i.e. `apply (pure id) eobj = eobj`. The first law is proved. The second (Homomorphism):

```
(apply (pure f) (pure x))[i] = -- def of apply
  (pure f)[i] (pure x)[i] =    -- def of pure (2 times)
  f x
```

On the other side:

```
(pure (f x))[i] = -- definition of pure
  f x
```

The second law is proved. The third law (Interchange):

```
(apply u (pure y))[i] = -- definition of apply
  u[i] (pure y)[i] =    -- definition of pure
  u[i] y
```

On the other side:

```
(apply (pure ($ y)) u)[i] = -- def of apply
  (pure ($ y))[i] u[i] =    -- def of pure
  ($ y) u[i] =              -- def of function application
  u[i] y
```

The third law is proved. The forth law (Composition):

```
(apply (apply (apply (pure (.)) u) v) x)[i] = -- definition of apply
  (apply (apply (pure (.)) u) v)[i] x[i] = -- definitiono apply
  (apply (pure (.)) u)[i] v[i] x[i] = -- definition of apply
  (pure (.))[i] u[i] v[i] x[i] = -- def of pure
  (.) u[i] v[i] x[i] = -- rewrite function composition in infix form
  (u[i] . v[i]) x[i] = -- definition of function composition
  u[i] (v[i] x[i])
```

On the other side:

```
(apply u (apply v x))[i] = -- def of apply
  u[i] (apply v x)[i] =    -- def of apply
  u[i] (v[i] x[i])
```

The forth law is proved. The theorem is proved. The definition of the Applicative is correct. □

Monad. The monad function `mreturn` is defined in the same way as the `pure` function:

```
(mreturn val)[i] = val
```

The monad function `mbind` takes a monad and a function that takes an ordinary (non-monad) value and returns a monad. Let us define the `mbind` function for grid expressions as follows:

```
(mbind gexp f)[i] = (f gexp[i])[i]
```

Theorem 3 (on the monad). The `mreturn` and `mbind` functions defined above obey monadic laws.

Proof. Let us rewrite the monad laws in terms of the `mreturn` and `mbind` functions:

```
1. mbind (mreturn x) f = f x
2. mbind eobj mreturn = eobj
3. mbind (mbind eobj f) g =
  mbind eobj (\x -> mbind (f x) g)
```

The first law:

```
(mbind (mreturn x) f)[i] = -- def of mbind
  (f (mreturn x)[i])[i] =  -- def of mreturn
  (f x)[i]
```

The first law is proved. The second law:

```
(mbind eobj mreturn)[i] = -- def of mbind
  (mreturn eobj[i])[i] =  -- def of mreturn
  eobj[i]
```

The second law is proved. The third law:

```
(mbind (mbind eobj f) g)[i] = -- def of mbind
  (g (mbind eobj f)[i])[i] =  -- def of mbind
  (g (f eobj[i])[i])[i]
```

On the other side:

```
(mbind eobj (\x -> mbind (f x) g))[i] = -- definition of mbind
  ((\x -> mbind (f x) g) eobj[i])[i] = -- beta-reduction, substitute
eobj[i] instead of x
  (mbind (f eobj[i]) g)[i] = -- def of mbind
  (g (f eobj[i])[i])[i]
```

The result is the same. The third law is proved. The theorem is proved. The definition of the Monad is correct. □

5.4 Grid Expressions as Functors, Applicatives, and Monads

Here is an example program that calculates the `axpy` function from the BLAS library:

```
#include <iostream>
#include <funcprog_data.hpp>

template<typename T>
void axpy(T a, math_vector<T> const& x, math_vector<T> &y){
  mv(y) = _([] __DEVICE __HOST
    (T a, T xi, T &yi, size_t /*i*/){
    yi += a * xi;
  }) / p(a) * mv(x);
}
int main(){
  size_t const N = 10;
  math_vector<double> x(N, 2), y(N, 3);
  axpy(5., x, y);
  std::cout << y[0] << std::endl; // 13
  return 0;
}
```

This program compiles without any modifications for CPU and for CUDA. First, two data vectors of length 10 (`x` and `y`) are created and initialized with initial values. On the CPU, the `math_vector` class is equivalent to the `std::vector` class, and for CUDA it is equivalent to the `thrust::device_vector` class. The `mv` function turns the `math_vector` into a grid function, and the `p` function calls the `pure` function from the grid expression applicative. The assignment operator in the `axpy` function starts a loop over the grid function `y`, assigning to each of its elements the corresponding element of the grid expression on the right side of the assignment operator. This loop for the CPU is parallelized using OpenMP, and for CUDA – using CUDA. All the specifics of this parallelization are hidden from the application programmer in this assignment operator inside the library.

7. Conclusion

Declarative programming languages, which include functional languages, allow, in contrast to imperative languages (such as C++), to concisely and at the same time clearly enough to write down the desired result without going into implementation details. The specific implementation can be hidden in the language and depend on the current software and hardware environment. The C++ language turned out to be powerful enough to allow the implementation of a functional programming library in it, allowing you to write programs in a style close to the style of purely functional languages such as Haskell. Such concepts from the world of functional programming as functors

and monads, implemented in the functional programming library, turned out to be a very convenient tool for transferring numerical problems to CUDA graphics accelerators. Grid expressions were defined as functors, applicatives, and monads, allowing functions to be applied to the values stored in them. More information about the C++ language can be found in the sources [10-14].

We are currently using the proposed approach for creating generic code that simulates compressible multicomponent viscous heat-conducting medium. The program is under testing.

Список литературы / References

[1]. TOP500. URL: https://www.top500.org/

[2]. OpenCL. URL: https://www.khronos.org/opencl/

[3]. OpenACC. URL: https://www.openacc.org/

[4]. NVIDIA CUDA. URL: https://developer.nvidia.com/language-solutions

[5]. М.М. Краснов. Библиотека функционального программирования для языка C++. Программирование, том 46, no. 5, 2020 г., стр. 47-59 / M.M. Krasnov. Functional Programming Library for C++. Programming and Computer Software, vol. 46, no. 5, 2020, pp. 330-340.

[6]. J.O. Coplien. Curiously recurring template patterns. C++ Report, vol. 7, issue 2, 1995, pp. 24–27.

[7]. D. Abrahams, A. Gurtovoy. C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond. Addison-Wesley, 2004, 408 p.

[8]. M.M. Krasnov. C++ templates metaprogramming in problems of mathematical physics. KIAM RAS, 2017, 84 p. DOI: 10.20948/mono-2017-krasnov.

[9]. T. Veldhuizen. Expression Templates. C++ Report, vol. 7, issue 5, 1995, pp. 26-31.

[10]. B. Stroustrup. The C++ Programming Language. Fourth Edition. Addison-Wesley, 2013, 1376 p.

[11]. B. Stroustrup. Programming: Principles and Practice Using C++. Second Edition. Addison-Wesley, 2014, 1312 p.

[12]. B. Stroustrup. A Tour of C++. Addison-Wesley, 2014, 192 p.

[13]. B. Stroustrup. The Design and Evolution of C++. Addison-Wesley, 1994, 480 p.

[14]. The C++ Resources Network. URL: http://www.cplusplus.com/.

Информация об авторах / Information about authors

Михаил Михайлович КРАСНОВ – кандидат физико-математических наук, старший научный сотрудник ИПМ им. М.В. Келдыша РАН, доцент кафедры информатики МФТИ. Сфера научных интересов: языки программирования, метапрограммирование, функциональное программирование, применение технологий программирования к численным методам.

Mikhail Mikhailovich KRASNOV – PhD in Physics and Mathematics, Senior Researcher at KIAM RAS, Associate Professor of the Department of Informatics of MIPT. Research interests: programming languages, metaprogramming, functional programming and application of programming technologies to numerical methods.

Ольга Борисовна ФЕОДОРИТОВА – старший научный сотрудник ИПМ им. М.В. Келдыша РАН. Сфера научных интересов: разработка моделей, методов, алгоритмов и программ для численного решения задач механики сплошной среды, численные моделирование ударно-волновых процессов в неоднородных многофазных средах.

Olga Borisovna FEODORITOVA – Senior Researcher at KIAM RAS. Research interests: development of models, methods, algorithms and programs for the numerical solution of problems in continuum mechanics, numerical simulation of shock-wave processes in inhomogeneous multiphase media.

DOI: 10.15514/ISPRAS-2021-33(5)-11

Автоматизация разработки на Vulkan: предметно-ориентированный подход

[1,2] *В.А. Фролов, ORCID: 0000-0001-8829-9884 <vfrolov@graphics.cs.msu.ru>*
[1,2] *В.В. Санжаров, ORCID: 0000-0001-6455-6444 <vadim.sanzharov@graphics.cs.msu.ru>*
[1] *В.А. Галактионов, ORCID: 0000-0001-6460-7539 <vlgal@gin.keldysh.ru>*
[2] *А.С. Щербаков, ORCID: 0000-0002-5360-4479 <alex.shcherbakov@graphics.cs.msu.ru>*

[1] *Институт прикладной математики имени М.В. Келдыша РАН (ИПМ РАН), 125047, Россия, Москва, Миусская площадь 4*
[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*

Аннотация. Предлагается новый высокоуровневый подход к разработке приложений на GPU на основе Vulkan API. Цель работы – снижение трудоемкости разработки и отладки приложений, реализующих сложные алгоритмы на GPU при помощи Vulkan. Предлагаемый подход использует технологию генерации кода путем трансляции программы на языке C++ в оптимизированную реализацию на Vulkan, включающую автоматически генерацию шейдеров, привязывание ресурсов и использование механизмов синхронизации. Предлагаемое решение не является технологией программирования общего назначения, а специализируется на конкретных задачах, но обладает при этом расширяемостью, позволяющей адаптировать решение под новые задачи. Для одной входной программы на языке C++, мы можем сгенерировать несколько реализаций для разных случаев (опций транслятора) или разного аппаратного обеспечения. Например, вызов виртуальных функций может быть реализован либо через конструкцию switch в одном вычислительном ядре, либо через сортировку потоков и непрямой вызов в разных ядрах, либо через т. н. callable shaders в Vulkan. Таким образом, предлагаемая технология позволяет обеспечить кроссплатформенность решения за счёт генерации разных реализаций одного и того же алгоритма под разные GPU. В то же время за счёт этого она позволяет обеспечить доступ к специфической аппаратной функциональности, необходимой в приложениях компьютерной графики.

Ключевые слова: генерация кода; программирование GPU; Vulkan

Для цитирования: Фролов В.А., Санжаров В.В, Галактионов В.А., Щербаков А.С. Автоматизация разработки на Vulkan: предметно-ориентированный подход. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 181-204. DOI: 10.15514/ISPRAS–2021–33(5)–11

Благодарности: Работа выполнена при поддержке Российского научного фонда (РНФ) № 21-71-00037**.**

Development in Vulkan: a domain-specific approach

[1,2] *V.A. Frolov, ORCID: 0000-0001-8829-9884 <vfrolov@graphics.cs.msu.ru>*
[1,2] *V.V. Sanzharov, ORCID: 0000-0001-6455-6444 <vadim.sanzharov@graphics.cs.msu.ru>*
[1] *V.A. Galaktionov, ORCID: 0000-0001-6460-7539 <vlgal@gin.keldysh.ru>*
[2] *A.S. Shcherbakov, ORCID: 0000-0002-5360-4479 <alex.shcherbakov@graphics.cs.msu.ru>*

[1] *Keldysh Institute of Applied Mathematics Russian Academy of Science, 4, Miusskaya sq., Moscow, 125047, Russia*
[2] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia*

Abstract. In this paper we propose a high-level approach to developing GPU applications based on the Vulkan API. The purpose of the work is to reduce the complexity of developing and debugging applications that implement complex algorithms on the GPU using Vulkan. The proposed approach uses the technology of code generation by translating a C++ program into an optimized implementation in Vulkan, which includes automatic shader generation, resource binding, and the use of synchronization mechanisms (Vulkan barriers). The proposed solution is not a general-purpose programming technology, but specializes in specific tasks. At the same time, it has extensibility, which allows to adapt the solution to new problems. For single input C++ program, we can generate several implementations for different cases (via translator options) or different hardware. For example, a call to virtual functions can be implemented either through a switch construct in a kernel, or through sorting threads and an indirect dispatching via different kernels, or through the so-called callable shaders in Vulkan. Instead of creating a universal programming technology for building various software systems, we offer an extensible technology that can be customized for a specific class of applications. Unlike, for example, Halide, we do not use a domain-specific language, and the necessary knowledge is extracted from ordinary C++ code. Therefore, we do not extend with any new language constructs or directives and the input source code is assumed to be normal C++ source code (albeit with some restrictions) that can be compiled by any C++ compiler. We use pattern matching to find specific patterns (or patterns) in C++ code and convert them to GPU efficient code using Vulkan. Pattern are expressed through classes, member functions, and the relationship between them. Thus, the proposed technology makes it possible to ensure a cross-platform solution by generating different implementations of the same algorithm for different GPUs. At the same time, due to this, it allows you to provide access to specific hardware functionality required in computer graphics applications. Patterns are divided into architectural and algorithmic. The architectural pattern defines the domain and behavior of the translator as a whole (for example, image processing, ray tracing, neural networks, computational fluid dynamics and etc.). Algorithmic pattern express knowledge of data flow and control and define a narrower class of algorithms that can be efficiently implemented in hardware. Algorithmic patterns can occur within architectural patterns. For example, parallel reduction, compaction (parallel append), sorting, prefix sum, histogram calculation, map-reduce, etc. The proposed generator works on the principle of code morphing. The essence of this approach is that, having a certain class in the program and transformation rules, one can automatically generate another class with the desired properties (for example, the implementation of the algorithm on the GPU). The generated class inherits from the input class and thus has access to all data and functions of the input class. Overriding virtual functions in generated class helps user to carefully connect generated code to the other Vulkan code written by hand. Shaders can be generated in two variants: OpenCL shaders for google “clspv” compiler and GLSL shaders for an arbitrary GLSL compiler. Clspv variant is better for code which intensively uses pointers and the GLSL generator is better if specific HW features are used (like hardware ray tracing acceleration). We have demonstrated our technology on several examples related to image processing and ray tracing on which we get 30-100 times acceleration over multithreaded CPU implementation.

Keywords: Code generation; GPU programming; Vulkan

For citation: Frolov V.A., Sanzharov V.V., Galaktionov V.A, Scherbakov A.S. Development in Vulkan: a domain-specific approach. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 5, 2021, pp. 181-204 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–11

Acknowledgements. This work is supported by the Russian Science Foundation (RSF) under grant #21-71-00037.

1. Введение

Существующие на сегодняшний день кроссплатформенные технологии программирования GPU, такие как OpenCL и OpenMP, не предоставляют доступ к новейшим возможностям современных GPU, таким как непрямой вызов вычислительных ядер, буферы команд, управляемая синхронизация, аппаратная трассировка лучей, вызов виртуальных функций и др. Технологии, предоставляющие такую возможность, являются либо закрытыми и ориентированными на конкретную платформу (CUDA, OptiX, Metal), либо трудоемкими в использовании (Vulkan, DirectX12).

Фундаментальной проблемой, которая является причиной высокой трудоемкости и стоимости разработки, является отсутствие алгоритмических оптимизаций в системе трансляции/компиляции в современных технологиях программирования GPU: многие эффективные алгоритмы для разных приложений реализуются на GPU схожим или одинаковым образом, но их реализация может зависеть от конкретного GPU. На практике разработчикам приходится поддерживать несколько вариантов одного и того же алгоритма для разных GPU для достижения высокого уровня производительности и совместимости. При этом различия в исходном коде и в производительности могут быть весьма значительными. Например, реализация алгоритмов обработки изображений может быть выполнена с использованием вычислительных шейдеров или с использованием графического конвейера с аппаратным альфа-смешиванием, или с использованием возможностей мобильных GPU с тайловой архитектурой. Суть алгоритма не меняется от того, как конкретно он реализован на GPU, но разработчикам необходимо реализовывать разные версии и работать с низкоуровневыми деталями, которые отличаются для разных графических процессоров. Таким образом, кроссплатформенная разработка ПО с доступом к аппаратному ускорению является фундаментальной проблемой на сегодняшний день.

Основная цель нашей технологии – обеспечение доступа к любым текущим и перспективным аппаратным возможностям современных GPU при сохранении кроссплатформенности и, таким образом, обеспечение технологической независимости разработчика программного обеспечения от производителя оборудования.

2. Существующие решения

Количество существующих технологий программирования GPU и другого специализированного аппаратного обеспечения (например, FPGA) в настоящее время огромное [1]. Это связано с высокой актуальностью задачи и существующими проблемами в области. Рассмотрим эти решения, группируя их по способу работы и их назначению.

2.1 Общие технологии

К общим технологиям отнесем решения, не ориентированные на задачи в конкретной области, а позиционируемые как универсальные средства разработки на GPU.

2.1.1 Технологии на основе директив компилятора (прагм)

Данная группа решений ориентирована на максимальное упрощение разработки и ускорение произвольного кода на GPU настолько, насколько это возможно автоматически. Потенциальными пользователями таких технологий являются учёные и разработчики, не интересующиеся деталями реализации алгоритма на GPU. Сюда можно отнести OpenMP, OpenACC [2], C++ AMP (Miscosoft), Spearmint [3], OP2[4], [5], [6], частично DVM/DVMH [7-9] (система DVMH представляет собой исключение и будет подробнее рассмотрена позже). Основной подход, который обычно используют подобные технологии, состоит в переносе реализации отдельных параллельных участков (обычно циклов) кода на GPU и решение

задачи эффективного копирования данных между CPU и GPU при помощи конвейерной обработки.

У этого подхода есть два ключевых недостатка. Первый – слабый контроль за операциями копирования и распределения данных между CPU и GPU, которые приходится делать автоматически, что становится узким местом и приводит к низкой производительности [10]. В решении этой проблемы интересные результаты достигнуты в работе [6], где была использован программный конвейер, сочетающий в себе стадии выполнения на CPU и GPU. Однако конвейер не всегда может помочь, поскольку низкая производительность может быть вызвана неделимостью некоторых этапов вычислений либо необходимостью передавать большой объём данных. Что более важно, гибридное выполнение приложений (одновременно на CPU и GPU) не всегда нужно. Кроме того, в определённых ситуациях, управление памятью и копирование должно строго контролироваться (например, в приложениях реального времени, для чего существует OpenGL SC и Vulkan SC).

Второй ключевой недостаток данной группы технологий заключается в исключительно слабой поддержке аппаратных возможностей GPU. Например, в OpenACC невозможно напрямую использовать разделяемую память, функции голосования (warp voting functions), отсутствует поддержка текстур (что неприемлемо для высокопроизводительных и энергоэффективных приложений компьютерного зрения и компьютерной графики). Аналогичные проблемы существуют и у других технологий данной группы.

2.1.2 Скелетное программирование

Технологии данной группы нацелены на то, чтобы иметь одну реализацию алгоритма на C++, которая может работать эффективно как на CPU, так и на GPU, либо выполняться гибридно (одновременно на CPU и GPU). Сюда можно отнести такие работы как SkePU[11, 12], SkelCL[13], SYCL [14] и его производные (Intel oneAPI).

Основная идея скелетного программирования состоит в том, чтобы конструировать код программы в виде комбинации определенных базовых примитивов и операций - «скелетов», которые могут быть реализованы эффективно на GPU и, кроме того, могут быть конвейеризованы. Например, такие комбинации, как map-filter и map-reduce.

Основным преимуществом, по сравнению с предыдущей группой решений, является более высокая эффективность в отдельных алгоритмах, которые активно используют базовые скелеты или их комбинации. Недостатком является существенное ухудшение читаемости, сопровождаемости кода и его гибкости, поскольку реализацию алгоритмов приходится подстраивать под использование базовых примитивов и операций [15]. Кроме того, скелетное программирование наследует недостатки предыдущей группы подходов, в частности, в плане поддержки специализированных аппаратных возможностей GPU. Однако, следует отметить, что в скелетном программировании применяются алгоритмические оптимизации на GPU: транслятор обладает знаниями об алгоритме, благодаря чему может сгенерировать эффективную реализацию, используя низкоуровневую технологию программирования как слой нижнего уровня (т.н. back-end).

2.1.3 Вычисления на GPU общего назначения (general purpose computing on GPU, GPGPU)

К этой группе относятся CUDA и OpenCL.

На сегодняшний день CUDA является доминирующей технологией благодаря технологическому лидерству Nvidia. Однако, несмотря на это, даже помимо отсутствия кроссплатформенности, у неё имеется много недостатков: это весьма тяжеловесная технология с огромным количеством функциональностей и версий (11 версий на текущий момент), которые не всегда одинаково хорошо работают на видеокартах разных поколений и архитектур. Перенос программной системы с одной версии CUDA на другую зачастую

сопряжён со значительными затратами времени (например, из-за прекращения поддержки определенных возможностей или существенных изменениях в них). Наконец, значительный недостаток CUDA состоит в наличии глобальных состояний (текстуры, константная память), что может быть удобно для небольших проектов, но становится проблемой при отладке большого проекта.

Технология OpenCL лишена вышеперечисленных недостатков CUDA, но обладает крайне слабой поддержкой существующих аппаратных возможностей современных GPU. Например, т.н. непрямой вызов вычислительных ядер [16], который так и не появился в недавно вышедшем стандарте OpenCL 3.0. Эта функциональность является критической для сложных алгоритмов, в которых поток управления зависит от результатов вычислений на GPU [17].

Также следует отметить, что существует несколько открытых реализаций технологии CUDA через OpenCL или SIMD инструкции CPU [18-20], а также AMD HIP [21]. Эти подходы обладают таким же недостатком, который можно назвать критическим, - отсутствие поддержки аппаратных возможностей в технологии, используемой в качестве слоя нижнего уровня. Этот недостаток приводит к тому, что алгоритм либо не заработает совсем, либо будет обладать низкой производительностью из-за программной эмуляции той или иной отсутствующей функциональности. Например, в OpenCL отсутствует поддержка атомарных операций для типов с плавающей точкой. Программная эмуляция такой функциональности существует [22], но она медленная. Однако в подавляющем большинстве случаев сохранить производительность всё-таки можно, если заменить атомарные операции на параллельную редукцию. Но для этого нужно переписать алгоритм, поэтому трансформация кода должна производиться на более высоком уровне нежели делается в этих работах.

2.1.4 Графические API

Сюда можно отнести такие технологии, как OpenGL4+, DirectX10--DirectX11, DirectX12, Metal, Vulkan.

Несмотря на название, графические программно-аппаратные интерфейсы не являются узкоспециальными и почти всегда предоставляют более широкие возможности, чем API общего назначения. Вычислительные шейдеры являются частью любого из упомянутых интерфейсов и позволяют получить доступ к той же функциональности, что и, например, CUDA, и широко используются в различных приложениях.

За последние пять лет программирование графических процессоров сильно изменилось в связи с появлением новых аппаратных возможностей в таких API, как Vulkan, DirectX12 и Metal. Далее будем рассматривать только Vulkan, потому что это кроссплатформенная технология, в отличие от DirectX12 и Metal.

Наиболее существенным отличием Vulkan от предыдущих интерфейсов программирования на GPU является наличие возможностей, которых нет в других технологиях совсем либо в той же степени. К таким функциям относятся: буферы команд, настраиваемая синхронизация на основе барьеров, управление компоновкой текстур через проходы рендеринга (render passes), управление памятью графического процессора, под-проходы (subpasses) для мобильных графических процессоров, конвейер трассировки лучей (что включает в себя шейдерные таблицы, а это, по сути, есть вызов виртуальной функции) и множество функций, которые представлены, например, в OpenGL, но не включены в технологии программирования общего назначения: непрямой вызов ядер, графический конвейер, меш-шейдеры, сжатие текстур, создание работы на GPU, управляемая страничная память (разреженный буфер/текстура) и некоторые другие.

Благодаря перекладыванию ответственности на разработчика Vulkan-драйвер решает меньше задач. Поэтому разработчики программных систем, умеющие тщательно использовать возможности Vulkan API, могут получить существенные преимущества как по производительности, так и по энергоэффективности создаваемых ими приложений [23].

Кроме того, поддержка Vulkan реализована в огромном числе современных настольных и мобильных устройств [24].

Однако, из широких возможностей для разработчика следует и высокая трудоёмкость создания программ, - до 10 раз [25] больше, чем для CUDA или OpenCL, что является следствием необходимости ручного управления описанными возможностями. Это на практике может затруднять достижение цели повышения производительности, т.к. модифицировать большой объем исходного кода на Vulkan трудно, и разработчики не успевают проверить все возможные оптимизации.

Традиционный способ снижения сложности кода заключается в создании вспомогательных библиотек общего назначения или ориентированных на целевое приложение. Однако, в них пользователю все равно приходится работать с теми же сущностями Vulkan, настраивая и соединяя связи между ними. Альтернативный способ - более тяжелые варианты библиотек или движков, которые инкапсулируют сущности Vulkan и обрабатывают некоторые вещи автоматически [26-28]. Подобные решения, по сути, выполняют работу драйвера, например, OpenGL, что может приводить к неоптимальной реализации и ошибкам, поскольку разработчик в значительной степени повторяет работу производителя этого драйвера.

2.1.5 Различные технологии

В данную группу входят различные уникальные решения.

В работе [29] используются так называемые многомерные гомоморфизмы, – формальное описание задачи на высоком уровне, позволяющее выразить вычисления при помощи параллельных паттернов, которые можно по-разному реализовать на различном оборудовании. Существенное ограничение работы [29] состоит в использование OpenCL в качестве слоя нижнего уровня (OpenCL код генерируется на выходе), что не позволяет использовать многие из новых возможностей современных GPU (в том числе мобильных) в силу ограниченности OpenCL. Описания алгоритмов в [29] производится на крайне ограниченном предметно-ориентированном языке, что также является существенным недостатком. Данная технология рассматривает довольно ограниченный набор оптимизаций параметров алгоритмов, - размеры блоков, порядок циклов и использование различных типов памяти.

В TaskFlow [30] предлагается модель, позволяющая выражать вычисления в виде статических и динамических графов задач для гетерогенных вычислительных систем. Задачи общаются потоками данных между собой по рёбрам графа через очереди, используя схему производитель-потребитель (producer-consumer). В TaskFlow есть возможность построения гетерогенных графов (использующих одновременно CPU и GPU) и их дальнейшего конвейерного выполнения аналогично работе [6]. Недостаток TaskFlow в том, что алгоритм необходимо явно описывать в терминах графов задач, что не очень удобно для разработки и отладки.

PACXX [31, 32] использует идеи скелетного программирования, но более удобен для использования в существующем коде. PACXX напрямую реализует конструкции современного C++ и транслирует использование STL контейнеров в буферы на GPU. Определенный нижний уровень (бэк-энд) выбирается при помощи дополнительных вызовов объекта исполнителя (Executor) и в случае реализации на CPU эти вызовы могут игнорироваться, а в случае реализации на GPU компилятор генерирует код на CUDA или OpenCL. Однако PACXX реализует лишь базовые операции программирования и не имеет доступа ко многим аппаратным возможностям (например, к текстурам) доступным в CUDA и OpenCL.

Chapel [33] ставит целью добиться кроссплатформенности, чтобы единственное описание алгоритма могло эффективно работать как на CPU, так и на GPU. Для этого предлагается новый язык общего назначения, изначально ориентированный на параллельное

программирование. Ключевой недостаток такого подхода в том, что пользователю приходится переносить описание значительной части своего алгоритма на новый язык, что обычно встречает сопротивление в индустрии. Причины такого сопротивления в том, что при использовании совершенно нового языка кроссплатформенность обеспечивается только компилятором этого языка. Из-за чего разработчик попадает в зависимость от единственного в мире разработчика компилятора этого нового языка.

Tiramisu [34] — многогранный компилятор (то есть рассматривающий различные варианты оптимизации одного и того же алгоритма), который специализируется на высокопроизводительной генерации кода для различных платформ. Этот компилятор ориентирован на кластерные вычисления на различных платформах. Таким образом, он поддерживает выполнение одного и того же кода на многопоточных CPU, GPU производства компании Nvidia, кластерах из нескольких вычислителей и FPGA. К достоинствам этой технологии можно отнести переносимость между различным аппаратным обеспечением и высокую скорость работы относительно других схожих подходов. В то же время, данный компилятор имеет ряд недостатков. Во-первых, он поддерживает только специализированный высокоуровневый язык, что затрудняет его внедрение в приложения на других языках и увеличивает временные затраты на перенос алгоритмов, написанных на других языках программирования. Во-вторых, данный компилятор адаптирован для кластерных вычислений и имеет существенные ограничения в плане аппаратного обеспечения.

Компилятор clspv [35] позволяет транслировать ядра OpenCL в промежуточное представление кода для GPU, называемое SPIR-V (используемое в Vulkan для задания шейдерных программ) и, таким образом, может быть использован при разработке на Vulkan. Однако, он поддерживает только вычислительные шейдеры и в настоящий момент официально находится в стадии прототипа (хотя уже является относительно стабильным, т.к. разрабатывается с 2017 года). Предлагаемая нами технология использует clspv как один из возможных компиляторов нижнего уровня.

Компилятор Circle появился в 2018 году, но лишь в 2020 году стал ориентированным на программирование для GPU [36]. В настоящий момент это один их двух компиляторов C++ в мире, поддерживающий графическую функциональность современных GPU (графический конвейер, конвейер трассировки лучей, меш шейдеры). Circle - традиционный компилятор, который сам по себе имеет все недостатки традиционного подхода: если разработчик начинает использовать Circle как основной инструмент (то есть не только для шейдеров, а для всего описания алгоритма), то он становится зависимым от него и сборка на любую платформу уже невозможна без Circle. В сочетании с тем фактом, что Circle является проектом с закрытым исходным кодом, его прямое использование неприемлемо в ряде случаев, т.к. опять же разработчик начинает зависеть от единственного закрытого компилятора.

Второй компилятор шейдеров на C++ для Vulkan появился совсем недавно [37] и ориентирован на использование в составе игрового движка Unreal Engine. Как и Circle, авторы [37] используют механизм атрибутов, появившийся в стандарте C++11 в качестве стандартизированного синтаксиса для расширений языка, для обозначения частей кода, которые должны быть обработаны на GPU. Например, атрибут *[[entry]]* определяет функцию как точку входа, где вызывается GPU код, а функции, которые должны вызываться на GPU обозначаются атрибутом *[[gpu]]*. При этом GPU код инкапсулируется в классы, а механизм виртуальных функций используется для генерации различных вариантов шейдеров транслятором, входящим в решение [37]. Транслятор, реализованный на основе Clang LibTooling [38], конвертирует C++ код в C++ и HLSL код, который уже обрабатывается существующими компиляторами этих языков.

Забегая вперёд, следует сказать, что работа [37] наиболее близка к нашей работе как по сути, так и по используемым средствам обработки исходного кода. Наиболее существенное

отличие работы [37] от нашей в том, что в [37] и “хостовый” (CPU) и “девайсовый” (GPU) код пишется централизованно в одном месте, что делает эту технологию в некоторой степени похожей на CUDA; в нашей технологии, с другой стороны, *по CPU коду генерирутеся его “зеркальный клон”* на GPU, который пользователю нужно связать со своим хостовым кодом на Vulkan самостоятельно. О преимуществах и недостатках такого подхода мы будем говорить в разделе про предлагаемое решение. Ещё одно существенное отличие в том, что в качестве слоя в [37] нижнего уровня используется существующий инструментарий “Unreal Engine 4” (UE4) и его специфические механизмы, что на наш взгляд в некоторой степени ограничивает кросс-платформенность т.е. UE4 – это очень тяжеловесное решение.

2.2 Специализированные технологии

В данном подразделе рассмотрены решения, ориентированные на решение узкого круга задач в конкретной области.

2.2.1 Библиотеки, предоставляющие реализации конкретных инструментов или алгоритмов

Сюда можно отнести такие известные библиотеки как TBB, Thrust [39], CUBLAS, IPP, NPP, MAGMA, [40], [41], HPX [42] и многие другие. Некоторые из них превращаются в довольно мощные программные решения, имеющие внутри себя специфические компиляторы предметно-ориентированных языков (Domain Specific Language, DSL) для нейросетей (например, PyTorch [43], TensorFlow [44] и [45]). Основной недостаток библиотек в их ограниченных возможностях. Кроме того, многие библиотеки намеренно лишены кроссплатформенности (например, TBB, Thrust, IPP, Intel Embree), поскольку выпускаются определённым производителем аппаратного обеспечения (CPU и/или GPU), в том числе, в целях продвижения собственного оборудования и именно поэтому делаются непереносимыми.

2.2.2 Предметно-ориентированные API

В эту группу входят такие технологии, как Nvidia OptiX [46] и Microsoft DirectML [47]. OptiX — это специализированная технология для ускорения приложений трассировки лучей GPU, которая принимает на вход на C++. OptiX включает две ключевые технологии: (1) аппаратно-ускоренную трассировку лучей и (2) ускорение вызовов виртуальных функций с помощью таблиц шейдеров. Эти 2 функции также доступны в Vulkan и DirectX12. Поскольку программировать в OptiX намного проще (а это единственная технология промышленного уровня, поддерживающая C++ для трассировки лучей на GPU), почти все промышленные системы рендеринга перешли на неё: Octane, VRay, IRay, RedShift, Cycles, Thea и другие. Хотя новейшие графические процессоры AMD имеют аппаратную поддержку трассировки лучей, на самом деле для пользователей нет альтернативы Nvidia. Аналоги [48, 49] существенно отстают от решений на основе OptiX.

2.2.3 Предметно-ориентированные языки и технологии (Domain Specific Language, DSL)

Данная группа решений ориентирована на класс разработчиков в конкретной предметной области, для которых важно добиться максимального упрощения и абстрагирования от деталей реализации алгоритма на GPU при высоком уровне производительности.

Например, языки Darkroom [50] и Halide [51-53] предназначены для создания алгоритмов обработки изображений. Кроссплатформенность в Halide достигается путём реализации большого количества слоёв нижнего уровня, а высокая эффективность за счёт применения оптимизации последовательностей фильтров. Одна из ключевых оптимизаций основывается на идее переупорядочивания операций: если рассматривать применение двух фильтров

последовательно к изображению, тогда зачастую изображение можно разбить на регионы и к каждому региону сразу применить всю цепочку фильтров. За счёт этого при передаче данных от одного фильтра к другому они не выгружаются из кэша (т.к. обрабатывается небольшой регион). Это именно тот тип оптимизаций, которые используются в под-проходах (subpasses) Vulkan и Metal. Хотя в них эта оптимизация реализована в более ограниченном варианте, поскольку необходимо дополнительно решать проблему перекрывающихся регионов изображения. Например, аналогично тому, как это делает система DVMH [9] на основе теневых границ. Следующая важная оптимизация, реализованная в Halide - блочное расположение данных изображения в памяти (что в Vulkan реализуют текстуры). Это дополнительно позволяет повысить эффективность кэша.

Основная мотивация создания нового языка в том, что в традиционных языках программирования недостаточно конструкций для выражения таких вещей как параллелизм и локальность данных. Поэтому помимо упрощения разработки, Halide на самом деле преследует цель повышения контроля над механизмом исполнения алгоритма и, как следствие, производительности: пользователь имеет возможность выбирать различные варианты реализации одной и той же цепочки фильтров и измерять полученную производительность, чтобы остановиться на лучшем варианте. Наконец, существенным преимуществом Halide является возможность генерировать код на C/C++ для целевой платформы на выходе. Таким образом, пользователь не должен подключать к своему проекту компилятор Halide на целевой платформе, а может использовать сгенерированный C++ код (+ шейдеры для GPU) и любую систему сборки.

Преимущества предметно-ориентированных технологий состоит в быстрой, высокоэффективной и иногда (Halide) кроссплатформенной разработке алгоритмов определенного класса. Отчасти это происходит благодаря ограничениям, которые предметно-ориентированные языки программирования привносят в процесс разработки. Это добавляет возможности для оптимизации и генерации высокопроизводительного кода.

Ключевой недостаток предметно-ориентированных языков состоит в том, что алгоритмы и знания, написанные на них, трудно переносить на другие области и трудно состыковать с остальной частью ПО, не использующей предметно-ориентированный язык. Кроме того, чем больше областей в программном решении затрагивается, тем больше различных DSL нужно будет использовать и состыковывать, что существенно усложняет разработку. Например, в современных приложениях компьютерной графики шейдеры по сути являются предметно-ориентированным языком. В настоящее время есть несколько вариантов графического конвейера и конвейера трассировки лучей, где один алгоритм, по сути, распределяется на нескольких программ (от 2 до 5), дополняемых настройкой состояний графического конвейера или конвейера трассировки лучей в C++ коде. Это делает процесс написания программы крайне неочевидным, поскольку нет одного связанного описания алгоритма, а вместо этого имеется множество разрозненных программ и настраиваемые связи между ними в отдельных местах.

DVM — это технология программирования на основе директив, однако же использующая ряд алгоритмических оптимизаций на нижнем уровне. Например, транслятор DVM может прозрачным образом менять расположение массива в памяти “по строкам/по столбцам”, что повышает эффективность выполнения ряда алгоритмов. DVM ориентирована на кластерные вычисления и расчёты на сетках с плотными структурами данных, поэтому можно сказать, что в некотором смысле это предметно-ориентированная технология программирования, хотя и с достаточно широкими “общими” возможностями. Интересной возможностью DVM системы является динамический режим выполнения с автоматическим подбором схемы распределения данных в каждом параллельном регионе.

Если DVM работает с плотными структурами данных, то Taichi [54-56] ориентирована на приложения, работающие с разреженными структурами данных, включая физическое

моделирование, трассировку лучей и нейронные сети. Taichi позволяет пользователям писать высокоуровневый код с использованием предлагаемого языка (интерфейс встроен в C++), как если бы они имели дело с обычными плотными многомерными массивами. Затем компилятор генерирует промежуточное представление, оптимизирует его и возвращает код C++ или CUDA. Taichi также занимается управлением памятью и использует функцию унифицированного доступа к памяти, доступную в CUDA. Возможность работать как с центральными процессорами (с использованием таких методов, как векторизация цикла), так и с графическими процессорами (хотя только с использованием CUDA) является сильной стороной этого решения. Тот факт, что Taichi нацелен на операции с конкретными структурами данных, позволяет ему создавать высоко оптимизированный и эффективный код, но в то же время ограничивает его потенциальные приложения. Будучи DSL, Taichi также имеет те же недостатки, однако тесная интеграция с C++ несколько смягчает их.

В дальнейшем Taichi развивался в нескольких различных направлениях, оставаясь при этом предметно-ориентированной технологией. Во-первых, это дальнейшее развитие физического моделирования и переход на Python-подобный язык (правда со статической типизацией) для описания алгоритмической составляющей моделей. Во-вторых, это дифференцируемое моделирование [55] и сжатие данных [56].

2.3 Резюме по существующим решениям

Технологии общего назначения не поддерживают достаточное количество аппаратных функций, что снижает производительность и энергоэффективность (например, PACXX, OpenACC, DVM, OpenCL, CUDA, TaskFlow). Промышленные API низкого уровня предоставляют такую поддержку, но разработка с их использованием чрезвычайно трудоемка (Vulkan, Metal, DirectX12). Предметно-ориентированные технологии и языки (Halide, Optix) являются хорошим решением как для графических процессоров, так и для других вычислительных систем (FPGA или ASIC). Однако их ключевой недостаток заключается в том, что алгоритмы и знания, реализованные с их помощью, трудно перенести в другие области и трудно интегрировать с остальным программным обеспечением, которое не использует язык, специфичный для предметной области.

Таким образом, вызов к технологии программирования заключается в том, чтобы обеспечить одновременно и доступ к специфическим функциям аппаратуры, и кроссплатформенность. Фундаментальное противоречие, вызванное этим требованием, необходимо решать. Наилучшие результаты в этом направлении, на наш взгляд, достигнуты в [29, 30, 34, 55].

3. Предлагаемое решение

Вместо создания универсальной технологии программирования для построения различных программных систем мы предлагаем расширяемую технологию, которая может быть настроена для определенного класса приложений. В отличие от, например, Halide [51-53], мы не используем предметно-ориентированный язык, а необходимые знания извлекаются из обычного кода на C++. Одним из основных преимуществ предлагаемой технологии является то, что исходный код не расширяется никакими новыми языковыми конструкциями или директивами. Предполагается, что исходный код является обычным исходным кодом на языке C++ (хотя и с некоторыми ограничениями), который может быть скомпилирован любым компилятором C++. Это значительно увеличивает возможность кроссплатформенной разработки с использованием предлагаемой технологии.

Мы используем сопоставление с образцом (pattern matching), чтобы находить *определённые шаблоны (или паттерны)* в C++ коде и преобразовывать их в эффективный код для GPU с использованием Vulkan. Шаблоны-паттерны выражаются через классы C++, функции-члены и отношения между ними.

Важно отметить разницу между предлагаемой технологией и большинством существующих технологий параллельного программирования, таких как DVM, CUDA, Taichi, Halide и другие: они расширяют язык программирования новыми конструкциями или предлагают новые языки, в которых параллельные конструкции отображаются в некоторую эффективную реализацию на определённом аппаратном обеспечении. Предлагаемый подход противоположен. Во-первых, мы не добавляем расширения для языка программирования, а, наоборот, вводим ограничения на использование его возможностей. Во-вторых, шаблоны-паттерны не определяют аппаратных функций, а выражают алгоритмические и архитектурные знания. Таким образом, за аппаратные функции отвечает транслятор, а не разработчик.

3.1 Шаблоны

Шаблоны (или паттерны) делятся на архитектурные и алгоритмические. Архитектурный шаблон определяет предметную область и поведение транслятора в целом (например, обработку изображений, трассировку лучей, нейронные сети, физическое моделирование и т. д.). Алгоритмические шаблоны выражают знания о потоках данных и управления и определяют более узкий класс алгоритмов, которые могут иметь эффективную аппаратную реализацию. Алгоритмические шаблоны могут встречаться внутри архитектурных шаблонов. Например, параллельная редукция, уплотнение (параллельное добавление элементов в буфер), сортировка, префиксная сумма, вычисление гистограммы, map-reduce и др.

Рассмотрим шаблоны, реализованные в текущей версии транслятора:

- Архитектурный шаблон *обработки изображений*. Этот шаблон предоставляет базовые возможности для т.н. GPGPU. Исходный код выглядит как обычный код на C++ с циклами в стиле OpenMP, за исключением того, что мы не используем директивы (прагмы). Вместо этого шаблон предполагает, что существует некий класс с управляющими функциями и функциями-ядрами (листинг 1). Функции-ядра содержат код внутри циклов, который предполагается переносить на GPU практически один в один. Управляющие функции — это функции, которые вызывают функции-ядра и, таким образом, определяют логику их запуска и привязку ресурсов (входных и выходных буферов и текстур). Функции-ядра имеют специальные префиксы (kernel1D_, kernel2D_, kernel3D_ и просто kernel_ в шаблоне трассировки лучей). Мы могли бы использовать для этого директивы компилятора или синтаксис [[kernel]] по аналогии с работами [36, 37]. Это хорошее решение, хотя на наш взгляд разница умозрительная. Используя префиксы, мы можем остаться в рамках стандарта C++ 98 и не вводить никаких расширения в язык, что было одной из наших целей.
- Архитектурный шаблон для *трассировки лучей*. Цель этого шаблона - предоставить доступ к трассировке лучей с аппаратным ускорением и эффективно вызывать виртуальные функции на графическом процессоре. Поэтому его можно рассматривать как кроссплатформенный аналог OptiX. Существенная разница между этим шаблоном и предыдущим заключается в том, что в шаблоне обработки изображений циклы предполагается размещать внутри функций ядра, в то время как в шаблоне трассировки лучей предполагается, что они находятся вне управляющей функции (и, следовательно, вне функций-ядер). Следовательно, шаблон трассировки лучей удобен, если для каждого потока или каждого обрабатываемого элемента данных используется сложный код, но цикл обработки данных прост. А шаблон обработки изображений удобен, когда количество потоков (обрабатываемых элементов данных) может изменяться во время работы алгоритма. Например, если нужно изменить размер изображения, можно обработать уменьшенную версию изображения, а затем снова масштабировать ее.
- Алгоритмический шаблон *параллельной редукции*. Для этого шаблона предлагаемое решение обнаруживает доступ к переменным данных класса (члены входного класса,

листинг 1) и генерирует код для параллельной редукции на GPU в соответствии с типом переменной и операций, участвующих в редукции (листинги 2 и 3).

- Алгоритмический шаблон *для параллельного добавления элементов в буфер* и связанный с ним *шаблон непрямого вызова ядра (indirect dispatch)*. Например, есть функция-член, которая обрабатывает входные данные и добавляет некоторые выходные данные в вектор через *«std::vector.push_back(...)»*. После чего выбранные данные обрабатываются в другой функции, при этом счетчик цикла зависит от *«vector.size()»*, и поэтому фактическое количество потоков на GPU должно быть другим: оно будет известно только после завершения первого ядра; поэтому здесь необходимо использовать непрямой вызов ядра.

В Листинге 1 показан пример входного кода, а в Листингах 2 и 3 – упрощенные примеры выходных данных.

```
1. class Numbers {
2. public:
3.   void CalcArraySumm(const int* a_data, uint a_dataSize) {
4.     kernel1D_ArraySumm(a_data, a_dataSize);
5.   }
6.   void kernel1D_ArraySumm(const int* a_data, size_t a_dataSize) {
7.     m_summ = 0;
8.     for(uint i=0; i<a_dataSize; i++) {
9.       int number = a_data[i];
10.      if(number > 0)
11.        m_summ += number;
12.    }
13.  }
14.  int m_summ;
15.};
```

Листинг 1. Пример исходного кода для архитектурного шаблона обработки изображений: вычисление суммы всех положительных чисел в массиве
Listing 1. Sample source code for an architectural imaging pattern: Calculate the sum of all positive numbers in an array

В примере Листинга 1 функция ядра и ее размеры (1D, 2D или 3D) извлекаются путем анализа имени функции (kernel1D_ArraySumm, строки 6-12). Управляющая функция извлекается путем анализа ее кода: если из этой функции вызывается хотя бы одна из функций-ядер, то это управляющая функция (CalcArraySumm, строки 3-5). Любые члены данных класса, к которым имеют доступ функции-ядра, помещаются в единственный «буфер данных класса» (m_summ, строка 14). Доступ для таких переменных анализируется дополнительно. Если в некотором ядре одна и та же переменная записывается на разных итерациях цикла определённым образом (строка 11), то генерируется параллельный код редукции в конце шейдера для этой переменной (листинг 3).

```
1. class Numbers_Generated : public Numbers {
2. public:
3.
   virtual void SetInOutFor_CalcArraySumm(VkBuffer a_dataBuffer) { ... }
4.   virtual void CalcArraySummCmd(VkCommandBuffer a_commandBuffer,
                                   uint a_dataSize) {
5.     m_currCmdBuffer = a_commandBuffer;
6.     vkCmdBindDescriptorSets(a_commandBuffer, ... , ArraySummLayout,
                               &allGeneratedDS[0], ... );
7.     ArraySummCmd(a_dataSize);
8.   }
9. protected:
10.  virtual void ArraySummCmd(size_t a_dataSize) {
11.    ...
```

```
12.     vkCmdBindPipeline   (m_currCmdBuffer, ...
    , ArraySummInitPipeline);
13.     vkCmdDispatch       (m_currCmdBuffer, 1, 1, 1);
14.     vkCmdPipelineBarrier(m_currCmdBuffer, ... );
15.     vkCmdBindPipeline   (m_currCmdBuffer, ... , ArraySummPipeline);
16.     vkCmdDispatch       (m_currCmdBuffer, a_dataSize/256, 1, 1);
17.     vkCmdPipelineBarrier(m_currCmdBuffer, ... );
18.   }
19. ...
20. }
```

Листинг 2. Получив некоторый класс в качестве входных данных, предлагаемое решение генерирует интерфейс и реализацию для GPU версии алгоритмов, реализованных в управляющих управления
Listing 2. Having received a certain class as input, the proposed solution generates an interface and implementation for the GPU version of the algorithms implemented in controllers

Из-за особенностей Vulkan необходимо сгенерировать две функции для каждой функции управления вводом. Для CalcArraySumm генерируются функции: SetInOutFor_CalcArraySumm и CalcArraySummCmd. Первая создает набор дескрипторов для входного буфера (a_dataBuffer в примере) и сохраняет его в allGeneratedDS[0]. Заметим, что параметр входного указателя a_data был удален. В сгенерированном коде параметры указатели в управляющих функциях и ядрах не используются, т.к. все данные теперь находятся на графическом процессоре, и доступ к ним осуществляется через наборы дескрипторов в шейдерах. В этом примере были сгенерированы два разных шейдера: ArraySummInitPipeline, который выполняет инициализацию цикла (обнуление суммы), а второй - ArraySummPipeline, который выполняет тело цикла (листинг 3).

```
1.  __kernel void kernel1D_ArraySumm(__global const int* a_data,
                                     __global NumbersData* ubo, ...) {
2.    __local int m_summShared[256*1*1];
3.    ...
4.    int number = a_data[i];
5.    if(number > 0)
6.      m_summShared[localId] += number;
7.    ...
8.    barrier(CLK_LOCAL_MEM_FENCE);
9.    m_summShared[localId] += m_summShared[localId + 128];
10.   barrier(CLK_LOCAL_MEM_FENCE);
11.   m_summShared[localId] += m_summShared[localId + 64];
12.   barrier(CLK_LOCAL_MEM_FENCE);
13.   m_summShared[localId] += m_summShared[localId + 32];
14.   m_summShared[localId] += m_summShared[localId + 16];
15.   m_summShared[localId] += m_summShared[localId + 8];
16.   m_summShared[localId] += m_summShared[localId + 4];
17.   m_summShared[localId] += m_summShared[localId + 1];
18.   if(localId == 0)
19.     atomic_add(&ubo->m_summ, m_summShared[0]);
20. }
```

Листинг 3. Пример сгенерированного шейдера для компилятора clspv
Listing 3. An example of a generated shader for the clspv compiler

В примере Листинга 3 исходное тело цикла преобразуется в строки 4–6. Видно, что доступ к m_summ был переписан на m_summShared [localId], который в дальнейшем используется при параллельной редукции в конце шейдера. В строках 13-17 реализован оптимизированный вариант параллельной редукции для графического процессора Nvidia, предполагающий, что размер подгруппы равен 32 потокам. Этот размер меняется в зависимости от входных параметров нашего транслятора. Например, можно отключить оптимизацию или задать меньший размер подгруппы (8 для мобильных графических процессоров) или использовать subgroupAdd вместо 13-17 строк (доступно только в GLSL).

3.2 Генерация кода

Предлагаемый генератор работает по принципу морфинга кода [57]. Суть этого подхода в том, что, имея определенный класс в программе и правила преобразования, можно автоматически сгенерировать другой класс с желаемыми свойствами (например, реализацию алгоритма на GPU). Правила преобразования определяются указанными шаблонами, в рамках которых осуществляется обработка и трансляция текущего кода. Сгенерированный класс наследуется от входного класса и, таким образом, имеет доступ ко всем данным и функциям входного класса.

Входной исходный код обрабатывается с помощью clang и libtooling [38]. Практически все задачи в предлагаемом трансляторе выполняются за два прохода. На первом проходе происходит обнаружение паттернов и их частей с помощью libtooling: вложенные циклы внутри функций ядра, редукция, доступ к членам данных класса и т. д. На втором проходе происходит переписывание исходного кода с использованием clang AST Visitors. Окончательный исходный код создается с помощью генерации текста по шаблону [58]. Таким образом, предлагаемое решение реализуется исключительно за счет source-to-source преобразования, т.е. в отличие, например, от Circle, который работает с LLVM-представлением кода. Хотя этот подход имеет определенные ограничения (нельзя изменить входной язык программирования, что возможно при использовании LLVM), он также имеет значительные преимущества:

I. Сгенерированный исходный код для обоих типов шейдеров (OpenCL C для clspv [35] или GLSL) и C++ код с вызовами Vulkan выглядят как обычный код, написанный вручную. Его можно отлаживать, изменять или комбинировать с другим кодом любым способом. Таким образом, в отличие от многих существующих технологий программирования легко отличить ошибки нашего транслятора от ошибок пользователя. Это проблема, например, для OptiX, Taichi или Circle, потому что непонятно, что технология программирования на самом деле делает с входным кодом.

II. Возможность генерировать исходный код для шейдеров дает значительную гибкость, потому что можно легко добавлять поддержку различного аппаратного обеспечения. Ранняя версия нашего инструмента использовала только clspv [35] для шейдеров. Однако мы быстро поняли, что возможностей clspv недостаточно для аппаратного ускорения трассировки лучей, массивов текстур (dynamic descriptor indexing), виртуальных функций и многого другого. Конечно, можно добавить такую поддержку в clspv, чтобы получить желаемые функции в SPIR-V из исходного кода шейдера OpenCL C, но этот путь дорогой и трудный, поскольку работа с исходным кодом SPIR-V и clspv требует специальных знаний и значительных усилий. В то же время добавить поддержку новых возможностей аппаратного обеспечения непосредственно в сгенерированном коде на GLSL относительно легко.

3.2.1 Передача данных CPU <=> GPU

Как упоминалось в обзоре, многие существующие решения автоматически решают проблему копирования данных. В большинстве случаев для программного обеспечения, использующего Vulkan, это нельзя делать по многим причинам. В основном, потому что копирование в Vulkan – нетривиальная операция, которая зависит от сценария использования алгоритма и возможностей аппаратуры. В предлагаемом подходе выполняется генерация кода для выполнения алгоритмов на графическом процессоре и для выполнения копирования, а затем пользователю предоставляется возможность самостоятельно вызывать этот код. Созданная функция с именем «UpdateAll» выполняет эту задачу. Если пользователю нужны данные обратно на CPU из некоторых внутренних данных сгенерированного класса, то можно создать новый класс, который наследуется от сгенерированного. В этом классе может быть реализован любой дополнительный алгоритм или функция копирования.

Предлагаемое решение реализует весь сгенерированный алгоритм на графическом процессоре, поэтому, как правило, все сгенерированные переменные и буферы находятся на графическом процессоре. Однако, поскольку сгенерированный класс наследуется от исходного, он также содержит все исходные переменные и векторы на CPU под их собственными именами. Пользователь либо предоставляет собственную реализацию копирования (через реализацию интерфейса т.н. “ICopyEngine” с несколькими виртуальными функциями типа Update/Read), либо использует предоставляемую нами реализацию того же интерфейса. Таким же образом пользователь может вручную удалить ненужные данные на CPU после вызова метода UpdateAll в классе, наследуемом от сгенерированного, а при необходимости возможно детальное управление копированием.

3.2.2 Сопоставление с ближайшими аналогами

Таким образом, ещё раз отметим, что наш инструмент по CPU коду генерирует его “зеркальный клон” на GPU, который пользователю нужно связать со своим кодом на Vulkan самостоятельно. Это более трудоёмко, чем подход CUDA или работы [37], но имеет и свои преимущества, поскольку не зависит от тяжеловесных инструментариев вроде UE4 и не порождает зависимости от себя самого в используемых проектах: сгенерированный код всегда можно дописать или переписать вручную и, таким образом, постепенно избавиться от нашего решения, если оно пользователя не устраивает. Кроме того, такой подход упрощает отладку и поиск ошибок: мы всегда можем понять в каком именно месте произошла ошибка – на стороне пользователя или в генераторе, т.к. выход представляет собой обыкновенный читаемый и отлаживаемый код на Vulkan и C++.

4. Экспериментальная оценка

Мы протестировали наш подход на нескольких приложениях, для которых мы генерировали различные реализации (GPU v1 – v3), используя разные опции транслятора. Эти опции отвечают за выбор различных реализаций одного и того же алгоритма на GPU. Поэтому в некоторых случаях производительность сильно отличается.

Рис. 1. Демонстрация применения разработанной технологии
Fig. 1. Demonstration of the application of the developed technology

Табл. 1 и 2 демонстрируют полученные результаты. Рис. 1 демонстрирует приложения, на которых мы тестировались.

В среднем на GPU мы достигали ускорения в 30-100 раз относительно многопоточной версии на CPU (14 ядер, OpenMP), что является адекватным показателем для рассматриваемых задач. Однако в данной работе мы не ставили своей целью обогнать какую-либо реализацию или технологию программирования. Важнее другое: производительность была разной для разных

сгенерированных реализаций и разных GPU. Это означает, что на практике для достижения желаемого уровня производительности разработчику приходится поддерживать все возможные версии реализаций на Vulkan вручную. В нашем трансляторе выбор между этими реализациями осуществляется автоматически простым переключением флагов.

Редукция (№1 – №3). В этих примерах мы демонстрируем возможность обнаруживать редукцию и генерировать параллельную редукцию на GPU с различными реализациями: GPU v1 представляет собой полностью кроссплатформенную реализацию, GPU v2 знает размер подгруппы (warp), а GPU v3 не только знает размер подгруппы, но и использует т.н. операции внутри подгруппы.

Задача N тел (№4). Это классическая проблема GPGPU с квадратичной сложностью, реализованная в рамках шаблона обработки изображений. В данном случае мы не применяли никаких специальных алгоритмических оптимизаций.

Тонирующий оператор **(*Bloom, №5*)** и денойзинг **(*Non Local Means, №6*).** В этих примерах мы демонстрируем возможность транслятора изменять используемую структуру данных для хранения изображений в шаблоне обработке изображений с буфера (GPU v1) на текстуру (GPU v2) и текстуру с половинной точностью (GPU v3). Интересно отметить что на использованной нами видеокарте Nvidia (табл. 1) реализация через буфер работает быстрее, а на использованной нами видеокарте от AMD победила реализация через текстуру (табл. 3).

Монте-Карло трассировщик путей (№7) демонстрирует перенос алгоритма трассировки путей на GPU с использованием аппаратного ускорения трассировки лучей несколькими способами: GPU v1 – реализация всего алгоритма в нескольких отдельных ядрах (при этом обе версии вызова виртуальной функции через конструкцию switch и через indirect dispatch имели приблизительно одинаковую производительность). GPU v2 – реализацию всего алгоритма с использованием т.н. “callable shaders” и GPU v3 – реализацию всего алгоритма в 1 ядре с преобразованием вызова виртуальной функции в конструкцию switch. В этом примере интересно отметить тот факт, что более простой способ через конструкцию switch был в 3 раза быстрее специально предназначенной для этого функциональности “callable shaders”. Дальнейший анализ (разд. 4.1) показал, что причина этого заключается в том, что реализованный нами код обработки материалов относительно несложный, и как следствие, вычислительные блоки GPU недогружены. В результате варианты GPU v1 и GPU v2, более интенсивно работающие с памятью, оказались существенно медленнее. Однако ситуация может поменяться в дальнейшем при расширении функциональности материалов и при увеличении количества типов материалов. Поэтому на наш взгляд нельзя сказать наперёд, какой именно из этих вариантов лучше. Предложенное решение позволяет разработчику поддерживать и тестировать несколько вариантов с низкой трудоёмкостью, переключая лишь флаги транслятора.

Табл. 1. Время выполнения в миллисекундах для различных (v1–v3) реализаций
Table. 1. Execution time in milliseconds for different (v1-v3) implementations

App/ Impl	CPU (1 core) Input source	CPU (14 cores) OpenMP	GPU v1 Ours	GPU v2 Ours	GPU v3 Ours
(#1) Int Arr. Σ	1.263 мс	0.271 мс	0.095 мс	0.089 мс	**0.084 мс**
(#2) FloatArr. Σ	1.420 мс	0.342 мс	0.104 мс	**0.096 мс**	**0.096 мс**
(#3) Sph. Eval.	39.73 мс	2.931 мс	0.399 мс	0.364 мс	**0.320 мс**
(#4) NBody	250400 мс	11920 мс	**118.0 мс**	-	-
(#5) Bloom	711.8 мс	52.74 мс	**0.733 мс**	1.420 мс	0.841 мс
(#6) NLM	88440 мс	6851 мс	422.0 мс	571.1 мс	**351.0 мс**
(#7) Path Trace	188.4 мс	14.928 мс	4.790 мс	1.360 мс	**0.460 мс**

Первая строка табл. 1 – задача вычисления суммы 1 миллиона целых чисел, изображённая на Листингах 1–3. Вторая строка – такая же сумма, но уже чисел с плавающей точкой, для которой транслятор генерирует дополнительные проходы после основного ядра (т.к. атомарных операций с плавающей точкой нет). Третья строчка — редукция для вычисления

сферических гармоник по изображениям. Эта задача схожа вычислению гистограммы. Четвёртая строчка – задача N тел, 5 и 6 строчки — упомянутые ранее фильтры обработки изображений. Последняя строчка — трассировка путей для изображения в разрешении 512x512 пикселей (256 тысяч путей). Для реализации трассировки пути на CPU мы использовали трассировку лучей из библиотеки Embree. Используемый CPU: Intel Core i9 10940X, графический процессор – Nvidia RTX 2080. Для трассировки пути было визуализировано изображение размером 512x512 (т. е. 256 тысяч путей). Измерения времени проводились при помощи Nvidia Nsight.

Табл. 2. Количество строк кода
Table 2. Number of lines of code

App/Lines	C++ (input source)	Vulkan (generated)	Vulkan (compact)	Vulkan (shaders only)
(#1) Int Arr. Σ	80	640	500	195
(#2) Float Arr. Σ	80	780	650	285
(#3) Sph. Eval.	120	1280	1000	445
(#4) NBody	115	850	700	140
(#5) Bloom	300	1460	1050	250
(#6) NLM	330	1290	1000	250
(#7) Path Trace	800	4500	3200	1470
Количество строк кода во входном примере (первый столбец), количество сгенерированных строк на Vulkan (второй столбец), оцененное некоторым образом, количество строк при ручном кодировании задачи (третий столбец) и количество строк кода в шейдерах (последний столбец).				

Табл. 3. Время выполнения в миллисекундах для различных (v1–v3) реализаций на двух видеокартах: AMD Radeon Vega 10 / Kirin 980 (мобильный GPU)
Table 3. Execution time in milliseconds for different (v1 – v3) implementations on two video cards: AMD Radeon Vega 10 / Kirin 980 (mobile GPU)

App/ Impl	GPU v1 Ours	GPU v2 Ours	GPU v3 Ours
(#1) Int Arr. Σ	0.24 мс / 1.51 мс	0.22 мс / 1.51 мс	**0.21 мс / -**
(#2) FloatArr. Σ	0.26 мс / 1.62 мс	0.26 мс / 1.61 мс	0.26 мс / -
(#3) Sph. Eval.	23 мс / 24 мс	22 мс / **22 мс**	**17 мс / -**
(#4) NBody	1260 мс / 768 мс	-	-
(#5) Bloom	20 мс / 273 мс	16 мс / 280 мс	**11 мс / 196 мс**
(#6) NLM	4510 мс / -	6000 мс / -	**2580 мс** / -
(#7) Path Trace	63.5 мс / 80 мс	-	**39.7 мс** / **52 мс**

Для примера #7 трассировка лучей была реализована программно, а вызываемые шейдеры на Vega 10 и Kirin 980 не поддерживаются. Измерения времени проводились при помощи таймера std::chrono. Фильтр Non Local Means не заработал на Kirin 980 из-за большого времени ожидания вычислительного шейдера.

4.1 Детали реализации вызова виртуальной функции на GPU

Вызов виртуальной функции является алгоритмическим шаблоном для нашего транслятора. Проблема, которую мы хотим решить – это эффективная реализация ветвлений на GPU для достаточно тяжёлого кода, когда потери на т.н. branch divergence становятся существенными.

В рассматриваемом примере трассировке путей реализованы четыре типа материалов: Ламберт, модель GGX [59] (чайник и зеркальные объекты справа от него), Смесь Ламберта и GGX по маске, вычисляемой при помощи так называемого шума Вороново [60] (цилиндр с розовыми прожилками), и самосветящийся материал с аналогом шумом Перлина [61] (источник света). Общий объём передаваемых в виртуальную функцию аргументов – 120 байт на один поток.

На примере этого подраздела мы демонстрируем анализ производительности, который можем производить для различных реализаций. Итак, мы рассмотрели три возможных способа реализации вызова виртуальной функции.

1) Сгенерировать конструкцию switch в коде вычислительного ядра. Этот способ наиболее тривиальный и универсальный.
2) Отсортировать потоки при помощи параллельного подсчёта гистограммы (где каждая ячейка гистограммы отвечает за количество объектов данного типа), после чего использовать последовательность непрямых вызовов (indirect dispatch).
3) Использовать функциональность Vulkan API “callable shaders”.

Для того чтобы вызов виртуальной функции можно было реализовать эффективно на GPU, мы вводим ряд ограничений и рассматриваем только специальный случай.

1) Один уровень вложенности. Большее количество уровней можно было бы поддержать (например, если всегда преобразовывать на последующих уровнях вызовы виртуальных функций в конструкцию switch), однако мы не исследовали это в нашей работе.
2) Объекты (относительно которых ведётся диспетчеризация) можно создавать, только выделяя их из пула. В настоящий момент предполагается, что пользователь создаёт объекты на CPU после чего они один раз загружаются на GPU путём перемещения всего пула на GPU в сгенерированной функции UpdateAll. Здесь необходимо сделать два важных уточнения:
   a. все данные для каждого объекта должны находиться в линейном и непрерывном участке памяти внутри пула;
   b. при необходимости память в пуле можно было бы выделать и на GPU (т.е. создавать новые объекты на GPU) при помощи параллельной префиксной суммы, однако мы не реализовывали этот сценарий.
3) Объекты нельзя удалять по отдельности. Можно лишь очистить пул целиком.
4) В текущей реализации может быть только функция-ядро целиком. При этом указатель на объект можно получить лишь специальным образом из другой функции ядра, которая формирует его по так называемому идентификатору объекта (Листинг 4).

```
TestClass::ControlFunc(uint tid, float4* out_color) {
  IMaterial* pMaterial = kernel_XXX(tid, ...);
  pMaterial->kernel_YYY_or_ZZZ(tid, ...);
}
```

Листинг 4. Пример того, как выглядит вызов виртуальной функции в пользовательском коде
Listing 4. An example of what a virtual function call looks like in custom code

На Листинге 4 kernel_XXX – это специальная функция-ядро-конструктор, которая внутри обязуется вызвать другую функцию “MakeObjPtr”, на которую уже реагирует наш транслятор. Функция kernel_YYY_or_ZZZ – виртуальная.

Мы измерили время выполнения итогового кода трассировки путей с различными реализациями вызова виртуальной функции и отдельно время на вызов виртуальной функции в тех случаях, когда это можно было сделать, используя Nvidia Nsight (табл. 4).

Табл. 3. Время выполнения
Table 4. Time of execution

Вариант реализации	Время на весь алгоритм	Время на вызов вирт.функ.	Накладные расходы
v1, 1 ядро, switch	**0.46 мс**	—	—
v2, множество ядер, switch	1.52 мс	0.22 мс	—
v2, множество ядер, indirect dispatch	1.72 мс	0.21 мс	0.20 мс
v3, callable shaders	1.36 мс	—	—
Для разрешения изображения 512x512. Мы использовали небольшой размер изображения, чтобы снизить нагрузку на память и улучшить работу L2 кэша в данном эксперименте, а итоговое изображение мы обходили блоками. При увеличении размера блока нагрузка на память в табл. 4 растёт.			

Табл. 5. Нагрузка на различные блоки GPU
Table 5. Load on various GPU blocks

Нагрузка на блоки GPU	switch (множество ядер)	indirect dispatch (множество ядер)	callable
VRAM ↓	**60.3%**	64.8%	—
L2 кэш ↓	22.5%	**21.3%**	—
L1 кэш ↓	10.4%	**7.0%**	—
SM↓	40.4%	**5.7%**	—
SM unused warps↓	30.5%	**9.0%**	—
SM compute warps↑	62.3%	**87.5%**	—
Память (VRAM), L1 и L2 кэши, загрузка вычислительных блоков мультипроцессора (SM) и среднее количество ожидающих подгрупп warp для него (SM unused warps). Поскольку время выполнения было практически одинаково для рассматриваемых вариантов (switch и indirect dispatch), лучше если при одном и том же временя выполнения загрузка блоков будет меньше. Это означает что теперь за то же самое время мы можем выполнить больше вычислений. Стрелками в первом столбце показано в какую сторону должно измениться число для того, чтобы мы могли рассматривать это изменение как улучшение.			

Кроме того, мы провели анализ различных вариантов сгенерированного кода на загрузку по отдельным блокам GPU на Nvidia Nsight (табл. 5), и на основе этого анализа можем сделать следующие выводы.

1) В рассматриваемом относительно простом примере трассировки путей быстрее всего оказался наиболее простой и универсальный способ через сгенерированную конструкцию switch.
   a. Причина этого в том, что в рассматриваемом примере трассировки путей вычислительные блоки всё ещё недогружены (40.4% в табл. 5), а при реализации через множество ядер память является узким местом (60.3% в табл. 5). Проще говоря, используемый нами GPU оказался слишком “прожорливым” в плане вычислений, и реализованный код с шумами не смог загрузить его настолько, чтобы вычисления стали узким местом.
   b. Для данного примера накладные расходы на подсчёт гистограммы в 0.2 мс слишком большие.
2) Тем не менее мы можем видеть, что при одном и том же времени выполнения ядра нагрузка на вычислительные блоки упала в 7 раз (с 40.4% до 5.7%), а количество ожидающих групп warp сократилось с 30.5% до 9% (аналогично количество занятых подгрупп warp выросло с 62.3% до 87.5%).

   a. Таким образом, эффективность выполнения этого участка кода в некотором смысле выросла, и теперь за то же самое время мы можем добавить больше вычислений для реализации обработки материалов. Детальный анализ этого феномена является предметом наших будущих исследований.

3) В рассматриваемом нами примере узким местом является память. После сортировки потоков объединение запросов к памяти работает, очевидно, хуже. Поэтому и улучшения по времени выполнения нет.
4) По своим временным характеристикам реализация через специально введённую функциональность вызываемых шейдеров (callable shaders) в Vulkan не была существенно лучше, чем реализация всего алгоритма трассировки путей через множество ядер (1.36 мс против 1.52 мс). Реализация же в одном ядре и конструкция switch существенно впереди (0.46 мс).
   a. Это свидетельствует о том, что нет единственно лучшего способа для реализации вызова виртуальной функции на GPU, поскольку несмотря на то что вызываемые шейдеры специально предназначены для решения проблемы ветвлений, даже при наличии достаточно тяжёлого кода вычисления шума [55] в исходном коде материалов, сохранение данных в память оказывается дороже.
   b. Тем не менее при дальнейшем росте сложности и функциональности обработки материалов вызываемые шейдеры скорее всего окажутся впереди, т.к. специально для этого разрабатывались.

5. Выводы

Мы предложили решение, способное снизить трудоёмкость реализации алгоритмов на Vulkan и GPU. На входе нашего решения обыкновенный C++ код с рядом ограничений, на выходе – C++ с необходимыми вызовами Vulkan API и кодом шейдеров (OpenCL C или GLSL). В процессе генерации кода могут применяться различные оптимизации для создания нескольких реализаций в зависимости от специфики проблемы и/или заданных конфигураций GPU. Сгенерированный код можно читать, отлаживать и использовать точно так же, как и код написанный на Vulkan вручную. Это позволяет нам легко отделять ошибки пользователей от ошибок генератора.

Наконец, анализ производительности на GPU – это вопрос исследовательский. Наши прогнозы о производительности той или иной реализации для обработки изображений, трассировки путей, вызова виртуальных функций (или чего-то другого) могут легко не сбыться. При этом такой анализ является достаточно трудоёмкой работой, которую не всегда можно выполнить для большой программы в разумных временных рамках. Предложенная нами технология программирования позволяет снизить трудоёмкость подобных экспериментов для достаточно большой кодовой базы за счёт автоматической генерации разных реализаций для одного и того же алгоритма.

6. Ограничения

У предлагаемой технологии есть два основных ограничения. Во-первых её использование возможно только профессиональными разработчиками на GPU, хорошо владеющими Vulkan. Предполагается, что разработчик связывает сгенерированный код на Vulkan со своим кодом вручную. Это более трудоёмко чем, например, разработка на CUDA, однако всё-же существенно менее трудоёмко, чем реализация сложных алгоритмов на Vulkan целиком вручную.

Во-вторых, на практике очень легко выйти за пределы шаблонов, предоставляемых текущей версией нашей системы, что делает невозможным её применение без постоянной адаптации под конкретный класс приложений или даже конкретное приложение. Мы видим в этой

адаптации новую “нормальность” для разработчиков на Vulkan так же, как, например, рано или поздно может потребоваться расширение любой используемой библиотеки в рамках обычной разработки на C++.

Список литературы / References

[1] J. Fang, Сю Huang et al. Parallel programming models for heterogeneous many-cores: a comprehensive survey. CCF Transactions on High Performance Computing, vol. 2, issue 4, 2020, pp. 382-400.

[2] OpenACC. URL: https://www.openacc.org/.

[3] N. Jacobsen. LLVM supported source-to-source translation. Translation from annotated C/C++ to CUDA C/C++. Master’s Thesis. University of Oslo, Norway, 2016, 134 p.

[4] G.D. Balogh, G.R. Mudalige et al. Op2-clang: A source-to-source translator using clang/llvm libtooling. In Proc. of the IEEE/ACM 5th Workshop on the LLVM Compiler Infrastructure in HPC (LLVM-HPC), 2018, pp. 59-70.

[5] P. Yang, F. Dong et al. Improving utility of GPU in accelerating industrial applications with user-centered automatic code translation. IEEE Transactions on Industrial Informatics, vol. 14, issue 4, 2017, pp. 1347-1360.

[6] J. A. Pienaar, S. Chakradhar, A. Raghunathan. Automatic generation of software pipelines for heterogeneous parallel systems. In Proc. of the International Conference on High Performance Computing, Networking, Storage and Analysis, 2012, pp. 1-12.

[7] Н.А. Коновалов, В.А. Крюков. DVM-подход к разработке параллельных программ для вычислительных кластеров и сетей / N.A. Konovalov, V.A. Kryukov. DVM-approach to the development of parallel programs for computing clusters and networks. 2002. URL: https://www.keldysh.ru/dvm/dvmhtm1107/publishr/dvm-appr-OpSys.htm (in Russian).

[8] В.А. Бахтин, А.В. Воронков и др. Использование языка fortran-dvm/openmp для решения больших задач. Труды Всероссийской суперкомпьютерной конференции «Научный сервис в сети Интернет: решение больших задач», 2008 г., стр. 185-191 / V.A. Bakhtin, A.V. Voronkov et al. Using the fortran-dvm / openmp language for solving large problems. In Proc. of the All-Russian Supercomputer Conference on Scientific Service on the Internet: Solving Big Problems, 2008, pp. 185-191 (in Russian).

[9] В.А. Бахтин, В.А. Крюков. DVM-подход к автоматизации разработки параллельных программ для кластеров. Программирование, том. 45, no. 3. 2019 г., стр. 43-56 / V.A. Bakhtin, V.A. Krukov. DVM-Approach to the Automation of the Development of Parallel Programs for Clusters, Programming and Computer Software, vol. 45, no. 3, 2019, pp. 121-132.

[10] M.A. Mikalsen. OpenACC-based Snow Simulation. Master’s Thesis. Norwegian University of Science and Technolog, 2013, 124 p.

[11] T. Öhberg. Auto-tuning Hybrid CPU-GPU Execution of Algorithmic Skeletons in SkePU. Master’s Thesis. Linkoping University, Sweden, 2018, 80 p.

[12] A. Ernstsson, L. Lu, C. Kessler. SkePU 2: Flexible and type-safe skeleton programming for heterogeneous parallel systems. International Journal of Parallel Programming, vol. 46, issue 1, 2018, pp. 62-80.

[13] M. Steuwer, P. Kegel, S. Gorlatch. Skelcl-a portable skeleton library for high-level GPU programming. In Proc. of the IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum, 2011, pp. 1176-1182.

[14] SYCL, cross-platform abstraction layer. URL: https://www.khronos.org/sycl/.

[15] SYCL for CUDA developers, examples, 2020. URL: https://developer.codeplay.com/products/computecpp/ce/guides/sycl-for-cuda-developers/examples.

[16] Vulkan specification, indirect dispatch command, 2021. URL: https://www.khronos.org/registry/vulkan/specs/1.2-extensions/man/html/vkCmdDispatchIndirect.html.

[17] A. Sherin. Resident Evil 2 Frame Breakdown, 2019. URL: https://aschrein.github.io/2019/08/01/re2_breakdown.html.

[18] T.D. Han, T.S. Abdelrahman. hiCUDA: High-level GPGPU programming. IEEE Transactions on Parallel and Distributed Systems, vol. 22, issue 1, 2010, pp. 78-90.

[19] J. Wu, A. Belevich et al. gpucc: an open-source GPGPU compiler. In Proc. of the IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2016, pp. 105-116.

[20] P. Sathre, M. Gardner, W. Feng. On the portability of CPU-accelerated applications via automated source-to-source translation. In Proc. of the International Conference on High Performance Computing in Asia-Pacific Region, 2019, pp. 1-8.

[21] HIP, C++ Runtime API and Kernel Language, 2021. URL: https://github.com/ROCm-Developer-Tools/HIP.
[22] A. Hamuraru. Atomic operations for floats in OpenCL – improved, 2016. URL: https://streamhpc.com/blog/2016-02-09/atomic-operations-for-floats-in-opencl-improved/.
[23] A. Kapoulkine. Getting Faster and Leaner on Mobile: Optimizing Roblox with Vulkan. 2019. URL: https://www.youtube.com/watch?v=hPW5ckkqiqA.
[24] Non-official Vulkan hardware database, 2021. URL: http://vulkan.gpuinfo.org/.
[25] N. Mammeri, B. Juurlink. VComputeBench: A vulkan benchmark suite for GPGPU on mobile and embedded GPUs. In Proc. of the IEEE International Symposium on Workload Characterization (IISWC), IEEE, 2018, pp. 25-35.
[26] V-EZ, an open source, cross-platform wrapper, 2018. URL: https://github.com/GPUOpen-LibrariesAndSDKs/V-EZ.
[27] Vuh, A Vulkan-based GPGPU computing framework, 2020. URL: https://github.com/Glavnokoman/vuh.
[28] Kompute, The general purpose GPU compute framework for cross vendor graphics cards, 2021. URL: https://github.com/KomputeProject/kompute.
[29] A. Rasch, R. Schulze, S. Gorlatch. Developing High-Performance, Portable OpenCL Code via Multi-Dimensional Homomorphisms. In Proc. of the International Workshop on OpenCL, 2019, article no. 4.
[30] T.-W. Huang, D.-L. Lin et al. Taskflow: A General-purpose Parallel and Heterogeneous Task Programming System. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2021.
[31] M. Haidl, S. Gorlatch. PACXX: Towards a unified programming model for programming accelerators using C++14. In Proc. of the Workshop on the LLVM Compiler Infrastructure in HPC, 2014, pp. 1-11.
[32] M. Haidl, S. Moll et al. Pacxxv2+ RV: an LLVM-based portable high-performance programming model. In Proc. of the Fourth Workshop on the LLVM Compiler Infrastructure in HPC, 2017, pp. 1-12.
[33] A. Sidelnik, S. Maleki et al. Performance portability with the chapel language. In Proc. of the IEEE 26th international parallel and distributed processing symposium, 2012, pp. 582-594.
[34] R. Baghdadi, J. Ray et al. Tiramisu: A polyhedral compiler for expressing fast and portable code. In Proceedings of IEEE/ACM International Symposium on Code Generation and Optimization (CGO), 2019, pp. 193-205.
[35] Google clspv. A prototype compiler for a subset of OpenCL C to Vulkan compute shaders, 2021. URL: https://github.com/google/clspv.
[36] S. Baxter. Circle C++ shaders, 2021. URL: https://github.com/seanbaxter/shaders.
[37] K.A. Seitz Jr, T. Foley et al. Unified Shader Programming in C++. arXiv preprint arXiv:2109.14682, 2021, 13 p.
[38] Clang documentation, 2021. URL: https://clang.llvm.org/docs/LibTooling.html.
[39] Thrust: a powerful library of parallel algorithms and data structures, 2021. URL: https://developer.nvidia.com/thrust.
[40] A. Kolesnichenko, C.M. Poskitt, S. Nanz. SafeGPU: Contract-and library-based GPGPU for object-oriented languages. Computer Languages, Systems & Structures, vol. 48, 2017, pp. 68-88.
[41] D. Beckingsale, R. Hornung et al. Performance portable C++ programming with RAJA. In Proc. of the 24th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2019, pp. 455-456.
[42] T. Heller, P. Diehl et al. Hpx – an open source c++ standard library for parallelism and concurrency. In Proc. of the Workshop on Open Source Supercomputing (OpenSuCo-2017), 2017, pp. 1-5.
[43] A. Paszke, S. Gross et al. Pytorch: An imperative style, high-performance deep learning library. arXiv preprint arXiv:1912.01703, 2019, 12 p.
[44] M. Abadi, A. Agarwal et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467, 2016, 19 p.
[45] T. Chen, M. Li et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274, 2015, 6 p.
[46] Nvidia OptiX, 2021. URL: https://developer.nvidia.com/optix.
[47] DirectML, 2021. URL: https://github.com/microsoft/DirectML.
[48] Wisp renderer, 2021. URL: https://github.com/TeamWisp/WispRenderer.
[49] Projects using RTX, 2021. URL: https://github.com.cnpmjs.org/vinjn/awesome-rtx.
[50] J. Hegarty, J. Brunhaver et al. Darkroom: compiling high-level image processing code into hardware pipelines. ACM Transactions on Graphics, vol. 33, no. 4, 2014, pp. 1-11.

[51] J. Ragan-Kelley, C. Barnes et al. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. In Proc. of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2013, pp. 519-530.
[52] R. T. Mullapudi, A. Adams et al. Automatically scheduling halide image processing pipelines. ACM Transactions on Graphics, vol. 35, no. 4, 2016, pp. 1-11.
[53] A. Adams, K. Ma et al. Learning to optimize halide with tree search and random programs. ACM Transactions on Graphics, vol. 38, no. 4, 2019, pp. 1-12.
[54] Y. Hu, T.-M. Li et al. Taichi: a language for high-performance computation on spatially sparse data structures. ACM Transactions on Graphics, vol. 38, no. 6, 2019, pp. 1-16.
[55] Y. Hu, Luke Anderson et al. Difftaichi: Differentiable programming for physical simulation. arXiv preprint arXiv:1910.00935, 2019, 20 p.
[56] Y. Hu, J. Liu et al. QuanTaichi: A Compiler for Quantized Simulations. ACM Transactions on Graphics, vol. 40, no. 4, 2021, pp. 1-16.
[57] S.S. Huang, D. Zook, Y. Smaragdakis. Morphing: Safely shaping a class in the image of others. Lecture Notes in Computer Science, vol. 4609, 2007, pp. 399-424.
[58] Inja, template engine for modern C++, 2021. URL: https://github.com/pantor/inja.
[59] W. Bruce, S.R. Marschner et al. In Proc. of the 18th Eurographics conference on Rendering Techniques, 2007, pp. 195-206
[60] Voronoi Noise, 2018. URL: https://www.ronja-tutorials.com/post/028-voronoi-noise/.
[61] Inigo Quilez. Fast 3D Noise, 2013. URL: https://www.shadertoy.com/view/4sfGzS.

Информация об авторах / Information about authors

Владимир Александрович ФРОЛОВ – кандидат физико-математических наук, старший научный сотрудник ИПМ РАН, научный сотрудник факультета ВМК МГУ. Сфера научных интересов: реалистичная компьютерная графика, моделирование освещённости, разработка программных систем оптического моделирования, параллельные и распределённые вычисления.

Vladimir FROLOV – PhD in computer graphics, senior researcher at Keldysh Institute of Applied Mathematics and researcher in computer graphics at Moscow State University. Research interests: realistic computer graphics, light transport simulation, elaboration of optical simulation software systems, GPU computing.

Вадим Владимирович САНЖАРОВ – младший научный сотрудник факультета ВМК МГУ, научный сотрудник ИПМ РАН. Сфера научных интересов: компьютерная графика, системы фотореалистичного синтеза изображений, параллельное и распределённое программирование.

Vadim SANZHAROV – junior researcher in computer graphics at Moscow State University and researcher at Keldysh Institute of Applied Mathematics. Research interests: realistic computer graphics, elaboration of optical simulation software systems, concurrent and distributed computing.

Владимир Александрович ГАЛАКТИОНОВ – доктор физико-математических наук, профессор, заведующий отделом компьютерной графики и вычислительной оптики. Сфера научных интересов: компьютерная графика, оптическое моделирование, создание программных систем оптического моделирования.

Vladimir GALAKTIONOV – Doctor of Science in physics and mathematics, Professor, Head of Computer graphics department. Research interests: computer graphics, optical simulation, elaboration of optical simulation software.

Александр Станиславович ЩЕРБАКОВ – аспирант 3-го года обучения факультета ВМК МГУ, ведущий разработчик в Gajin Entertament. Сфера научных интересов: компьютерная графика, системы фотореалистичного синтеза изображений, параллельное и распределённое программирование.

Alexander Stanislavovich SHCHERBAKOV – 3rd year postgraduate student at the Faculty of Computational Mathematics and Cybernetics, Moscow State University, Lead Developer at Gaijin Entertainment. Research interests: realistic computer graphics, elaboration of optical simulation software systems, concurrent and distributed computing.

DOI: 10.15514/ISPRAS-2021-33(5)-12

A Requirements Solution While Training Professional Project Leaders

[1] *V.A. Vargas-Pérez, ORCID: 0000-0002-8438-1239 <vanessa.atenea@gmail.com>*
[2] *L.S. Vargas-Pérez, ORCID: 0000-0001-7605-9779 <laura.silvia.vargas@gmail.com>*
[3] *A.F. Gutiérrez-Tornés, ORCID: 0000-0002-8634-9152 <afgutierrezt@uagro.mx>*
[2] *M. Soto-Hernández, ORCID: 0000-0002-8660-3413 <sotohana@gmail.com>*
[4] *E.M. Felipe-Riverón, ORCID: 0000-0002-9828-3568 <edgardo@cic.ipn.mx>*

[1] *Universidad Internacional Iberoamericana (UNINI)*
Calle 15 num. 36, entre 10 y 12, IMI III, Campeche, México, CP 24560
[2] *Instituto Tecnológico de Cd. Madero (ITCM)*
Av. Primero de Mayo 1610, Los Mangos, Cd Madero, Tamps., Mexico, 89460
[3] *Universidad Autónoma de Guerrero (UAGro)*
Av. Las Colinas 37b, Fracc. Las Playas, Acapulco de Juárez, Guerrero, México, 39390
[4] *Centro de Investigación en Computación del Instituto Politécnico Nacional (CIC IPN)*
Av. Juan de Dios Bátiz W/N, Gustavo A. Madero, Ciudad de México, México, PO 07738

Abstract. Globalization urges the necessity of having world class leaders, and very frequently in its formation, is required to teach them computational tools to mainly improve their project management facilities. In order to give project management a greater quality, a very important point is to control and follow up all users' diverse requirements, from the beginning to the end of each or any type of project: administrative, academic, engineering, software, video games, virtual and mixed reality, etc. This research shows a project requirements manager's approach which allows knowing the specific requirements in each of its phases and give an advanced description of its various types and their traceability. This prototype promotes project requirement management best practices, and link them to the various areas of engineering, administration, planning and also with society, industry, commerce and academic sectors.

Keywords: management; professional projects; training; leaders; engineering requirements

For citation: Vargas-Pérez V.A., Vargas-Pérez L.S., Gutiérrez-Tornés A.F., Soto-Hernández M., Felipe-Riverón E.M. A Requirements Solution While Training Professional Project Leaders. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 205-218. DOI: 10.15514/ISPRAS–2021–33(5)–12

Управление требованиями при обучении профессиональных руководителей проектов

[1] *В.А. Варгас-Перес, ORCID: 0000-0002-8438-1239 <vanessa.atenea@gmail.com>*
[2] *Л.С. Варгас-Перес, ORCID: 0000-0001-7605-9779 <laura.silvia.vargas@gmail.com>*
[3] *А.Ф. Гутьеррес-Торнес, ORCID: 0000-0002-8634-9152 <afgutierrezt@uagro.mx>*
[2] *М. Сото-Эрнандес, ORCID: 0000-0002-8660-3413 <sotohana@gmail.com>*
[4] *Э.М. Фелипе-Риверон, ORCID: 0000-0002-9828-3568 <edgardo@cic.ipn.mx>*

[1] *Международный латиноамериканский университет*
Кампече, Мексика, CP 24560
[2] *Технологический институт Сьюдад-Мадеро*
Сьюдад-Мадеро, Тамаулипас, Мексика, 89460
[3] *Автономный университет Герреро*
Акапулько-де-Хуарес, Герреро, Мексика, 39390
[4] *Центр вычислительных исследований Национального политехнического института*
Мехико, Мексика, 07738

Аннотация. Глобализация требует наличия лидеров мирового класса, и очень часто в процессе подготовки требуется научить их вычислительным инструментам, главным образом, для улучшения средств управления проектами. Чтобы повысить качество управления проектами, очень важным моментом является контроль и отслеживание разнообразных требований всех пользователей от начала до конца проекта любого типа: административного, академического, инженерного, программного обеспечения, видеоигр, виртуальной и смешанной реальности и т.д. Это исследование демонстрирует подход к управлению требованиями в проекте, позволяющий узнать конкретные требования на каждой из фаз проекта и дать подробное описание их различных типов и возможность их отслеживания. Наш прототип продвигает передовые методы управления требованиями к проектам и связывает их с различными областями проектирования, администрирования, планирования, а также с обществом, промышленностью, коммерцией и академическим сектором.

Ключевые слова: управление; профессиональные проекты; обучение; лидеры; инженерные требования

Для цитирования: Варгас-Перес В.А., Варгас-Перес Л.С., Гутьеррес-Торнес А.Ф., Сото-Эрнанде М., Фелипе-Риверон Э.М. Управление требованиями при обучении профессиональных руководителей проектов. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 205-218 (на английском языке). DOI: 10.15514/ISPRAS–2021–33(5)–12.

1. Introduction

Over the past decades, different types of projects failure are widely known. For example, not meeting deadlines, overflowing budgets, not corresponding functionalities with customer's specifications, and as a consequence, a negative impact has been obtained within entrepreneurs and investors.

It is common knowledge that when a project ends the deliverables are given to the client, assuming that all the needs and requirements requested have been met in accordance with signed contracts, but it turns out that the resulting products are not those that the users expected, so the final results are not satisfactory [1], [2], [3], [4].

As observed, there are common errors, such as not specifying agreements due to the informality with which the client's requests are treated. This has to do with the lack of correct detection, definition and formalization of client or users´ requirements.

We hereby propose to streamline the development process, reducing the costs of both the resulting products and the entire project, by establishing a designing model with four quality levels.

Objectives: These must be defined at the project beginning in order to formalize the results of each request submitted by clients or users and the holistically system to which they will be connected.

Problematic: In university training there are no internship programs supporting software tools for monitoring and controlling the multiple and diverse requirements within different types of projects. There are no tools allowing the linking, in an integral and quality way, the projects with other areas. Project management and evaluation tasks, within the study programs, are carried out using simple formats, so that subjective, deficient and minimal quality results are obtained [5], [6], [7].

To express users' requirements it is preferred to use the natural language method, even today when more exact instruments are known by software and computer systems engineers.

So, for any project that is carried out, the most important requirements source are the interviews between clients or users and systems analysts [2].

Justification: Software control tools are necessary in order to follow up and monitor project requirements, as well as other standardized instruments, which will allow to successfully complete the related projects, resulting in the strengthening of those companies linked to their development.

Companies manage to benefit through different skills and techniques, such as the following:

- Define in a more correct, clear and effective way the project´s requirements;
- Increase procedures efficiency in investment, development, quality, etc., in order to clearly visualize users' requirements and objectives;
- Use national and international standards adapting them to particular business realities;
- Apply standardized methods to raise the quality of both, projects and products in general.

2. Related Tasks

Requirements Engineering can be defined as: "the processes that aim to describe the precise users' needs and translate them into corresponding instructions, without any ambiguity, which might be, subsequently used in the development of the system" [8].

The organization, monitoring and control of projects requirements are considered as best practices concentrates, that provide knowledge of the various problems which must be solved, and also provide the best ways to solve them [9], [10], [11], [12].

The competencies related to carrying a good project general requirements definition are the following [13]:

- Identify and define the projects needs to be carried out;
- Analyze, evaluate and negotiate the projects needs with various users;
- Define and specify requirements according to detected needs; • Design the project requirements model according to its technical specifications;
- Validate those requirements defined in the model;
- Manage and implement the determined requirements.

Identify and define the project's needs: various interviews are carried out with the client, with diverse users and with all those involved in the projects, mainly with those who are experts, in order to know the company objectives, and the system or product to be created. It is about having the maximum knowledge of what is expected to be obtained as deliverables at the end of the projects, seeing how they will adapt to the client's needs, and later, knowing how they will be used, in order to detect in time the problems may arise.

Analyze, evaluate and negotiate the projects needs with various users: The needs become requirements when defined and grouped by categories and subsets. All are analyzed, one by one, in terms of their ambiguity, consistency, completeness and are classified according to the needs of the users, especially the client. Contradictory requirements are frequently requested, insisting that they are vital in meeting important needs. It is also very common for users to state that they need more than what should be done, according to the project.

Define and specify the requirements according to the detected needs: when the specifications are defined, flexibility is required to implement them. Then the effort needed to implement each requirement is estimated and its impact is assessed against the results delivery time and the project

cost. Iterative processes are used to combine, modify and eliminate the requested requirements according to the defined objectives. In large systems, a written document with graphic models must reflect the specifications of all defined requirements.

Specify and define the system requirements model: a requirements document is then formalized, containing the functions and characteristics descriptions of each system module to be created. It must also contain its development restrictions. Each system element is described in the specification, including all the information that enters and leaves it. A model is designed together with its blueprint to describe the system and its aesthetics, as well as to evaluate each component and its relationships, in order to determine and justify the requirements.

Validate those requirements defined in the model: it refers to the specifications of the system requirements examination to determine whether they contain or not omissions, ambiguities or inconsistencies. Also, to confirm the correction of detected errors and the system adapts regarding its processes and products, to established standards. Formal technical reviews are carried out by engineers, users, or clients to validate requirements

Manage and implement requirements: these are activities carried out by the project work team for requirements identification, monitoring and control. Management begins by identifying each requirement, to which a unique identifier is assigned. Its formalization is as follows: <type of requirement> <requirement #> <description>.

There are software tools for Requirements management, in the world market. Requirements engineering (RE) is essential to prevent major systems and projects from failure. The following are some representative statistics of the lack of RE and its negative impact on projects [13], [14]:

1) Poorly collected and incomplete requirements: 13.1%;
2) Users with little involvement in the project: 12.4%;
3) Allocate few resources: 10.6%;
4) Express unreal expectations: 9.9%;
5) Management little involved and interested: 9.3%;
6) Constant changes in specification: 8.7%;
7) A wrong planning: 8.1%;
8) The system is canceled, or is no longer required: 7.5%;
9) Poor management or lack of it: 6.2%;
10) Lack of technology: 4.3%;
11) Various factors: 9.9%.

As aforementioned, support tools are necessary to successfully complete projects, which is why true monitoring systems and requirements control are required.

Currently there are software systems available that require technical knowledge, not only for its potential application, but also for the proper use of the tool. They are widely recognized and disseminated, have broad support from the companies that develop them and made possible system functionalities improvement.

- Regarding CASE (Computer Assisted Software Engineering) tools, some focused upon requirements management are at disposal [15], [16], [17]:
- These tools are concentrated only to determine project software requirements and their development. Some of the specialized software that cover the described functions are: IRqA 3.0, DOORS ERS, CaliberRM, IBM Rational RequisitePro, etc.
- Project management success lies upon experience and organizing efforts through the use of world leading standards and institutions, such as International Project Management Association (IMPA) [18] and Project Management Institute (PMI) [19], [20], [21].
- IMPA is an organization based in England, focused in linked within the projects, so that they are carried out successfully [18].
- PMI is an international organization based in the USA, focused in performance and the dissemination of project management best practices and knowledge. It stands out in producing

the most recognized standard in this field, the Project Management Body of Knowledge (PMBOK) which provides basis, by guiding project leaders to successful results and achievements [19], [20], [21].

This proposal provides requirements monitoring and controlling guidelines for various types of projects, and is one of the firsts of its kind in Mexico [22], [23], [24].

3. Methodology

RE consists in applying the best techniques to collect users´ requirements, facilitating the understanding of what the client needs and wants, by analyzing its feasibility. Each specification must be validated. Requirements must be managed to negotiate, without ambiguity, so that they become a reasonable solution, an operational system [8], [17]. RE is based on Software Engineering Management, which can be adapted to any different type of project. Its phases are shown in fig. 1, [9], [11], [12], [17]. They are the following:

Feasibility Study: in this phase a complete report is carried out by the project development team evaluating whether the project is justified to be implemented.

Capture and Analysis: the project teams contacts end user in order to identify the services to be provided by the system and determine their performance level.

Specification: customers' requirements are transferred into a document, where each is precisely, completely and verifiably described, along with their respective hardware and system developing restrictions.

Validation: here each one of the detected requirements is verified and their specifications are approved and accepted. Finally, a management process must be executed in order to monitor and control requirements stability or volatility).

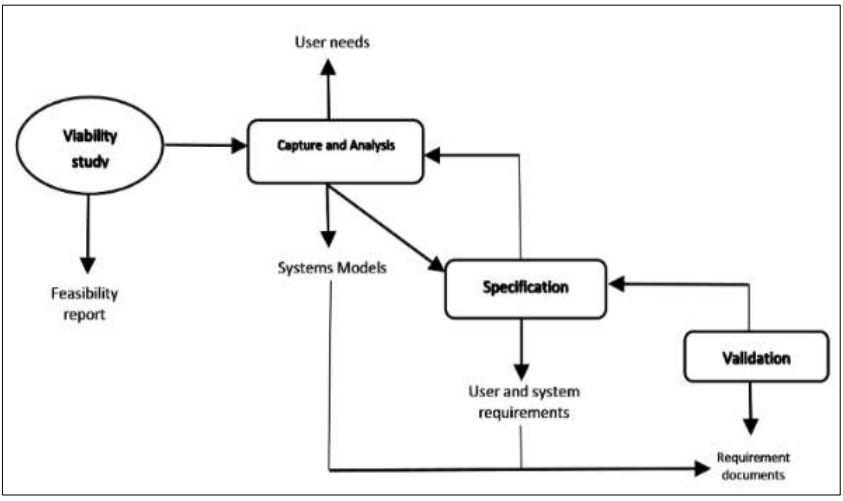


Fig. 1. RE phases

During a requirements selection process, for example, using the prototype model, the system validates those requirements obtained. A small negotiation may be required if several users are involved at the same time. The processes may have different variations, depending on the application`s nature (resources, uncertainty, risks, embedding, etc.), or on the project`s nature (tailor-made or market directed) [10].

Several techniques exist and are used for requirements specification: questionnaires, interviews and meetings, sampling, observation, examination of files, etc.

When requirements have been identified, a set of matrices for their monitoring is elaborated. They are called traceability matrices, which identify the requirements linked to the system or its

environment. Traceability matrices are frequently related as part of a database requirement, being useful to quickly access various requirements´ aspects affected by changes, of the system to be implemented [21].

Project management is a modern discipline, with a professionalization, that was born and grew in the USA military environment. It began in the sixties, when complex systems were needed to coordinate the joint work of teams from diverse disciplines. Today, project management has been implemented in all kind of institutions, as well as banks, industry, commerce, hospitals, pharmacies and also in advertising, accounting, construction, governments, etc.

Therefore, it is necessary to adapt the PMBOK standard to each project in a technical domain, backed by ANSI / PMI 99-001-2008 [17] and ISO 21500 [20] standards. The Project Management Professional (PMP) is considered the most important certification of its king and it is promoted by PMI and IMPA.

RE techniques, applied in each stage or phase are widely used, since they have many advantages (although also some disadvantages). A comparative table is presented below (Table 1) [9].

Table 1. Requirement Engineering (Re) Techniques

	Analysis of the problem	Evaluation and negotiation	Requirement specification	Validation	Evaluation
Interviews and Surveys	X				X
Brainstorming		X			X
Prototypes			X	X	
Hierarchical Analysis		X			X
Use Cases	X				X

Interviews vs. Use Cases. The information obtained during the various users´ interviews serve to build use cases, and through them the development team can better understand the environment involved. Analysts, when they need to understand a task, use Use Cases compiled from the questionnaires answers or delivered during the interviews, and use the diagrams to understand the system environment.

Brainstorming vs. Interviews. The ideas that are manifested by the working groups are mostly collected from the information gathered from interviews or questionnaires. While brainstorming tries to find answers to the difficulties that arose when understanding the client's requirements and expresses them in terms and concepts understandable by all participants, reaching a consensus.

Brainstorming vs. Use Cases. A list of ideas, obtained from user interviews, can be graphically represented by Use Cases means.

This proposal is oriented to requirements follow up and control. It allows a complete description of the different types of requirements and guarantee their traceability. It does so while following RE canons, including the conformation of Use Cases, Test Cases and other tools. The software so created on purpose integrates the main RE modules: Document Management, Traceability between work documents, Administration and Configuration, Reports and statistics [14], [22].

Classical and agile methodologies techniques are used, both in the design of the software tool, such as the RAD model, evolutionary prototyping and SCRUM, and RE techniques in the requirements organizer design, such as the monitoring of requirements traceability matrices [22].

This software system provides, in addition to its functionalities, the following:

- Optimal user's permissions
- Change control management improvement, registering the different generated versions (Configuration Management).

- Various reports generation: basics or specific, requested by different users and those resulting from advanced searches.

Requirements as a software development stage is confirmed by the processes determining, documenting and managing stakeholder's needs and expectations in order to meet the project objectives. Collecting requirements will provide the basis for defining and managing the project scope. The following figure represents the methodology to follow when managing project requirements, formed by the inputs, tools and outputs of these processes (fig. 2) [20].

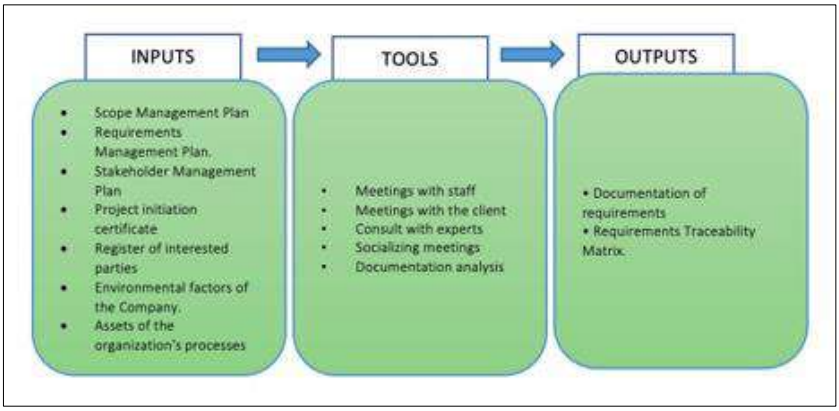


Fig. 2. Requirements Management Processes

Inputs: are made up by seven very important elements, which, through the transformation processes using the five here mentioned tools become the representation of the projects´ requirements, and their sub-processes, which will become the inputs, of the following management processes.

Outputs: correspond to the requirements documentation established from the requirements list. The second output of the process corresponds to the requirements traceability matrix. Both outputs will be part of the Scope Management processes of the project.

4. Results´ Discussion

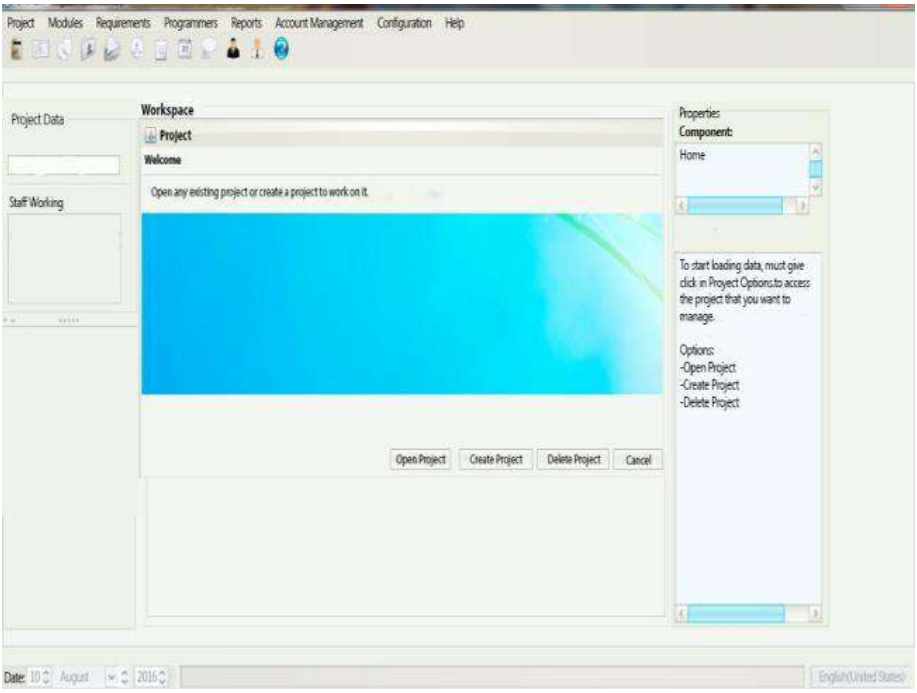
The proposed requirements organizer prototype is based on the RE and project management methodological model under standards guidance [13], [15], [18], [19], [20], [21], among others. This improve the negotiation and prioritization of certain requirements, making the corresponding reviews, which allow the determination of which requirements are urgent and should be implemented immediately, which are unnecessary and therefore should be eliminated, and which can be postponed.

This software prototype was developed in a responsive environment, with HTML5, CSS3 JavaScript and PH programming languages, in addition to a MySQL database manager that was installed on an HP ProLiant DL380 server with a Windows Server 2008 network operating system.

This software system is in its final phase as a prototype. It was designed and implemented so that different types of projects, from different areas, can optimized their requirement`s control. Its principal interfaces are shown form fig. 3-8.

Some teachers and students already tested this prototype in the Systems and Computing Department at ITCM. Its use has been well accepted by designers and business leaders.

The problem that managers or administrators face when developing a project is the lack of information provided by participants or stakeholders. An exact definition cannot be obtained and the difference between what is desired and what is required to be done is not established, resulting in a requirement misinterpretation and in addition, not monitoring the project objectives in detail.



Fig, 3. Prototype 1- Main screen

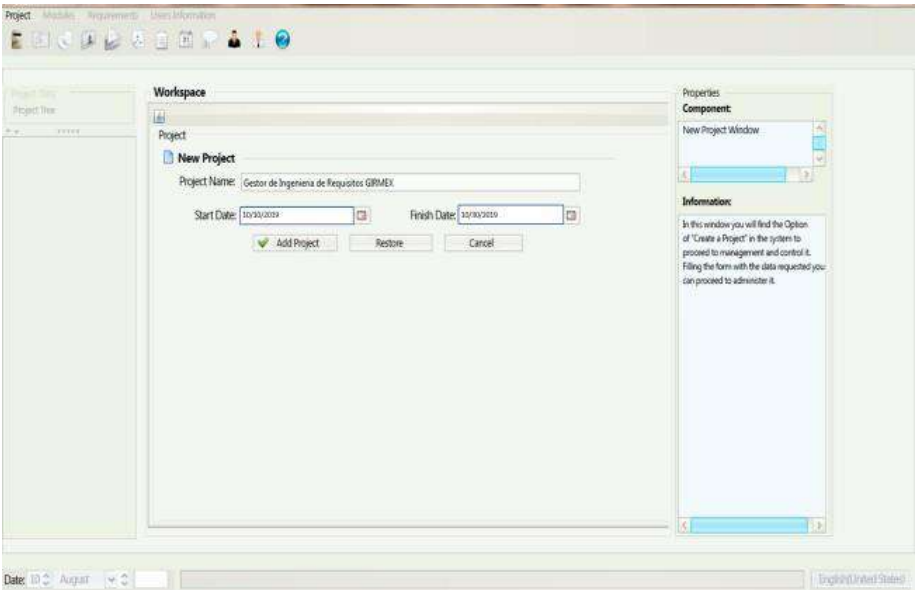


Fig. 4. Create a new project

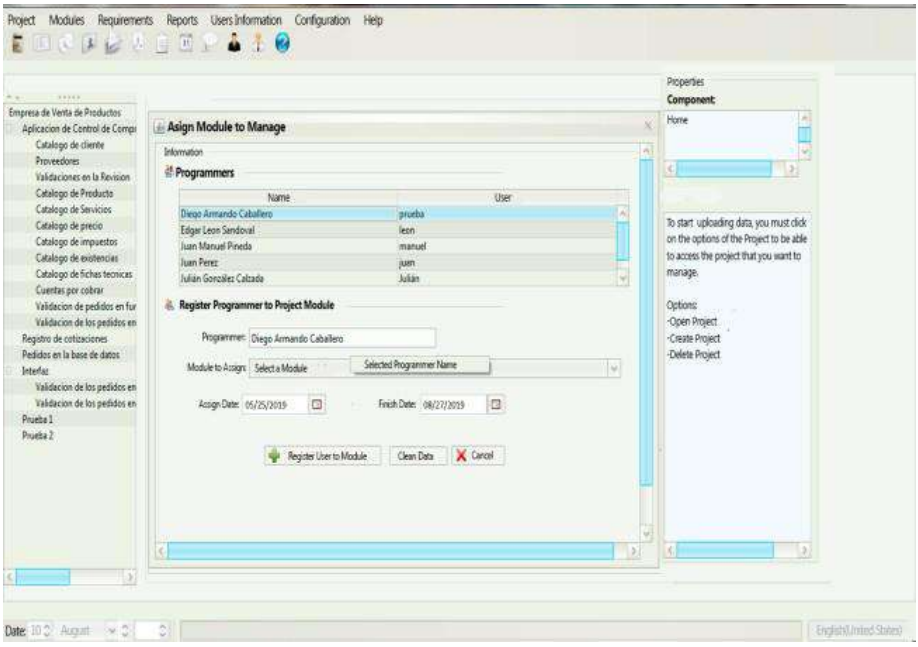


Fig. 5. Assign staff to the project

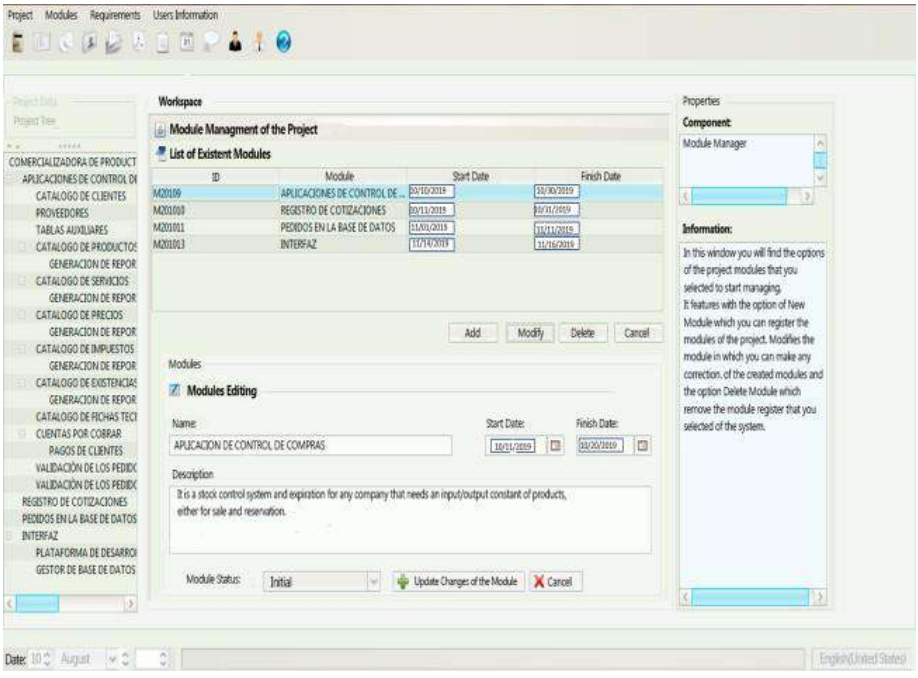


Fig. 6. Module Management of the project

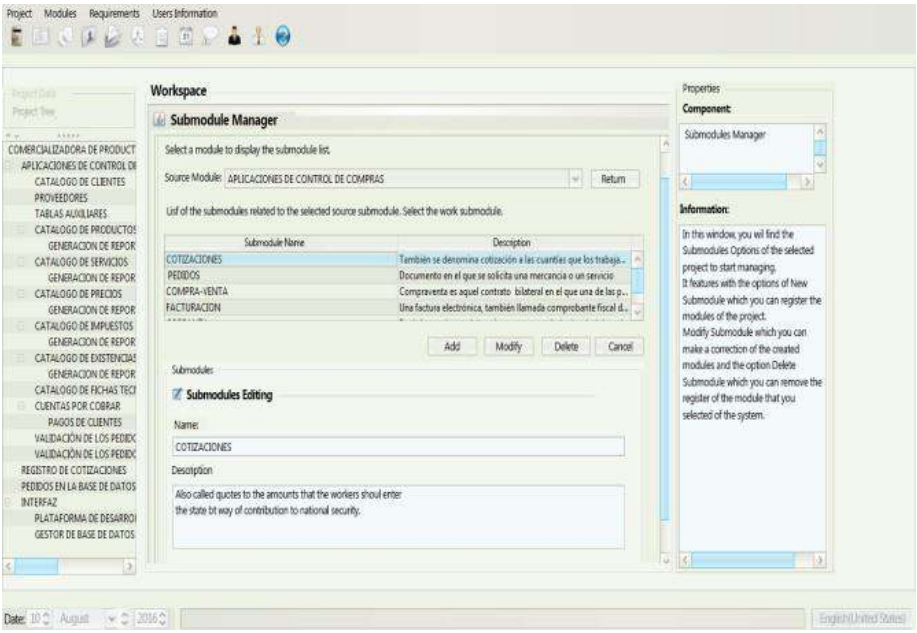


Fig. 7. Description of the project submodules

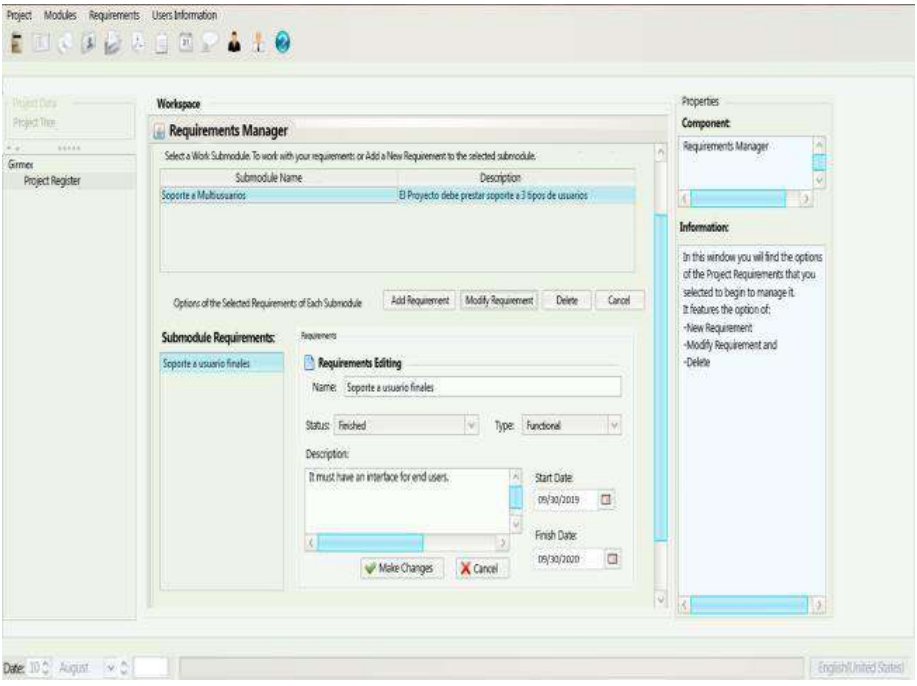


Fig. 8. Operations allowed in the requirements for modules and submodules of projects

Therefore, the construction of a requirements organizing system is proposed to solve the problems described. It is also intended to serve as a support for project leaders or managers, by providing detailed descriptions of each requirements and of the different projects sections of any kind, grouping them into four levels: modules, sub- modules, requirements and sub-requirements,

optimally and consistently; This will help to complete the projects, within the established deadlines, and deliver them with quality.

This tool could be useful, also, for investment projects, because it will guide its deliverance on time and within the assigned budget, according to the needs raised, which would increase the confidence of investors, customers and users.

Therefore, business relationships would have greater growth and maturity, which would facilitate the link between the different sectors.

Currently, tools that guide project leaders regarding their financial feasibility, but not technical feasibility, are common, although there are other tools that perform software project management. The software here proposed will help to track project requirements, designed in a four- level quality and precision model (module / submodule / requirement / sub requisite), which will be reflected in a correct definition of its technical feasibility, rather than its financial feasibility.

5. Conclusions

Requirements (RE) is a very important and standard methodology used worldwide. It requires the highest possible precision to determine all the application requirements. It will permit to avoid ambiguities, which is difficult, considering the pressures of the environment (time, costs). There are many drawbacks and challenges, such as the integration of domain or interface requirements, the functional and non- functional activities, and the evaluation of alternative specifications, among others.

According to requirements meetings results, it is necessary to obtain the correct information from what is said there, allowing ambiguity cancelation and as they are accurate, avoiding corrections and raising future costs. The negotiating and prioritizing requirements manners are reviewed as they help to define which requirements can be implemented as a priority, which can be changed, which are canceled, and which are postponed.

The here proposed tool creates a simpler way of doing the requirements tracking process, with greater effectiveness and consistency, regardless of the type of projects. It will allow them to be delivered within the budget and within the agreed term, thus the clients acquire greater confidence, which will tend to greater growth and maturity in their business relationships.

This proposal of a requirements organizer is designed to have better control and monitoring of project requirements, to contribute to the training of world-class project leaders of both research projects and other applications, to do so more completely, and contributing to deliver quality results.

This prototype can be installed in a multi-user environment, as a finished software product; in a WEB environment with a client-server architecture, which will contribute to make the process of monitoring and controlling requirements more effective, consistent and simple.

This tool allows the designer to create good teaching practices in the areas of Software Engineering, Requirements Engineering, Software Quality, Virtual and Mixed Reality, etc. It can also be used in video games, or in other engineering and undergraduate degrees.

References

[1] The Chaos Report (1994). The Standish Group, 1995. URL: https://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf.

[2] Goguen J.A., Linde C. Techniques for requirements elicitation. In Proc. of the IEEE International Symposium on Requirements Engineering, 1993, pp. 152-164.

[3] Duque N. Estadísticas de fallo en los proyectos y 10 factores clave para tener éxito / Project failure statistics and 10 key factors to be successful. URL: https://stratomindsblog.com/direccion_de_proyectos/estadisticas-de-fallo-en-los-proyectos-y-10-factores-clave-para-tener-exito/ (in Spanish).

[4] The Chaos Report 2015. The Standish Group, 2015. URL:

https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf.
[5] Christensen R., Eichhorn K. et al. Supporting learning leaders for the effective integration of technology into schools. Technology, Knowledge and Learning, vol. 23, issue 3, 2018, pp. 457-472.
[6] Clifford V., Montgomery C. Challenging Conceptions of Western Higher Education and Promoting Graduates as Global Citizens. Higher Education Quarterly, vol. 68, issue 1, 2014, pp. 28-45.
[7] Rico Chagollán M., García Herrera E., Rodríguez Campos J.C. Educación superior con tecnologías de la información para formar a los líderes del mañana. ANFEI Digital, no. 7, 2017, págs. 1-8 / Rico Chagollán M., García Herrera E., Rodríguez Campos J.C. Higher education with information technologies to train the leaders of tomorrow. ANFEI Digital, no. 7, 2017, pp. 1-8 (in Spanish).
[8] Nuseibeh B., Easterbrook S. Requirements Engineering: Roadmap. In Proc. of the Conference on The Future of Software Engineering, 2000, pp. 35-46.
[9] Bracket J. W. Software Requirements. SEI Curriculum Module SEI-CM-19-1.2, Carnegie Mellon University Software Engineering Institute, 1990, 31 p.
[10] Hofmann H.F. Requirements Engineering: A Survey of Methods and Tools. Technical Report 93.05, Department of Computer Science, University of Zurich, 1993.
[11] Gerlits E., Khoroshilov A., Kildishev D. Elicitation of Functional Requirements from the Application Programming Interface Documentation for Functional Testing. In Proc. of the Spring/Summer Young Researchers' Colloquium on Software Engineering, 2021, 10 p.
[12] Gutiérrez-Tornés A.F., Carrascal-Romero, L.A. Creando requerimientos excelentes. 1er Coloquio Internacional de Computación y Sistemas Inteligentes Empresariales, 2013, págs. 16-27 / Gutiérrez-Tornés A.F., Carrascal-Romero, L.A. Creating excellent requirements. In Proc. of the 1st International Colloquium on Computing and Intelligent Business Systems, 2013, pp. 16-27 (in Spanish).
[13] IEEE Recommended Practice for Software Requirements Specifications. IEEE Standard 830-1998, 1998, 40 p.
[14] Saiedian H., Dale R. Requirements Engineering: Making the connection between the software developer and customer. Information and Software Technology, vol. 42, issue 6, 2000, pp. 419-428.
[15] IEEE Standard for Developing Software Life Cycle Processes. IEEE Standard 1074-1991, 1992, 112 p.
[16] León-Martínez N.E., Gómez-Flórez L.C., Pimentel-Ravelo J.I. Herramienta computacional para la gestión y evaluación de proyectos software enmarcados en actividades de investigación. Scientia et Technica Año XVII, no 47, 2011, págs. 141-146 / Computational Tool for Management and Assessment of Software Projects Under Research Activities (in Spanish).
[17] Pressman R. Software Engineering: A Practitioner's Approach. 7th edition. McGraw-Hill Education, 2009, 928 p.
[18] Individual Competence Baseline for Project, Programme & Portfolio Management. Version 4.0. International Project Management Association (IPMA), 2015, 432 p.
[19] IEEE Guide – Adoption of the Project Management Institute Standard A Guide to the Project Management Body of Knowledge (PMBOK Guide) – Fourth Edition. IEEE Standard 1490-2011, 2011.
[20] ISO 21500:2012. Guidance on project management. ISO, 2012, 44 p.
[21] Project Management International. A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 6th edition. Project Management International, 2017, 756 p.
[22] Sommerville I. Software Engineering. 9th Edition. Pearson, 2010, 792 p.
[23] Pacheco Gómez E., Vargas Pérez V.A. y otros. Viabilidad y Comercialización de un Gestor de Requisitos para proyectos, con técnicas de Ingeniería de Requisitos (IR). 25° Encuentro Nacional de Investigación Científica y Tecnológica del Golfo de México, 2013, págs. 9-10 / Pacheco Gómez E., Vargas Pérez V.A. et al. Viability and Commercialization of a Requirements Manager for projects, with Requirements Engineering techniques. In Proc. of the 25th National Meeting of Scientific and Technological Research of the Gulf of Mexico, 2013, pp. 9-10 (in Spanish).
[24] Vargas Pérez L.S., Gutiérrez Tornés A.F. y otros. Gestión de proyectos: fundamental en la formación ingenieril para vincular a los diversos sectores. ANFEI Digital, no. 9, 2018, págs. 1-11 / Vargas Pérez L.S., Gutiérrez Tornés A.F. et al. Project management: fundamental engineering training to link the various sectors. ANFEI Digital, no. 9, 2018, pp. 1-11 (in Spanish).

Information about authors / Информация об авторах

Vanessa Atenea VARGAS-PÉREZ – PhD Student. Research interests: project management, requirement management.

Ванесса Атенеа ВАРГАС-ПЕРЕС – аспирант. Область научных интересов: управление проектами, управление требованиями.

Laura Silvia VARGAS-PÉREZ – Full Professor & Senior Researcher. Research interests: Software Engineering and Quality.

Лаура Сильвия ВАРГАС-ПЕРЕС – профессор и старший научный сотрудник. Область научных интересов: программная инженерия и качество программ.

Agustín Francisco GUTIÉRREZ-TORNÉS – PhD, Full Professor & Senior Researcher. Research interests: Software Architecture, Engineering Models and Quality.

Агустин Франсиско ГУТЬЕРРЕС-ТОРНЕС – кандидат наук, профессор и старший научный сотрудник. Область научных интересов: архитектура программного обеспечения, инженерные модели и качество.

Ana María SOTO-HERNÁNDEZ – PhD, Full Professor & Senior Researcher. Research interests: Software Engineering and Educational Sciences.

Ана Мария СОТО-ЭРНАНДЕС – кандидат наук, профессор и старший научный сотрудник. Область научных интересов: программная инженерия и педагогические науки.

Edgardo Manuel FELIPE-RIVERÓN – PhD, Senior Researcher. Research interests: Digital Image Processing, Digital Image Analysis.

Эдгардо Мануэль ФЕЛИПЕ-РИВЕРОН – кандидат наук, старший научный сотрудник. Область научных интересов: цифровая обработка изображений, анализ цифровых изображений.

DOI: 10.15514/ISPRAS-2021-33(5)-13

CC BY

Agile Software Development for Housing Reconstruction: The 2017 Earthquake case in Mexico

[1] *J. Ortiz-Hernandez, ORCID: 0000-0002-6481-1537 <javier.oh@cenidet.tecnm.mx>*
[1] *V.J. Ruiz-Martinez, ORCID: 0000-0001-6866-3642 <josue.rm@cenidet.tecnm.mx>*
[1] *M.Y. Hernandez-Pérez, ORCID: 0000-0002-8842-0899 <yasmin.hp@cenidet.tecnm.mx>*
[2] *R. Mijarez-Castro, ORCID: 0000-0001-8369-4092 <rmijarez@ineel.mx >*

[1] *The National Technological Institute of Mexico/Campus CENIDET, Interior Internado Palmira S/N, Col. Palmira, 62490, Cuernavaca, México*
[2] *National Institute of Electricity and Clean Energies (INEEL) Reforma 113, Palmira 62490, Cuernavaca, México*

Abstract. On September 19, 2017 an earthquake occurred in Mexico with epicenter in the limits of the states of Puebla and Morelos. It had a magnitude of 7.1 on the Richter scale. 31,090 homes were affected, of which 1659 with total damage. In order to attend to the contingency, the government of Morelos formed an inter-institutional committee in the first hours to carry out an initial diagnosis of the damage and to provide emergency services. This article presents a case study and lessons learned from the software engineering support for the development of a data-driven platform in the various phases of contingency response: census of damaged homes, identification of aid beneficiaries, determination of aid packages according to a damage assessment, logistics and follow-up of aid package delivery, data-driven decision making, and a public portal for open data and budget transparency.

Keywords: Decision-Making Platform; Agile Methodologies; Housing Reconstruction; Earthquake

For citation: Ortiz-Hernandez J., Ruiz-Martinez J., Hernandez Y., Mijarez-Castro R. Agile Software Development for Housing Reconstruction: The 2017 Earthquake case in Mexico. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 219-236. DOI: 10.15514/ISPRAS–2021–33(5)–13

Acknowledgements. This work has been partially funded by the TecNM 2021 and PRODEP CA-18 projects.

Гибкая разработка программного обеспечения для реконструкции жилья: землетрясение в Мексике в 2017 г.

[1] *Х. Ортис-Эрнандес, ORCID: 0000-0002-6481-1537 <javier.oh@cenidet.tecnm.mx>*
[1] *В.Х. Руис-Мартинес, ORCID: 0000-0001-6866-3642 <josue.rm@cenidet.tecnm.mx>*
[1] *М.Я. Эрнандес-Перес, ORCID: 0000-0002-8842-0899 <yasmin.hp@cenidet.tecnm.mx>*
[2] *Р. Миярес-Кастро, ORCID: 0000-0001-8369-4092 <rmijarez@ineel.mx >*

[1] *Национальный технологический институт Мексики/Кампус CENIDET, 62490, Мексика, Куэрнавака*
[2] *Национальный институт электричества и чистой энергии 62490, Мексика, Куэрнавака*

Аннотация. 19 сентября 2017 года в Мексике произошло землетрясение с эпицентром в пределах штатов Пуэбла и Морелос. Оно имело интенсивность в 7,1 баллов по шкале Рихтера. Пострадало 31 090 домов, из них 1659 были полностью разрушены. Для принятия мер в условиях чрезвычайной ситуации правительство Морелоса в первые же часы сформировало межведомственный комитет для проведения первоначальной диагностики ущерба и оказания экстренной помощи. В этой статье представлены тематическое исследование и уроки, извлеченные из поддержки разработки программного обеспечения для разработки платформы на основе данных на различных этапах реагирования на чрезвычайные ситуации: перепись поврежденных домов, выявление получателей помощи, определение пакетов помощи в соответствии с оценкой ущерба, логистика и отслеживание доставки пакетов помощи, принятие решений на основе данных, а также общедоступный портал для открытых данных и прозрачности бюджета..

Ключевые слова: платформа поддержки принятия решений; гибкие методы; реконструкция жилья; землетрясение

Для цитирования: Ортис-Эрнандес Х., Руис-Мартинес В.Х., Эрнандес-Перес М.Я., Миярес-Кастро Р. Гибкая разработка программного обеспечения для реконструкции жилья: землетрясение в Мексике в 2017 г. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 219-236 (на английском языке). DOI: 10.15514/ISPRAS–2021–33(5)–13.

Благодарности. Работа частично профинансирована проектами TecNM 2021 и PRODEP CA-18.

1. Introduction

On September 19, 2017 at 1:14 p.m. an earthquake occurred with an epicenter in the limits of Puebla and Morelos with a magnitude of 7.1 degrees on the Richter scale and a duration of one minute. 74 people died and 1,944 people required medical attention. Likewise, 31,090 homes were affected, of which 1,659 with total damage and the rest with varying degrees of damage.

A few hours after the seismic event, informal rescue networks and relief groups were set up in the main disaster areas, and at the institutional level, triage and emergency medical care teams and civil protection teams were dispatched to deal with the situation. There were also some spontaneous aid initiatives by the affected population itself, particularly in the most isolated rural areas.

For earthquake response there are official protocols directed to the population and also for the actions of state and municipal authorities. Unfortunately, however, when earthquakes occur in non-seismic zones, as was the case, neither the population nor the civil protection authorities were prepared.

During the first weeks there was no clearly identified entity to lead and coordinate the relief, medical assistance and reconstruction work. Each of the heads of the state government agencies, by official decree, took charge of the needs as a whole of each of the 33 municipalities, with the support of the municipal agencies of Health, Civil Protection, Education, etc.

This article addresses the support given by the state government's digital government office for housing reconstruction. The article does not describe the efforts made by other state offices and civil society organizations to attend to the contingency. Although the data obtained were widely used by

all these offices to meet the information needs detected in the areas of health, education, mobility, etc.

The objective and focus of this article are the description, from a software engineering point of view, of the process of developing a single comprehensive data platform for damaged housing, as well as the implementation of a management and logistics system to offer reconstruction assistance and to provide information to the population and federal authorities under the open data modality.

Due to the haste with which action had to be taken to address the contingency, it was not possible at that time to think of a properly planned development process. The imperative was to meet the demand and urgency of data that were being required by the various agencies, depending on the resources available to be able to obtain and deliver them.

Due to the urgency with which the contingency had to be dealt with, at the beginning there was no planned development process or system architecture. The imperative was to meet the demand for data required by the various agencies and with the resources available at the time at the various locations affected.

In relation to the subject of this article, most of the works found in the literature are related to various prevention proposals to mitigate damage in seismic zones.

In relation to the subject of this article, in [1], [2] we find several prevention proposals to a) mitigate damage in seismic zones, b) reduce risks to the population and c) reduce material damage.

In [3], [4], they point out the effectiveness of protocols to reduce losses and indirect costs such as interruption of productive activities and basic services and the effects of post-traumatic stress.

Other works [5], [6], [7], emphasize the complexity of disaster scenarios and the need for the different actors involved in damage repair to share unique information, in particular the use of mobile computing and social networks.

We did not find references of works to provide systematized and structured data to meet the needs throughout the various stages of the contingency, as in this case study.

2. Setting up of a specialized agency to support the contingency

In the first hours, the government of Morelos created an organization called Unidos por Morelos (UxM) with advisory, technical and legal support, coordination and planning units. Its objective was to implement the necessary actions for the reconstruction of the areas affected by the earthquake, through self-construction, citizen participation and priority attention and linking the needs of the affected population with the public, private and social sectors.

In order to have complete, reliable and timely data during the entire process of assistance to the affected population, the presidency of the Technical Committee of the UxM organization decided to have a technological arm for data support.

To address this need, UxM turned to the Digital Government office of the government of Morelos. The instruction was to support the process of planning and organizing the work from practically zero. It was the first experience of its kind, since Morelos had no history as a seismic zone.

3. The work team

After the formalization of the UxM organization and the integration of the digital government office, the development of the Integral Platform for Logistics and Decision Making Based on Data for the Reconstruction of Housing in the State of Morelos began. At first, no precise scopes or goals were specified, only the intention of accompanying the various UxM actors in their various information requirements and in parallel to integrate and interoperate the various systems that were built over the following months. In some cases, the systems were discarded, the data were backed up and new, and more appropriate systems were built. In other words, the platform specifications were obtained gradually over several months. In some cases, the same data, but with a different name and format,

were required by different authorities. There we had to reconcile with the various requesters in order to keep, as far as possible, a single type of name and format for a given piece of data.

It was a two-speed development, on the one hand those in charge of identifying information needs and formalizing them with the various requesting groups, and on the other hand, in a very distinct manner, the solution design and software development group.

3.1 Integration of the work team

The team was composed as follows:

- A project leader, interlocutor with the various UxM agencies. His responsibility was to ensure deadlines for the delivery of information or systems required, and to negotiate with the corresponding instances the requirements of personnel, equipment and facilities for the development of the project. Try to be the only visible face of the development team vis-à-vis UxM and other authorities.
- A software architect, in charge of ensuring the a) integration and interoperability of the systems created, b) the sizing and planning of the development effort, and c) the quality assurance of the data obtained. It is important to mention that as the platform grew, so did the amount of data store. It was necessary to constantly review the causes of incomplete or erroneous data. For example, there were people who had more than one CURP identity card, where it was necessary to search the database of the National Population Registry (RENAPO) and take the first CURP on the list, as well as cases in which more than one person sought to request support for the same affected house, etc. The main activity and commitment of the architect was very oriented to ensure the quality of the data.
- A senior developer, responsible for coordinating the programmers, both those assigned to the same area of digital government and those located in other units.
- Programmers, a variable group of 5 to 8 people divided into developers of the software logic, specific utilities such as web services, database design and configuration, user interfaces, etc.

Likewise, for the physical deployment of the systems in the territory of Morelos, the digital government office coordinated with various computer and telecommunications areas of the state government:

- The state agency of computing and telecommunications infrastructure, (i) to identify the areas with coverage of the state radio and internet telecommunications network, as well as the areas where commercial broadband internet and mobile data are available, (ii) to provide connectivity to a provisional office in each of the capitals of the 33 municipalities, (ii) to provide computing and network equipment for the applications that were being developed.
- The state public security agency, i) to provide personnel for the development of the first systems implemented in the affected municipalities, and ii) to install the developing versions of the systems in its data center before uploading them to the cloud for operation.

4. The development methodology used

Although several members of the team were familiar with agile methodologies, in particular Scrum. From the beginning it was difficult for us to apply it because the objective was not so much the development of a system but to meet very specific information needs. There was no client to show a prototype or a graphical interface, it was not the right time. The information was needed under any support, even preferably in spreadsheets. The client was rather ourselves, as we sought to obtain a single system that could house the various systems that were generated. In any case, we did agree to support the development in some agile methodology because of the need to respond quickly and systematize all our activities. Under these considerations, we found that the closest methodology was eXtreme Programming, better known as XP [6], [7], [8], [9], [10].

The following qualities of the XP methodology were taken into account, at first intuitively but later deliberately:

- Iterative and incremental development. Due to the lack of requirements specification because the UxM staff was only interested in having the data they needed as the contingency evolved. Within the work team we had to decide the type and level of granularity of the data and show it to know if it was the required data before starting any development.
- Pair programming. It was a very convenient strategy, if not the programming totally in pairs, it was very convenient to have two programmers supporting and complementing each other in the development of the code, the choice of algorithms, tests and documentation.
- Continuous unit testing. To ensure the integrity of the software and a robust development of the platform, it was established as a requirement to perform unit tests of all subsystems and modules.
- Periodic bug fixes. It was decided not to let bugs slip through for correction later. We had to be sure to have an error-free development for a correct integration of the platform's subsystems and modules.
- Integration of the programming team with the client. The project leader was in charge of interacting and establishing agreement with the authorities of the UxM organization. In some cases, he was only accompanied by the software architect to quickly agree on solution options and delivery deadlines. This decision was taken to expedite decision making.
- Simplicity, shared code ownership and code refactoring. These were some of the premises of the software development: to be lightweight and portable; to work together for the development of a platform of which we are all co-authors; and finally, to review the consistency and efficiency of the coding. Evolutionary software development projects that are prolonged over time, as in this case, involve different actors. If, in addition, we work against the clock and under pressure, we will have an accumulation of programming errors that it is important to control and correct in time [11], [12].

5. Technologies and standards implemented

Some of the technologies implemented were:

- The design of the platform using the following languages as appropriate: Java, PHP, Scalable Vector Graphics (SVG), HTML5 and CSS.
- Implementation of dynamic and interactive infograms in web browsers from the data of the affected houses using the D3.js library. This solution enabled the design of dashboards for monitoring and data-driven decision making.
- Responsive and adaptive web development allowing the display of information on desktops, laptops, tablets and smartphones.
- Implementation of biometric identification mechanisms such as registration and verification of fingerprints, for the validation of beneficiaries for the UxM.
- Implementation of the use of bank-type terminals for the download of points (material for reconstruction) according to the level of support assigned to the beneficiary.
- By having a centralized database and web access with low bandwidth requirements, access in general was very agile, for these purposes a cloud platform service was contracted with 99.99 availability, with mirror in the data center of the state public security agency.

Some of the standards and norms implemented were:

- Implementation of INEGI's own Technical Standard for Geographic Addresses for the survey of addresses of affected homes [13].
- Implementation of a geographic information system (GIS) for georeferencing affected homes. This allowed the precise location of affected homes on a map and the generation of heat maps and identification of the areas most affected by the earthquake.

- Implementation of the open data guide, for the cleaning and stabilization of data, as well as for the corresponding integration of the same, in the open data sets of the Government of the State of Morelos.

The objective was to build a data platform under a flexible, robust, secure and reliable architecture for UxM, and in turn feed a transparency portal aimed at the population of Morelos, an open data portal, as well as various dashboards with indicators for other state and federal government agencies.

6. Platform development stages

6.1 Rapid Census

Starting on the third day of the earthquake (September 22, 2017), a rapid census was initiated to obtain a diagnosis of the situation in the 33 municipalities of Morelos. A mobile application based on JotForm was developed with which 1,200 brigadistas classified homes with structural risk or located in risk zone, identified injured people as well as other basic needs of shelter, clothing and food [14]. The application was developed to store data locally when broadband internet or mobile data was not available on site. Once internet or mobile data was available, the data was synchronized with the central server. The first census teams were sent to the municipalities and communities where the main effects were known. The work was carried out by brigades deployed simultaneously in the 33 municipalities and coordinated by a command center made up of personnel from the state civil protection, the state security commission and the secretary of government, in conjunction with municipal civil protection authorities and personnel.

Fig. 1. A coordination meeting of UxM with the municipal authority of Huizilac

Features of the rapid census application:

- Standardized form for the census including name of the head of household, address, type of dwelling, data on injuries and other non-material damage.
- Photographic record of up to 10 images.
- Characterization of 11 types of damage and their qualification of partial or total damage.
- Georeferencing of damaged homes.

Fig. 2. Rapid census of damage to affected homes

This first development made it possible to quickly obtain coarse data; however, it had the disadvantage of making an open description of the addresses, without catalogs of roads, human settlements, or specifications of the types of damage. In rural areas, georeferencing can have an error of up to 100 meters. It should be noted that 80% of the homes in Morelos are located in rural areas, and most of this population does not even have an official address according to INEGI's Technical Standard on Geographic Addresses. In rural areas the specification of the address is very different than in urban areas. A lot of desk work was required to clean the information and export it to tables in a standardized database. However, these data configured a first map, which was very useful to start the next phases of this project.

6.2 The PVC ID Card System

To ensure the completeness and correctness of the affected housing census in a timely manner, from October 15 through November 2017, one month after the earthquake, the data previously captured through the JotForm application were migrated to a more robust database. Some specific utilities were developed for data cleaning, particularly for the elimination of duplicate records and removal of invalid records, among other errors.

Subsequently, it was decided to carry out a process of elaboration of PVC ID Cards for the beneficiaries and at the same time to carry out a new validation and completing of the data already obtained through the quick census. Three hundred volunteers were trained to prepare and deliver a PVC ID Card to each of the heads of household of the 31,090 affected households. This activity was carried out in public spaces set up in the 33 municipalities. As a result, an electronic file was created for each of the beneficiaries with their complete identity data, official identification code CURP validated by the National Population Registry (RENAPO), address with georeferencing according to INEGI's technical standard, scan of an official identity document, address reference, as well as photographs and an official damage report. It is worth mentioning that of the total of 31,090 homes affected, the National Disaster Fund assisted 15,586 homes and the remaining 15,504 were assisted by UxM. Both datasets were integrated into our platform to avoid duplication and to follow up on both types of support.

The kit for the elaboration of the Plastic PVC Card was composed of a lap top to capture the personal data of the beneficiary and the affected home, a scanner printer to digitalize the official identification

of the head of household and proof of address, a camera to take a photograph and a scanner of the fingerprint of the owner of the affected home. All integrated to make and deliver the plastic PVC Card with barcode and ID associated to the affected person in only 3 minutes. Likewise, two sheets of paper are printed in duplicate: the official request for assistance of the affected person and the delivery of the aid package, documents that must have a handwritten signature for legal reasons. The beneficiary receives a copy of these two documents. It is important to mention that in order to offer greater certainty to the beneficiaries and to give greater legality to the procedure, it was decided to recur to the printing of documents and to request autographic signatures, although originally it was considered a totally digital development.

Fig. 3. Web capture sheet for the elaboration of the Plastic PVC Card, 1 of 2

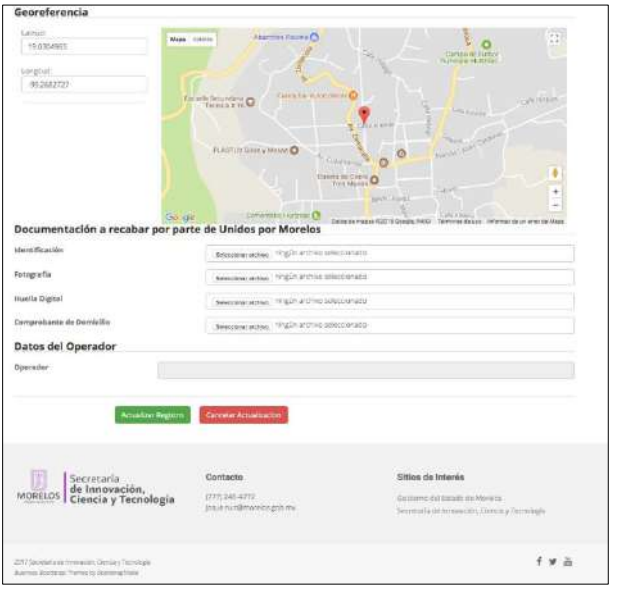


Fig. 4. Web capture sheet for the elaboration of the Plastic PVC Card, 2 of 2

Information obtained:

a) identification and biometric data of the beneficiary and at least two landline or cell phone numbers,

b) data on the dwelling: address according to INEGI's Technical Standard on Geographic Addresses, including its georeference by means of a map; 3 photographs of the damage; characterization of the damage according to 11 types of damage defined by civil engineering experts and the qualification of the damage according to the following criteria: minor damage, partial damage, severe damage and total damage.

   The responsive web application was developed in PHP and Java with MySQL database. It is basically a formulary to integrate data, digital documents and images. To ensure the quality of the data, specific validation mechanisms were incorporated for each type of data and, where possible, catalogs were incorporated to facilitate data entry, see fig. 3 and 4.

A training program was carried out for all personnel in charge of credentialing. The work team consisted of capturers, a supervisor in charge of resolving doubts of the people who came to the module and a technician to resolve connectivity problems, equipment configuration and to supply consumables, fig. 5.

Fig. 5. Training for the personnel in charge of Plastic PVC Card elaboration

This application allowed data capture and delivery of Plastic PVC Cards simultaneously in the 33 municipal stations. A calculation was made to determine the number of Plastic PVC Card production stations per municipality, taking into account the number of homes affected. All digital files were integrated into a central database. Only in some of the municipal modules with deficient internet access was a server installed to concentrate the information obtained, to later integrate it into the central database.

Fig. 6. Elaboration and delivery process of the Plastic PVC Card in all 33 municipalities

The development of this system took 2 and a half months, started on October 15 and concluded in December 2017. Although it began to be used to deliver Plastic PVC Cards on November 8, 2017 (fig. 6-8).

F*ig. 7. Data capture and delivery of the Plastic PVC Card*

Fig. 8. A Plastic PVC Card delivered to a beneficiary

6.3 Technical Assessment System

Its purpose was to program on-site attention by volunteer civil engineers from the National Polytechnic Institute, to review and complement the first damage survey of the rapid census, diagnose the level of damage to housing and propose repair solutions for self-construction to those affected. It was developed in 3 stages, corresponding to three modules of this system. It had as

input the information captured from the Plastic PVC Cards system, in particular the data related to the characterization of housing damage:

Stage 1: Module for scheduling and assigning emergency personnel. Based on the information generated by the Plastic PVC Cards system, the scheduling of visits by paramedics, civil protection personnel and civil engineers for the care of injured people and people at risk was generated.

Stage 2: On-site technical evaluations module. Scheduling of visits by civil engineering experts to assess damage to each of the affected homes using a mobile application to determine the types of damage and prioritize their attention. In cases of total damage to schedule routes and workloads of demolition and debris removal teams.

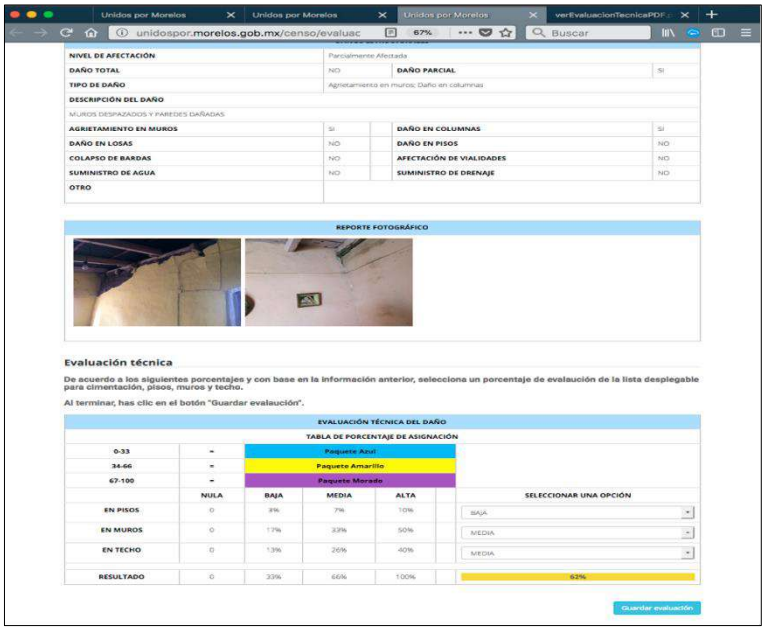


Fig. 9. Web capture sheet of the technical evaluation system

Stage 3: Module that allows remotely determining material assistance packages for reconstruction based on housing damage. The aid packages are established on the basis of a damage assessment based on data captured in the census relating to 11 types of structural damage, a free description of the damage, as well as 3 to 7 photographs of the damage. It was designed with the support of civil engineers specialized in structures. It uses heuristics based on artificial intelligence to assign 4 types of aid packages in construction materials, which are yellow package for minor damage, blue package for moderate damage, purple package for severe damage and green package for major reconstruction, and whose materials are economically equivalent to $15,000, $30,000, $45,000 and a variable amount of up to $60,000 Mexican pesos. For homes with total damage, i.e., not recoverable, 5 different foundations were used to build new homes with amounts ranging from $145,000 to $160,000. For damage to buildings in 11 housing units, structural evaluation techniques were used and aid was assigned depending on the type of building and other technical considerations that automatically result in the corresponding aid package. It is important to mention that for each of the houses with total damage, a complete electronic file was created, with information and scans of official documents such as demolition permits, structural report, documents of legal ownership of the property, etc. Finally, 253 damaged houses located in risk areas were identified, where the experts' recommendation was to relocate them to safer areas. See fig. 9.

It is important to mention that in order to carry out the damage assessments it was necessary to have complete information. In the event of not having enough elements for the assessment, for example, if the photographs were not clear or the description of the damage did not correspond to the photographs, an on-site visit was scheduled to verify data or to take the photographs again. These activities were carried out in coordination between UxM agency staff and personnel from the municipality

6.4 Operational Coordination System

This system was developed to plan, organize and coordinate the supply and delivery of materials to the beneficiary population, as well as to follow up on the reconstruction of each of the houses affected by total or partial damage based on the damage diagnosis. The platform has a module that allows budgeting, scheduling for purchase and managing the reception of construction materials from suppliers in the state of Morelos and neighboring states. It also generates distribution logistics in the 33 distribution centers, one for each municipality in Morelos.

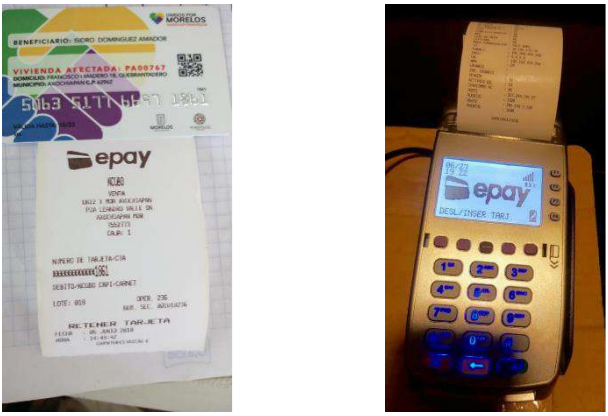


Fig. 10. Point-of-sale terminal for delivery control of construction materials

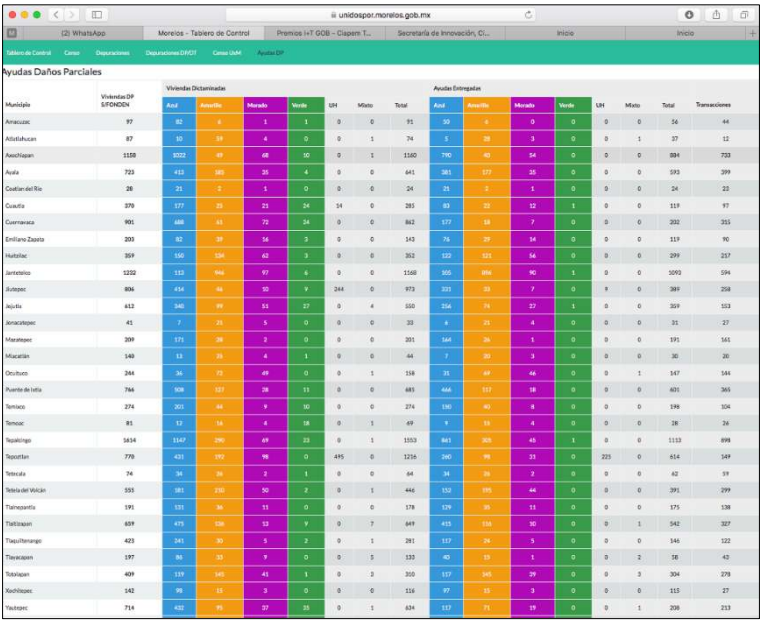


Fig. 11. Material delivery progress control sheet by municipality

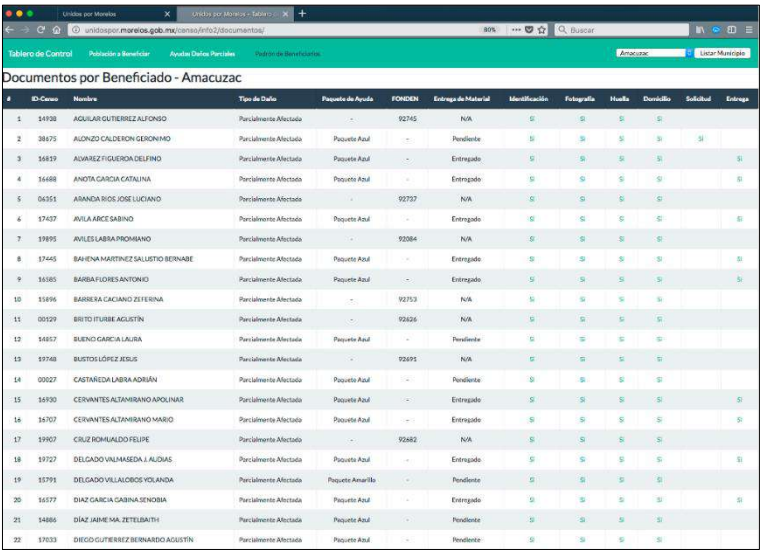
Through the platform, a person responsible for logistics in each municipality schedules appointments with the beneficiaries to go to the distribution center to receive their materials. The aid packages consist of cement, sand, blocks, rebar, paint, among other materials. Using point-of-sale terminals, the material delivered is downloaded from the balance of the Plastic PVC Card and an inventory control is kept at each distribution center, see fig. 10.

At the central level, there is access to the platform for monitoring the delivery of materials. We delivered 15,501 material aid packages for the reconstruction of affected homes in Morelos. Likewise, those affected were provided with advice and support from civil engineers for the reconstruction of their homes at all times, see fig. 11.

This system started to be developed since October 15, 2017, a first version was released on November 19, 2017, and continued its development and improvement for several months, its final version was completed on May 15, 2018.

6.5 Follow-up system for UxM representants

Each of the 33 UxM representatives in each of the municipalities had web access to a control panel to verify the progress of the delivery of material packages to the affected people. The dashboard provides information on the types of packages delivered and to be delivered, damaged houses classified by total and partial damage. Whether FONDEN support is available or not. They can consult the information corresponding to their municipality: access the registry of each of the damaged houses and review their location, damage data, the affected person, as well as review the integration of the file that includes identification of the affected person with fingerprint and photograph, as well as photographs of the damage. The data can also be viewed or exported to a file as a report. See fig. 12.



#	ID-Censo	Nombre	Tipo de Daño	Paquete de Ayuda	FONDEN	Entrega de Material	Identificación	Fotografía	Huella	Domicilio	Solicitud	Entrega
1	14938	AGUILAR GUTIERREZ ALFONSO	Parcialmente Afectada	-	92745	N/A	SI	SI	SI	SI		
2	38675	ALONZO CALDERON GERONIMO	Parcialmente Afectada	Paquete Azul	-	Pendiente	SI	SI	SI	SI	SI	
3	16819	ALVAREZ FIGUEROA DELFINO	Parcialmente Afectada	Paquete Azul	-	Entregado	SI	SI	SI	SI		SI
4	16488	ANOTA GARCIA CATALINA	Parcialmente Afectada	Paquete Azul	-	Entregado	SI	SI	SI	SI		SI
5	06351	ARANDA RIOS JOSE LUCIANO	Parcialmente Afectada	-	92737	N/A	SI	SI	SI	SI		
6	17437	AVILA ARCE SABINO	Parcialmente Afectada	Paquete Azul	-	Entregado	SI	SI	SI	SI		SI
7	19895	AVILES LABRA PROMIANO	Parcialmente Afectada	-	92084	N/A	SI	SI	SI	SI		
8	17445	BAHENA MARTINEZ SALUSTIO BERNABE	Parcialmente Afectada	Paquete Azul	-	Entregado	SI	SI	SI	SI		SI
9	16585	BARBA FLORES ANTONIO	Parcialmente Afectada	Paquete Azul	-	Entregado	SI	SI	SI	SI		SI
10	15896	BARRERA CACIANO ZEFERINA	Parcialmente Afectada	-	92753	N/A	SI	SI	SI	SI		
11	00129	BRITO ITURBE AGUSTÍN	Parcialmente Afectada	-	92626	N/A	SI	SI	SI	SI		
12	14857	BUENO GARCIA LAURA	Parcialmente Afectada	Paquete Azul	-	Pendiente	SI	SI	SI	SI		
13	19748	BUSTOS LÓPEZ JESUS	Parcialmente Afectada	-	92691	N/A	SI	SI	SI	SI		
14	00027	CASTAÑEDA LABRA ADRIÁN	Parcialmente Afectada	Paquete Azul	-	Pendiente	SI	SI	SI	SI		
15	16930	CERVANTES ALTAMIRANO APOLINAR	Parcialmente Afectada	Paquete Azul	-	Entregado	SI	SI	SI	SI		SI
16	16707	CERVANTES ALTAMIRANO MARIO	Parcialmente Afectada	Paquete Azul	-	Entregado	SI	SI	SI	SI		SI
17	19907	CRUZ ROMUALDO FELIPE	Parcialmente Afectada	-	92682	N/A	SI	SI	SI	SI		
18	19727	DELGADO VALMASEDA J. ALDIAS	Parcialmente Afectada	Paquete Azul	-	Entregado	SI	SI	SI	SI		SI
19	15791	DELGADO VILLALOBOS YOLANDA	Parcialmente Afectada	Paquete Amarillo	-	Pendiente	SI	SI	SI	SI		
20	16577	DIAZ GARCIA SABINA SENOBIA	Parcialmente Afectada	Paquete Azul	-	Entregado	SI	SI	SI	SI		SI
21	14886	DÍAZ JAIME MA. ZETELBAITH	Parcialmente Afectada	Paquete Azul	-	Pendiente	SI	SI	SI	SI		
22	17033	DIEGO GUTIERREZ BERNARDO AGUSTÍN	Parcialmente Afectada	Paquete Azul	-	Pendiente	SI	SI	SI	SI		

Fig. 12. Progress in the delivery of construction material in the municipality of Amacuzac

This system began its development in January 2018, in its first version was completed on February 15, 2018 and continued its improvement and update until its final version on May 15, 2018.

6.6 A Data-driven decision-making system

It was developed with the purpose of offering in real time an updated and reliable source of information for the operational and technical directorates of UxM, for the internal control body and for the various agencies of the government of Morelos involved such as the State Evaluation Commission, the Comptroller's Office, the Office of Social Development and the Office of Public

Works and Services. In addition, this system feeds data to the state open data portal and the citizen transparency portal.

This system started its development in January 2018, in its first version was completed on February 15, 2018 and continued its improvement and upgrade until its final version on May 15, 2018.

6.7 Transparency and Accountability Portal and Open Data Portal

The platform built made it possible to feed the Transparency and Accountability Portal of the Morelos government's Treasury Office as well as the Morelos State Open Data Portal. These portals were built by instruction of the federal government to offer the general population data of public interest generated by the various government agencies. In particular, the Transparency and Accountability Portal offering detailed information on the application and expenditure of budgetary resources and the Open Data Portal with detailed information on the georeferencing of the damages by type of damage and the progress of reconstruction. See figures 13 and 14.

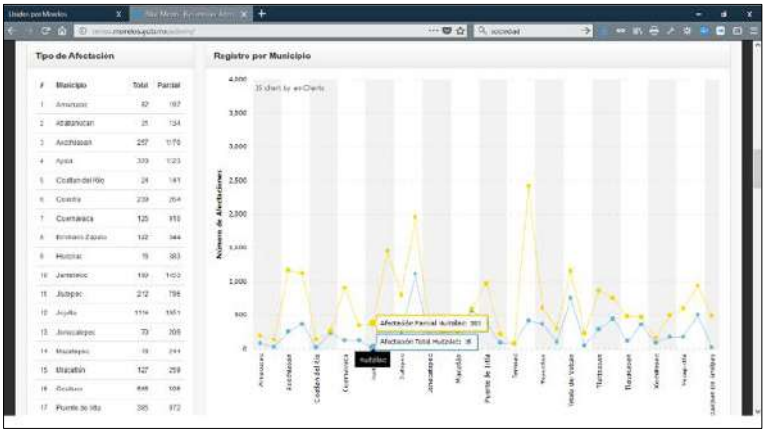


Fig. 13. Graphical description of types of housing damage by municipality

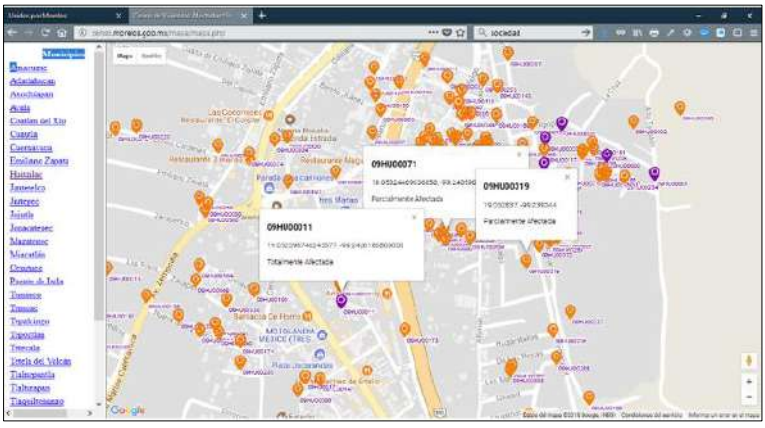


Fig. 14. Deployment of housing damage in the municipality of Huitzilac

7. *Conclusions*

The development of this platform was fundamental to support all activities related to the seismic contingency. From the first days of this unfortunate event until the closing of the UxM agency's activities in mid-2018, work on updating and improving it never stopped. There were always new needs to be addressed throughout this time, in particular special reports to the office of the governor of Morelos or to federal government authorities.

In this article we show the most relevant methodological aspects of the development of the platform. That is to say, the aspects that concern the support given to the UxM organization to have at all times the data for its planning, management and decision-making activities. In this sense, for us it was not so important the development of this platform, but to be in a position to provide data in the format and at the time they were required as the attention to the reconstruction of the affected houses evolved.

The development team had to solve many problems on the fly, not only technical, but also of communication, coordination and in general of organization to establish agreements between all the parties involved. For example, when for the same project we need to serve different types of users with different types of requirements, it happens that the concepts to name the same object or the same event may have different meanings, may have different levels of urgency and may vary the level of granularity of the information they need about that object or event.

Some of the main lessons learned were the following:

- It is essential that in the meetings for requirements gathering or project follow-up, all users and key stakeholders must be present, we have to ensure that this is the case.
- It is very important to establish and define exactly each of the terms and ensure that everyone understands them well.
- It is necessary not to compromise, due to urgency, the delivery of new versions of the developed systems.
- The systems area must be recognized as a strategic and allied area to provide useful information and not as a group of programmers subordinated to do only what they are asked to do.

Some examples of problems that appeared on the fly and that were addressed were the following:

- When there was never full agreement of the civil engineering experts to generate the catalog for damage characterization, it was necessary to conciliate with them by showing them proposals until an authority declared that the last proposal presented remained.
- When there was no standard for the survey of addresses, nor data for the georeferencing of the affected houses. After an analysis of the INEGI Technical Standard for geographic addresses, we had to develop a web application that implements it.
- When there was no way to obtain the information directly from the developed systems at the time, it was often necessary to export the data to a spreadsheet, adapt it and deliver it to the requesting entity. Users do not care about getting the information directly from the platform, they care about having the information at the time and in the format, they need it.

The development was in charge of the internal staff of the Digital Government Office of Morelos. There was no external contracting. It was entirely based on free software. Only 100 kits were purchased for the Plastic PVC Card process, which were reused internally at the conclusion of this project.

We conservatively consider that a contract to carry out all the support for the development of a platform of this type would have cost around 20 million pesos. But the most important thing is that it was an entirely customized development that will become part of the assets of the government of Morelos. This platform received 3 national awards:

- First Place National PREMIOS I+T GOB EDITION 2018, State Category, Technological Innovation Award, granted by the Committee of Informatics of the State and Municipal Public Administration A.C. [15].
- National Digital Government Award u-GOB 2018, State Category, granted by the specialized magazine U-GOB Technology in Government [16].
- 4th place National Award "The Most Innovative" in its 2018 edition awarded by Innovation Week Magazine [17].

Finally, to say that out of a total of 31,090 homes damaged as a result of the earthquake, 15,586 had support from the National Disaster Fund FONDEN and 15,504 had support from the UxM agency. A total of 13,592 construction materials were provided to repair partially damaged homes. 1,659 homes with total damage were rebuilt through 5 foundations with 50% state participation and 253 homes in high-risk areas were relocated to safer areas.

We believe that the support of this digital platform was fundamental to ensure coordination between all federal, state and municipal agencies involved, as well as some civil society organizations, in particular to share a single database, with views and privileges for all actors involved for an opportune attention to the reconstruction of homes damaged by the earthquake.

References

[1] Basaglia A., Aprile A. et al. Assessing community resilience, housing recovery and impact of mitigation strategies at the urban scale: a case study after the 2012 Northern Italy Earthquake. Bulletin of Earthquake Engineering, vol. 18, issue 13, 2020, pp. 6039-6074.

[2] Roudbari S., Heris M. et al. Mediating Design Claims: The Social Media and Housing Disaster of the 2017 Halabja Earthquake, Natural Hazards Review, vol. 21, issue 2, 2020.

[3] Lallemant D., Soden R. et al. Post-Disaster Damage Assessments as Catalysts for Recovery: A Look at Assessments Conducted in the Wake of the 2015 Gorkha, Nepal, Earthquake. Earthquake Spectra, vol. 33, issue 1_suppl, 2017, pp 435-451.

[4] Mannella A., Di Ludovico M. et al. Analysis of the Population Assistance and Returning Home in the Reconstruction Process of the 2009 L'Aquila Earthquake. Sustainability, vol. 9, no. 8, 2017, article no. 1395. DOI 10.3390/su9081395.

[5] Platt S, Reconstruction in Chile post-earthquake. ReBuillDD field trip September 2011. Cambridge Architectural Research, 2012, 43 p.

[6] Gil M. Why use a project management methodology? Available at: https://nae.global/en/why-use-a-project-management-methodology/, accessed April 10, 2021.

[7] Maarif M.Y., Shahar S.M. et al. The Challenges of Implementing Agile Scrum in Information System's Project. Journal of Advanced Research in Dynamical and Control Systems, vol. 10, issue 09-Special, 2018, pp. 2357-2363.

[8] A. Srivastava, S. Bhardwaj, and S. Saraswat. SCRUM model for agile methodology. In Proc. of the International Conference on Computing, Communication and Automation (ICCCA), 2017, pp. 864-869.

[9] Khalid A., Butt S.A. et al. Agile Scrum Issues at Large-Scale Distributed Projects: Scrum Project Development at Large. International Journal of Software Innovation (IJSI), vol. 8, issue 2, 2020, pp. 85-94.

[10] Girma, N. M. Garcia, and M. Kifle. Agile Scrum Scaling Practices for Large Scale Software Development. In Proc. of the 4th International Conference on Information Systems Engineering (ICISE), 2019, pp. 34-38.

[11] Asri S.A., Sunaya I.G.A.M. et al. Web Based Information System for Job Training Activities Using Personal Extreme Programming (PXP). Journal of Physics: Conference Series, vol. 953, 2017, pp 27–28.

[12] Duran M., Juáres-Ramirez R. et al. User Story Estimation based on the Complexity Decomposition using Bayesian Networks. Programming and Computer Software, vol. 46, issue 8, 2020, pp. 569-583.

[13] Norma tecnica sobre domicilios geograficos. Instituto nacional de estadistica y geografia, 2010, 16 p. / Technical standard on geographic addresses. National Institute of Statistic and Geography, 2010. 16 p. (in Spanish).

[14] Online form builder Jot Form. Available at: https://www.jotform.com/es/integrations/platform/mobile-app, accessed April 10, 2021.

[15] Comité de Informática de la Administración Pública Estatal y Municipal A.C. (CIAPEM), Premios I+T Gob Edición 2018 / Computing Committee of the State and Municipal Public Administration A.C. (CIAPEM), I + T Gob Awards 2018 Edition. Available at: https://www.ciapem.org/itgob2018/proyectos-ganadores-premios-it-gob-edicion-2018/, accessed April 10, 2021 (in Spanish).

[16] Laboratorio de Innovación U-GOB, Gobierno Digital: Proyectos ganadores de la 4a Entrega de los Premios u-GOB, 20 marzo, 2019 / U-GOB Innovation Laboratory, Digital Government: Winning projects of the 4th u-GOB Awards, March 20, 2019. Available at https://u-gob.com/gobierno-digital-proyectos-ganadores-de-la-4a-entrega-de-los-premios-u-gob/, accessed April 10, 2021 (in Spanish).

[17] Innovation Week Magazine, Premio las más Innovadoras 2018 / Innovation Week Magazine, Most Innovative Award 2018, Available at: http://www.lasmasinnovadoras.com/sectorpublico/pdf/Innovadoras-SP-2018.pdf, accessed April 10, 2021 (in Spanish).

Information about authors / Информация об авторах

Javier ORTIZ-HERNÁNDEZ, PhD in Informatics and Automatic, Full Professor. Research interests: Process Modeling and Requirements, Digital Government, Information Systems.

Хавьер ОРТИС-ЭРНАНДЕС, кандидат наук в области информатики и автоматики, профессор. Область научных интересов: моделирование процессов и требования, цифровое правительство, информационные системы.

Víctor Josue RUIZ-MARTÍNEZ, Master of Sciences in Information Systems. Research interests: Web and Mobile App Development, Digital Government, Information Systems.

Виктор Хосуэ РУИС-МАРТИНЕС, магистр наук в области информационных систем. Область научных интересов: разработка веб-приложений и мобильных приложений, цифровое правительство, информационные системы.

María Yasmín HERNÁNDEZ-PÉREZ, PhD in Computer Sciences, Full Professor. Research interests: Intelligent Tutoring Systems, Data Mining, Affective modeling.

Мария Ясмин ЭРНАНДЕС-ПЕРЕС, кандидат компьютерных наук, профессор. Область научных интересов: интеллектуальные системы обучения, интеллектуальный анализ данных, моделирование эмоций.

Rito MIJAREZ-CASTRO, PhD in Electrical and Electronic Engineering, Researcher and Group leader. Research interests: Digital signal processing, Embedded systems, Wave propagation.

Рито МИЯРЕС-КАСТРО, кандидат наук в области электротехнической и электронной инженерии, исследователь и руководитель группы. Область научных интересов: цифровая обработка сигналов, встроенные системы, распространение волн.

DOI: 10.15514/ISPRAS-2021-33(5)-14

Математические модели и методы расчета процессов, сопровождающих обледенение летательного аппарата

[1,2] *И.А. Амелюшкин, ORCID: 0000-0002-4281-3531 <Amelyushkin_Ivan@mail.ru>*
[2] *М.А. Кудров, ORCID: 0000-0003-2056-1932 <MKudrov@mail.ru>*
[2] *А.О. Морозов, ORCID: 0000-0003-4620-9662 <Morozov.AO@phystech.edu>*
[1,2] *А.С. Щеглов, ORCID: 0000-0002-0669-3392 <Shcheglov@phystech.edu>*

[1] *Центральный аэрогидродинамический институт им. проф. Н.Е. Жуковского, 140180, Россия, г. Жуковский, ул. Жуковского, д. 1*
[2] *Московский физико-технический институт, 141701, Россия, Долгопрудный, Институтский пер., д. 9*

Аннотация. Исследование режимов обледенения ЛА, при которых необходимо учитывать эффект дробления капель, представляет большой интерес при расчетах обледенения летательных аппаратов, оптимизации гидрофобных и противообледенительных свойств покрытий и актуально в ряде других практических приложении. Большой практической значимостью является разработка высокопроизводительных методов расчета взаимодействия аэрозольных течений с твердым телом. Данная работа посвящена развитию модели динамики частиц, а также модели дробления переохлажденных капель аэрозольного потока при его взаимодействии с поверхностью обтекаемого тела. Развитые физико-математические модели могут быть использованы в программных комплексах численного моделирования обледенения летательных аппаратов.

Ключевые слова: переохлажденные капли; несферические кристаллы; хаотизация; дробление; льдофобные покрытия.

Для цитирования: Амелюшкин И.А., Кудров М.А., Морозов А.О., Щеглов А.С. Математические модели и методы расчета процессов, сопровождающих обледенение летательных аппаратов. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 237-248. DOI: 10.15514/ISPRAS–2021–33(5)–14.

Благодарности: Работа выполнена при поддержке РФФИ (проект № 19-29-13016).

Mathematical models and methods of numerical investigation of processes which accompany aircraft icing

[1,2] *I.A. Amelyushkin, ORCID: 0000-0002-4281-3531 <Amelyushkin_Ivan@mail.ru>*
[2] *M.A. Kudrov, ORCID: 0000-0003-2056-1932 <MKudrov@mail.ru>*
[2] *A.O. Morozov, ORCID: 0000-0003-4620-9662 <MorozovAO@ phystech.edu>*
[1,2] *A.S. Sheglov, ORCID: 0000-0002-0669-3392 <Shcheglov@phystech.edu >*

[1] *Central Aerohydrodynamic Institute,*
1, Zhukovskogo st., Zhukovsky, 140180, Russia.
[2] *Moscow Institute of Physics and Technology,*
9, Insitutsky per., Dolgoprudny, 141701, Russia

Abstract. The study of aircraft icing modes, in which it is necessary to take into account the effect of droplet crushing, is of great interest in calculating the icing of aircraft, optimizing the hydrophobic and anti-icing properties of coatings, and is relevant in a number of other practical applications. Of great practical importance is the development of high-performance methods for calculating the interaction of aerosol flows with a solid. This work is devoted to the development of a model of particle dynamics, as well as a model of fragmentation of supercooled droplets of an aerosol flow during its interaction with the surface of a streamlined body. Developed physical and mathematical models can be used in software systems for numerical modeling of aircraft icing.

Keywords: supercooled drops; non-spherical crystals; chaos; splash; ice-phobic coatings

For citation: Amelyushkin I.A., Kudrov M.A., Morozov A.O., Shcheglov A.S. Mathematical models and methods of numerical investigation of processes which accompany aircraft icing Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 237-248 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–14

Acknowledgements. The work is supported by Russian Foundation of Fundamental Research (project No 19-29-13016).

1. Введение

Исследование взаимодействия многофазных потоков с обтекаемыми телами – одна из актуальных задач аэрогидромеханики. Это обусловлено широким кругом явлений, возникающих в технических приложениях, в частности в задачах противодействия обледенению, управления тепломассообменом и при визуализации процессов в аэрогидродинамических установках. Присутствие дисперсной примеси даже в малых количествах зачастую приводит к существенным изменениям характеристик физического процесса, в частности к существенному снижению аэродинамического качества и повышению веса при обледенении. Исследование режимов обледенения ЛА, при которых необходимо учитывать эффект дробления капель представляет большой интерес при расчетах обледенения летательных аппаратов, оптимизации гидрофобных и противообледенительных свойств покрытий и актуально в ряде других практических приложении.

К основным физическим проблемам и задачам фундаментальных исследования обледенения следует отнести: 1) моделирование и управление взаимодействием воды и льда с поверхностью; 2) описание процессов, сопровождающих кристаллизацию переохлажденных метастабильных капель: преодоление потенциальных барьеров, поток энергии на межфазной границе и образование кристаллов при прохождении фронта кристаллизации; 3) описание движения кристаллов сложной формы и их взаимодействия с твердым телом; 4) реологические особенности пленки, содержащей кристаллы; 5) механизмы абляции и эрозии льда в потоке.

Создание высокопроизводительных методов расчета взаимодействия аэрозольных течений с твердым телом представляет большой практический интерес в задачах противодействия обледенению летательных аппаратов [1, 2]. Стохастические процессы, возникающие при движении частиц, и дробление капель о поверхность летательного аппарата могут существенно повлиять на процесс обледенения.

Данная работа посвящена развитию модели динамики частиц [3, 4], а также модели дробления переохлажденных капель аэрозольного потока при его взаимодействии с поверхностью обтекаемого тела. Развитые физико-математические модели могут быть использованы в программных комплексах численного моделирования обледенения летательных аппаратов.

2. Модель динамики частиц с учетом влияния стохастических процессов

При Лагранжевом рассмотрении движении частиц в задачах обледенения можно использовать модель сферической частицы. Сила, действующая на частицу может быть описана формулой:

$$\boldsymbol{F}_p = -\frac{1}{2} C_D \pi a_p^2 \rho_a \left|\mathbf{V}_{p,a}\right| \mathbf{V}_{p,a} + m_p \mathbf{g} - m_p \mathbf{g} \frac{\rho_a}{\rho_p} - m_a \frac{\mathrm{d}\mathbf{V}_{p,a}}{\mathrm{d}t} - 6 a_p^2 \sqrt{\pi \rho_a \mu_a} \int_0^t \frac{1}{\sqrt{t-\tau}} \frac{\mathrm{d}\mathbf{V}_{p,a}(\tau)}{\mathrm{d}\tau}, \quad (1)$$

где $\mathbf{V}_p$ – скорость частицы, $\mathbf{V}_a$ – скорость среды в текущей точке нахождения частицы, $\mathbf{V}_{p,a}$ – скорость частицы относительно среды (индексы a и p относятся к воздуху и частице, соответственно); C_D – коэффициент сопротивления частицы, зависящий от режима ее обтекания – главным образом от чисел Рейнольдса и Маха. $m_p = \rho_p \frac{4}{3} \pi a_p^3$ – масса частицы; a_p – объемноэквивалентный радиус частицы; t – текущее время; ρ_p – плотность частицы; ρ_a – плотность среды; μ_a μ_a – вязкость газа.

Первое слагаемое в (1) соответствует силе аэродинамического сопротивления, второе – силам тяжести, третье – силе Архимеда (g – вектор ускорения свободного падения), четвертое – силе, обусловленной наличием «присоединенной» массы (здесь $m_a = m_p \frac{\rho_a}{\rho_p}$), пятое – «наследственной» силе Бассе. Как показывают оценки, в случае газового аэрозольного потока ввиду различия на три порядка плотностей газа и твердых частиц (капель) значимы только первое и второе слагаемое.

После принятых допущений сила, действующая на сферическую частицу, примет следующий вид:

$$\boldsymbol{F}_p = -\frac{1}{2} C_D \pi a_p^2 \rho_a \left|\mathbf{V}_{p,a}\right| \mathbf{V}_{p,a} + m_p \mathbf{g}.$$

Большой интерес также представляет случай кристаллического обледенения. Кристаллы, в отличие от капель, характеризуются большим разнообразием форм. В качестве характерной формы можно рассмотреть сфероид (эллипсоид, две полуоси которого равны). В отличие от частиц сферической формы, для сфероидов следует учитывать подъемную силу, а также наличие вращательного момента силы. В таком случае уравнения динамики частицы примут вид:

$$\begin{cases} m_p \dfrac{\mathrm{d}\mathbf{V}_p}{\mathrm{d}t} = -\dfrac{1}{2} C_D\left(a, \mathrm{Re}_p, E\right)\pi a_p^2 \rho_a \left|\mathbf{V}_{p,a}\right|^2 \mathbf{e}_D + \dfrac{1}{2} C_L\left(a, \mathrm{Re}_p, E\right)\pi a_p^2 \rho_a \left|\mathbf{V}_{p,a}\right|^2 \mathbf{e}_L + m_p \mathbf{g} \\ I_{pc} \dfrac{\mathrm{d}\omega_{pc}}{\mathrm{d}t} = \dfrac{1}{2}\rho_a \left|\mathbf{V}_{p,a}\right|^2 \dfrac{\pi\left(2a_p\right)^2}{8} C_T\left(a, \mathrm{Re}_p, E\right)\mathbf{e}_T \\ \dfrac{\mathrm{d}\mathbf{r}_p}{\mathrm{d}t} = \mathbf{V}_p \\ \dfrac{\mathrm{d}\vartheta_p}{\mathrm{d}t} = \omega_p \end{cases}$$

Здесь $\mathbf{r}_p$ – радиус-вектор положения частицы в неподвижной системе координат; ϑ_p – угол между осью симметрии частицы и осью x неподвижной системы координат (рис. 1); I_{pc} – момент инерции частицы относительно оси z, проходящей через центр масс частицы; ω_{pc} – проекция угловой скорости вращения частицы на ось z; $E = \frac{a}{b}$ – удлинение частицы; a – длина главной полуоси частицы; b – длины боковых полуосей частицы; $\mathbf{e}_D, \mathbf{e}_T, \mathbf{e}_L$ – единичные направляющие вектора; C_D, C_T, C_L – коэффициенты силы лобового сопротивления, подъемной силы и вращательного момента соответственно, зависящие угла атаки частицы a, от числа Рейнольдса обтекания частицы $\mathrm{Re}_p = \frac{\rho_a d_p V_{p,a}}{\mu_a}$ и ее удлинения E; μ_a – динамическая вязкость среды.

Для моделирования C_D, C_T, C_L можно использовать зависимости, полученные на основе численного моделирования в работах авторов Фролиха (Frohlich), Сандживи (Sanjeevy), Заставни (Zastawny), Шиллера и Науманна (Schiller&Naumann) [5–8] (рис. 1).

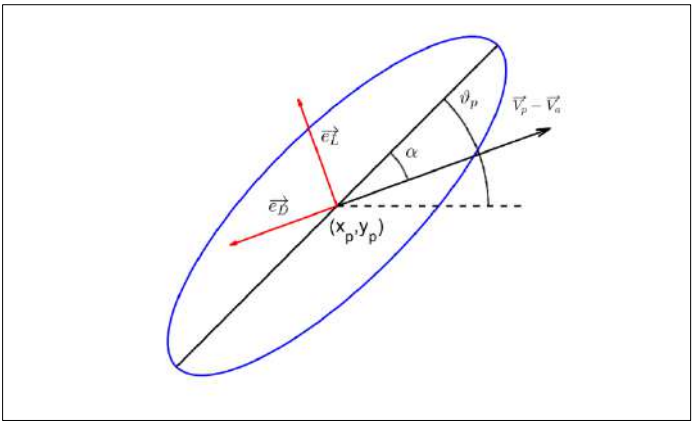


Рис. 1. Основные величины для сфероидной частицы
Fig. 1. Scheme of the main forces acting on a spheroid particle near a streamlined body

В процессе движения частица может испытывать стохастическое влияние следующих видов (рис. 2):

- турбулентные пульсации потока;
- броуновское движение малых частиц и молекул;
- стохастические эффекты при столкновении частиц друг с другом и с обтекаемым телом;
- эффекты неоднородности и нестационарности электромагнитного поля.

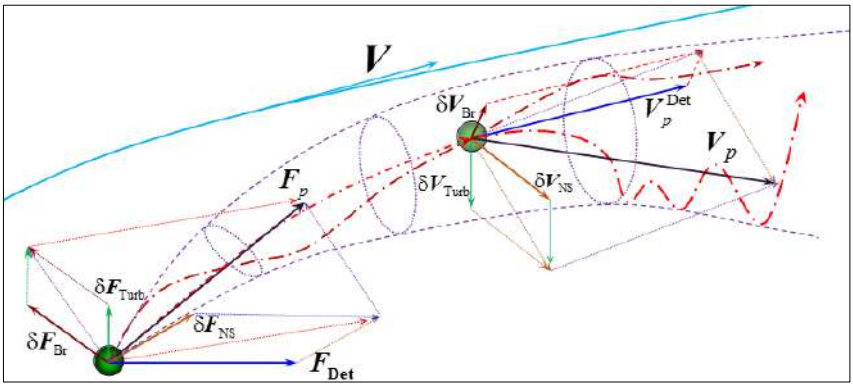


*Рис. 2. Схема движения частицы аэрозольного потока. Здесь **F** – сила, **V** – скорость, δ – малая величина, индексы p – частица, Turb – турбулентный, Br – броуновское, Det – детерминистический*
Fig. 2. Aerosol flow particle motion diagram. Here F is the force, V is the velocity, δ is a small value, the subscripts p is a particle, Turb is turbulent, Br is Brownian, Det is deterministic.

Оценим вклады различных типов стохастического влияния. Отношение характерных величин скоростей турбулентных и тепловых пульсаций можно оценить следующим образом:

$$\sqrt{\frac{\langle V_{\text{турб}}^2\rangle}{\langle V_{\text{тепл}}^2\rangle}} \cong \sqrt{\frac{I^2(u_{a\infty})^2 8\rho_p \pi a_p^3}{9kT}}.$$

Для ледяных частиц с плотностью $\rho_p = 920\,\frac{\text{кг}}{\text{м}^3}$, диаметром $d_p = 260$ мкм, при скорости потока $\langle u_{a\infty}\rangle = 9\frac{\text{м}}{\text{с}}$, интенсивности турбулентных пульсаций $I = 0.05$ и температуре $T = 268K$ имеем $\sqrt{\frac{\langle V_{\text{турб}}^2\rangle}{\langle V_{\text{тепл}}^2\rangle}} \cong 10^6$.

В практических задачах массовое содержание частиц имеет порядок десятка г/м^3. Объемная доля частиц в таком случае составляет порядка 10^{-5}, что позволяет пренебречь взаимодействием частиц [9].

Таким образом, из всех стохастических механизмов наиболее существенным оказывается наличие турбулентных пульсаций потока. Классический подход исследования турбулентности заключается в представлении параметров потока в каждой точке пространства в виде суммы среднего значения и мгновенной пульсационной составляющей:

$$\mathbf{V}_a(\mathbf{r},t) = \langle \mathbf{V}_a(\mathbf{r},t)\rangle + \mathbf{V'}_a(\mathbf{r},t).$$

При использовании моделей $k-\varepsilon$ и $k-\omega$ с помощью RANS-уравнений вычисляется среднее поле скорости $\langle \mathbf{V}_a(\mathbf{r},t)\rangle$ $\langle \mathbf{V}_a(\mathbf{r},t)\rangle$ и моделируются характеризующие турбулентность величины k, ε, ω. $k = \frac{1}{3}\langle \mathbf{V'}_{ai}\mathbf{V'}_{ai}\rangle$ представляет собой удельную кинетическую энергию турбулентных пульсаций и характеризует средний квадрат пульсаций скорости в данной точке пространства. Часть информации о турбулентной скорости газа при усреднении теряется. Необходима модель для генерации пульсационной скорости.

Предполагая локальную изотропность пульсаций скорости, рассмотрим одну компоненту пульсационной скорости движения частицы среды. Воспользуемся описанием движения с помощью уравнения Ланжевена:

$$V'(t+\mathrm{d}t) = V'(t) + V'(t)\frac{\mathrm{d}t}{T_{corr}} + \sqrt{\frac{2\sigma^2}{T_{corr}}\mathrm{d}W(t)}, \qquad (2)$$

где $\sigma = \sqrt{\langle V'^2 \rangle} = \sqrt{\frac{2k}{3}}$ – среднеквадратическое значение компоненты пульсационной скорости в данной точке пространства; $\mathrm{d}W(t)$ – Гауссов белый шум с дисперсией $\mathrm{d}t$ и нулевым средним, T_{corr} – локальный временной масштаб турбулентности, который из соображения размерностей моделируется в виде: $T_{corr} = C\frac{k}{\varepsilon}$, где C – эмпирическая константа.

Уравнение (2) содержит как демпфирующее слагаемое, вызванное трением о соседние слои жидкости, так и слагаемое, моделирующее стохастические импульсы. Уравнение (2) преобразуется к дискретной форме для конечных величин Δt:

$$V'(t+\Delta t) = V'(t)R_L(\Delta t) + \sigma\xi\sqrt{1 - R_L^2(\Delta t)}, \quad (3)$$

где $R_L(\Delta t) = \frac{\langle V'(t+\Delta t)V'(t)\rangle}{\langle V'^2(t)\rangle} = \exp(-\Delta t/T_{corr})$ – Лагранжева автокорреляционная функция пульсационной компоненты скорости; $\xi \sim N(0.1)$.

3. Результаты моделирования движения несферических частиц вблизи тела

Моделируются ледяные частицы ($\rho_p = 917\,\frac{\text{кг}}{\text{м}^3}$) вблизи цилиндра диаметром $D = 315$ мм при скорости потока воздуха $V_{a\infty} = 9\,\frac{\text{м}}{\text{с}}$ и температуре $T = 268K$. Интенсивность турбулентных пульсаций на левой границе задана равной $I = 5\%$. Данные условия были выбраны в соответствии с экспериментальными данными [10]. Расчет аэродинамических полей вблизи цилиндра осуществляется с использованием модели турбулентности SST. Вязкость воздуха вычисляется по формуле Сазерленда.

Интерес представляет интегральная характеристика взаимодействия частиц с поверхностью:

$$\eta = \frac{\dot{m}_{imp}}{\dot{m}_{inc}},$$

где $\dot{m}_{imp}$ – масса частиц, сталкивающихся с поверхностью тела, в единицу времени; $\dot{m}_{inc}$ – масса частиц, проходящих в единицу времени через поверхность, ширина которой равна ширине тела (красная поверхность на рис. 3). На рис. 3 расчетная область обозначена пунктирной линией, а частицы – зелеными точками. Также на рис. 3 показано каким образом определяется угол попадания частицы θ.

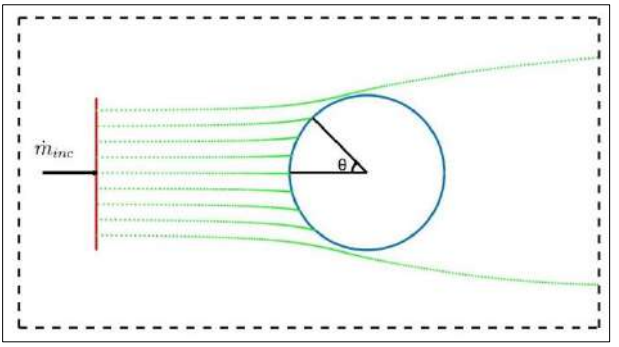


Рис. 3.Схема расчетной области
Fig. 3. Scheme of the computational domain

Результаты моделирования с различными моделями расчета аэродинамических коэффициентов приведены на рис. 4. Видно, что η зависит от отношения больших полуосей сфероидов E для цилиндра $D = 315$ мм. Результаты моделирования показывают, что как наличие удлинения у частиц ($E > 1$), так и наличие сплюснутости ($E > 1$) приводит к дополнительному рассеянию, вызванному несферичностью частицы. Максимум η

наблюдается для сферических частиц. Максимум величины η наблюдается для сферических частиц ($E > 1$) и совпадает с [10] при использовании моделей Шиллера-Науманна и Фролиха.

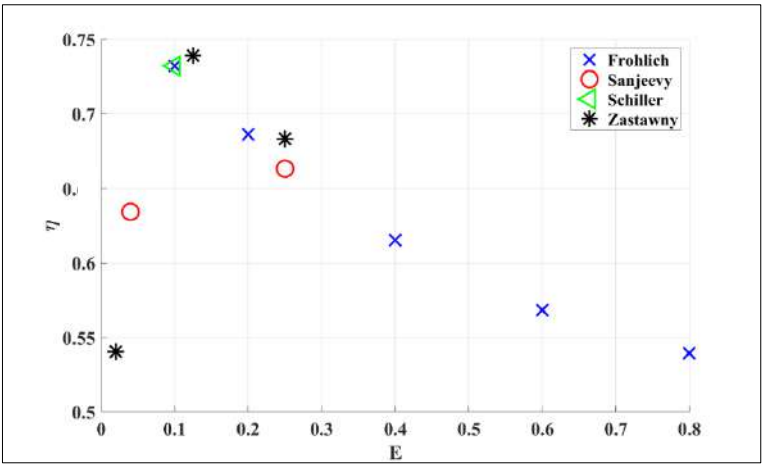


Рис. 4. Результаты моделирования величины η в зависимости от удлинения сфероидов E для цилиндра D=315 мм

Fig. 4. The results of modeling the value of η depending on the elongation of the spheroids E for the cylinder D = 315 mm

На рис. 5 представлены траектории сферических частиц с диаметром $D_p = 3 -$ мкм с учетом влияния на них турбулентных пульсаций и без учета последних. Для наглядности на этом же рисунке приведено поле скорости $\langle \mathbf{V}_a \rangle$.

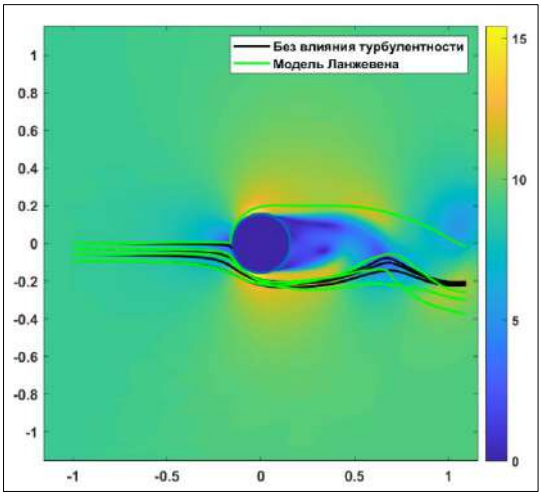


Рис. 5. Траектории частиц вблизи цилиндра

Fig. 5. Particle trajectories near the cylinder

На рис. 6 приведена зависимость предельного угла попадания θ_{max} частиц на поверхность цилиндра от числа Стокса $\mathrm{Stk}_\infty = \frac{4}{9}\frac{\rho_p V_{a\infty} a_p^2}{\mu_a D}$. Число Стокса варьировалось путем изменения диаметра частиц d_p. Учет стохастического влияния турбулентных пульсаций приводит к увеличению θ_{max}.

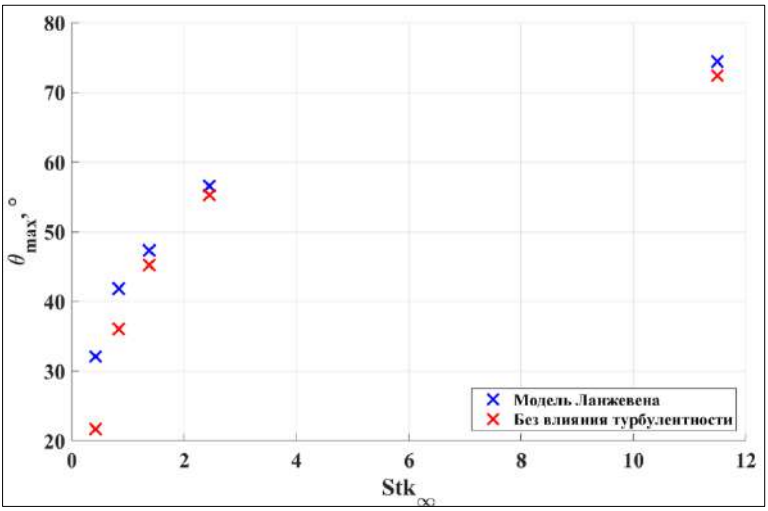


Рис. 6. Зависимость максимального полярного угла попадания частиц на поверхность θ_{max} от числа Стокса Stk_∞ с учетом влияния турбулентности (синие точки) и без учета (красные точки)
Fig.6. Dependence of the maximum polar angle of particle hitting the surface θ_{max} on the Stokes number Stk_∞ taking into account the effect of turbulence (blue points) and not taking into account (red points)

4. Модели дробления капель при соударении с поверхностью

При значительном отличии счетной концентрации ввиду отличия количества капель, которые моделируют брызги, распределения массовых концентраций капель практически не отличаются при $N \geq 5$. Таким образом, оптимальное значение N около 5. При расчете использовались следующие параметры: $a_p = 50$ мкм; начальное число частиц $N_p = 200000$; радиус цилиндра $R = 0.1$ м; $V_{a\infty} = 100\,\frac{\text{м}}{\text{с}}$; давление атмосферное; температура $T = 263K$. Распределение толщины пограничного слоя δ вблизи обтекаемого аэрозольным потоком поперечного цилиндра можно оценить на основании следующего хорошо согласующегося с более точным решением уравнений газовой динамики выражением $\delta(s) = \delta_0\left(1 + \left(\frac{s}{\pi R/2}\right)^2\right)$, здесь s – координата на поверхности цилиндра радиуса R.

В настоящей работе с учетом зависимости [11] максимальной скорости удара капель $\frac{V_{in}^{max}}{V_{p\infty}} \cong \exp(-1/4\text{Stk}_\infty)$ получен критерий режима дробления капель аэрозольного потока который обтекает крыло летательного аппарата с характерным радиусом кривизны профиля R:

$$K = \frac{\text{We}_n}{\sqrt[5]{\text{Oh}^2}} = \frac{2a_p\rho_p V_{pn}^2}{\sigma\sqrt[5]{\text{Oh}^2}} = \frac{2a_p\rho_p V_{\infty}^2}{\sigma\sqrt[5]{\text{Oh}^2}}\,{}^{-2\text{Stk}}\sqrt{e},$$

где $V_{p\infty}$ – скорость капли вдали от обтекаемого тела с характерным радиусом кривизны передней кромки R.

Если значение $K > 1$ при обтекании тела, капли аэрозоля будут дробиться при ударе о поверхность. Здесь учтена зависимость числа Вебера $\text{We}_n = \frac{\rho_p d_p V_{pn}^2}{\sigma_p}$ удара капли по нормали к поверхности тела от числа Стокса Stk_∞, рассчитанное по нормальной компоненте скорости удара, $\text{Oh} = \frac{\mu_a}{\sqrt{\rho_a\sigma_p d_p}}$ – число Онезорге, σ_p – коэффициент поверхностного натяжения жидкости.

Зная скорость подлета $\mathbf{V}_0$ и отскока $\mathbf{V}_{\text{Rebound}}$ капли, конечную и начальную массы капли m_0 и m_{Rebound}, а также массу брызг m_{Splash}, скорость последних можно найти согласно следующему выражению:

$$\mathbf{V}_{\text{Splash}} \cong \frac{m_0\mathbf{V}_0 - m_{\text{Rebound}}\mathbf{V}_{\text{Rebound}}}{m_{\text{Splash}}}.$$

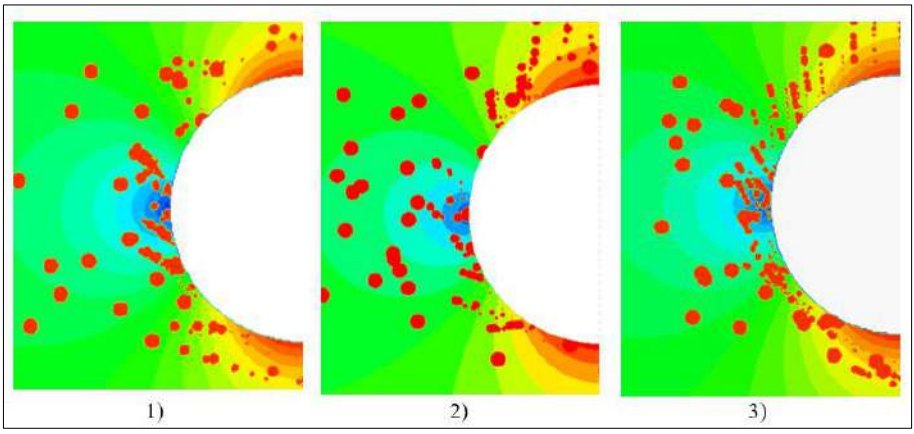


Рис. 7. Визуализация поведения капель при их дроблении для различных значений N: 1) N = 2; 2) N = 5; 3) N = 10

Fig. 7. Visualization of the behavior of drops during their crushing for different values of N: 1) N = 2; 2) N = 5; 3) N = 10

Поскольку капля дробится на колоссальное количество брызг, в настоящей работе брызги моделировались набором одинаковых более мелких частиц, суммарный импульс которых рассчитывался в соответствии с изменением массы и скорости удара капли о поверхность. Направления разлета брызг – равновероятные в полусфере, которая перпендикулярна поверхности обтекаемого тела. В расчетах предполагалось, что если размер капли уменьшается на 99 %, то считается, что капля полностью распалась при ударе на брызги. В расчетах предполагалось, что брызги моделируются заданным количеством N отдельных капель массой $m'_{\text{Splash}} = m_{\text{Splash}}/N$, разлетающихся под произвольными углами в полупространстве, ограниченном касательной плоскостью к точке удара о поверхность обтекаемого тела.

На рис. 7 показана визуализация изменения размеров капель при их взаимодействии с цилиндром в воздушном потоке. Поскольку размеры капель на 5–6 порядков меньше радиуса цилиндра, на рисунке их размеры увеличены в 10^4 раз. Из рисунка видно, что качественно распределения частиц в пространстве отличаются незначительно. Наибольшие отличия вблизи поверхности тела, т.к. мелкие брызги в силу малости их инерции и времени релаксации быстро увлекаются потоком и сносятся вниз по течению. Влияние количества капель на распределение массовой концентрации незначительное.

На рис. 8 показано интегральное значение разницы концентраций вблизи обтекаемого тела. Под счетной концентрацией частиц понимают их количество в единице объема, под массовой – массу частиц в единице объема. Область интегрирования: $x \in [-2R, 0], y \in [-2R, 2R]$, R – радиус цилиндра. Значения объемных концентраций нетрудно получить, домножив значения массовой концентрации на плотность материала частиц.

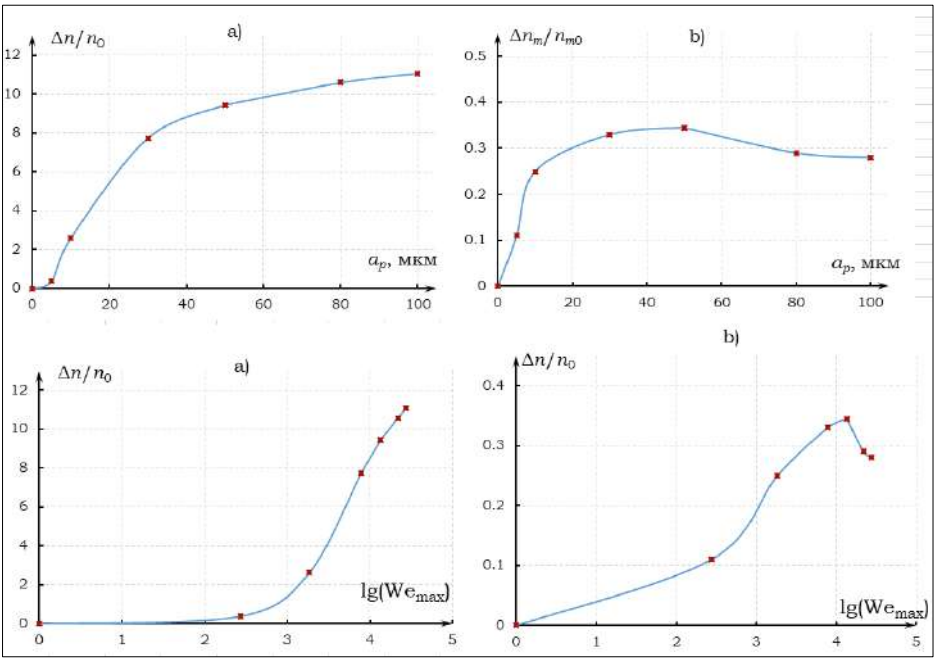


Рис. 8. Интегральная разность концентраций капель: а) счетной, б) массовой. Внизу показаны зависимости разности концентраций от максимального значения числа Вебера: входящая в выражение для числа максимальная скорость удара капель о поверхность зависит от числа Стокса
Fig. 8. Integral difference of droplet concentrations: a) countable, b) mass. Below are shown the dependences of the concentration difference on the Maximum Weber number: the maximum velocity of droplet impact on the surface included in the expression for the number depends on the Stokes number

5. Заключение

Исследовано и иллюстративно показано влияние формы частиц на коэффициент их захвата поверхностью в приложении к задачам расчета обледенения летательных аппаратов. Приведены оценки характеристик основных механизмов стохастического движения тел в неоднородных средах. Показано, что учет стохастического влияния турбулентных пульсаций приводит к увеличению предельного угла попадания частицы на поверхность.

Сформулирован критерий режима дробления соударяющихся с поверхностью капель, зависящий от параметров аэрозольного потока для оценки наличия дробления при обтекании тела при заданных параметрах. Показано влияние дробления капель на интегральную разницу в распределении массовой и счетной концентраций капель. Представленные в работе результаты аналитических и численных исследований могут быть использованы в расчетах обледенения и послужить основой в задачах противодействия этому явлению.

Список литературы / References

[1] Амелюшкин И.А., Кудров М.А. и др. Модели процессов, сопровождающих кристаллизацию переохлажденных капель. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 235-244 / Amelyushkin I.A., Kudrov M.A. et al. Models of processes accompanying crystallization of supercooled metastable droplets. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 235-244 (in Russian). DOI: 10.15514/ISPRAS-2020-32(4)-17.

[2] Кашеваров А.В., Стасенко А.Л. Взаимодействие частиц различной формы с несущим континуальным потоком (обзор). Ученые записки ЦАГИ, том XLV, вып. 5, 2014 г., стр. 3-17 / Kashevarov A.V., Stasenko A.L. Interaction of particles of various shapes with a carrying continuous flow (review). TsAGI Scientific Notes, vol. XLV, issue 5, 2014, pp. 3-17 (in Russian).

[3] Амелюшкин И.А., Стасенко А.Л. Стохастика и детерминизм движения капель воды в переохлажденном аэрозольном потоке, обтекающем твердое тело. Вестник МГОУ, серия 'Физика-математика', no. 4, 2014 г., стр. 61-69 / Amelyushkin I.A., Stasenko A.L. Stochastics and determinism of water droplet motions in a supercooled aerosol flow which circumfluents a solid body. Bulletin MSRU, series Physics and Mathematics, no. 4, 2014, pp. 61-69 (in Russian).

[4] Кашеваров А.В., Стасенко А.Л. Дискретно-капельный режим образования льда. Журнал технической физики, том 90, вып. 1, 2020 г., стр. 46-52 / Kashevarov A.V., Stasenko A.L. Discrete-Drop Mode of Ice Accretion on a Cylinder in Transverse Supercooled Flow. Technical Physics, vol. 65, issue 1, 2020, pp. 41-47.

[5] Fröhlich K., Meinke M., and Schröder W. Correlations for inclined prolates based on highly resolved simulations. Journal of Fluid Mechanics, vol. 901, 2020, article A5.

[6] Zastawny M., Mallouppas G. et al. Derivation of drag and lift force and torque coefficients for non-spherical particles in flows. International Journal of Multiphase Flow, vol. 39, 2012, pp. 227-239.

[7] Schiller L. and Naumann A.: Fundamental calculations in gravitational processing. Zeitschrift Des Vereines Deutscher Ingenieure, vol. 77, 1933, pp. 318–320.

[8] Sanjeevi S.K.P., Kuipers J.A.M., & Padding J.T. Drag, lift and torque correlations for nonspherical particles from Stokes limit to high Reynolds numbers. International Journal of Multiphase Flow, vol. 106, 2018, pp. 325-337.

[9] Pozorski J. Models of Turbulent Flows and Particle Dynamics. In Particles in Wall-Bounded Turbulent Flows: Deposition, Re-Suspension and Agglomeration. CISM International Centre for Mechanical Sciences (Courses and Lectures), vol 571, Springer, 2017, pp. 97-150.

[10] Vigano A. Modélisation numérique et expérimentale des phénomènes de givrage par accrétion de neige collante. Autre. ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique - Poitiers, 2012. 192 p.

[11] Амелюшкин И.А., Миллер А.Б., Стасенко А.Л. Оценка периода шероховатости противообледенительных покрытий тела в потоке воздуха с переохлажденными каплями. Вестник МГОУ, серия 'Физика-математика', no. 1, 2021 г., стр. 54-63 / Amelyushkin I.A., Miller A.B., Stasenko A.L. Estimation of the roughness period of anti-ice body coatings in air flow with supercooled droplets. Bulletin MSRU, series Physics and Mathematics, no. 1, 2021, pp. 54-63 (in Russian).

Информация об авторах / Information about authors

Иван Алексеевич АМЕЛЮШКИН – кандидат физико-математических наук, старший научный сотрудник ЦАГИ и МФТИ, Научные интересы: динамика многофазных сред, высокопроизводительные вычисления, обратные задачи и их приложения, молекулярное моделирование и многомасштабный анализ.

Ivan Alekseevich AMELYUSHKIN – Candidate of Physical and Mathematical Sciences, Senior Researcher at TsAGI and MIPT, Research interests: dynamics of multiphase environments, high performance computing, inverse problems and their applications, molecular modeling and multiscale analysis.

Максим Александрович КУДРОВ – кандидат технических наук, доцент, ведущий научный сотрудник, заведующий лабораторией информационных технологий и прикладной математики МФТИ. Научные интересы: разработка комплекса программ для компьютерного моделирования физических процессов.

Maksim Aleksandrovich KUDROV – Candidate of Technical Sciences, Associate Professor, Leading Researcher, Head of the Laboratory of Information Technologies and Applied Mathematics MIPT. Research interests: development of software for computer modeling of physical processes.

Алексей Олегович МОРОЗОВ – инженер МФТИ. Научные интересы: компьютерное моделирование физических процессов, численные методы в задачах аэрогидродинамики.

Alexey Olegovich MOROZOV – engineer at MIPT. Research interests: computer modeling of physical processes, numerical methods in problems of aerohydrodynamics.

Андрей Сергеевич ЩЕГЛОВ – инженер ЦАГИ, аспирант МФТИ. Научные интересы: математическое моделирование физических процессов, численные методы, динамика полета.

Andrey Sergeevich SHCHEGLOV – engineer at TsAGI, post-graduate student at MIPT. Research interests: mathematical modeling of physical phenomena, numerical methods, flight dynamics.

DOI: 10.15514/ISPRAS-2021-33(5)-15

Особенности реализации эффективного алгоритма параллельных вычислений для моделирования обледенения стреловидного крыла с профилем GLC-305

[1,2] *К.Б. Кошелев, ORCID: 0000-0002-7124-3945<koshelevkb@mail.ru>*
[2] *А.В. Осипов, ORCID: 0000-0001-9223-4274 <a.osipov@ispras.ru>*
[2] *С.В. Стрижак, ORCID: 0000-0001-5525-5180 <s.strijhak@ispras.ru>*

[1] *Институт водных и экологических проблем СО РАН,*
656038, Алтайский край, г. Барнаул, ул. Молодежная, д.1
[2] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, г. Москва, ул. А. Солженицына, д. 25

Аннотация. В работе рассматривается возможность библиотеки ICELIB, разработанной в ИСП РАН, для моделирования процессов ледообразования на поверхности летательных аппаратов. В качестве тестового примера для сравнения точности моделирования физических процессов, возникающих при эксплуатации самолета, исследовалась поверхность стреловидного крыла с профилем GLC-305. В статье обсуждаются возможности эффективного алгоритма распараллеливания с использованием модели жидкой пленки, динамической сетки и геометрического метода биссектрис. Разработанная библиотека ICELIB это совокупность трех решателей. Первый решатель iceFoam1 предназначен для предварительной оценки зон обледенения поверхности фюзеляжа и крыла летательного аппарата. Изменением геометрической формы исследуемого тела пренебрегаем, толщина образования льда пренебрежимо мала. Данная версия решателя не имеет ограничений на количестве вычислительных ядер при распараллеливании. Вторая версия решателя iceDyMFoam2 предназначена для моделирования образования двух типов наледи гладкой (“Glaze ice”), так и рыхлой (“Rime ice”) для которой зачастую форма льда принимает сложный и причудливый вид. Учитывается влияние изменения формы тела на процесс обледенения. Ограничения связаны с особенностями построением сетки вблизи пограничного слоя обтекаемого тела. Для движения передней и задней границ пленки используются разные алгоритмы, которые оптимизированы для своих случаев. Прирост производительности ограничен и достигается при фиксированном числе ядер. Третья версия решателя iceDyMFoam3 также позволяет учитывать влияние изменения поверхности твердого тела при образовании наледи на сам процесс обледенения. Для случая образования наледи гладкого типа последняя версия решателя пока уступает по своим возможностям второй при сложных формах поверхности льда. В третьей версии пока используется несколько упрощенный и более единообразный подход для расчета движения обеих границ пленки наледи. Проведена оценка результатов расчета с данными эксперимента M. Papadakis для различных профилей и стреловидного крыла для случая “Rime ice”. Получено хорошее согласование с результатами эксперимента.

Ключевые слова: газокапельный; поток; крыло; расчет; решатель; сетка; обледенение; жидкая пленка; параллельные вычисления; координаты; узел; толщина льда; область; декомпозиция; осреднение; распределение по ядрам.

Для цитирования: Кошелев К.Б., Осипов А.В., Стрижак С.В. Особенности реализации эффективного алгоритма параллельных вычислений для моделирования обледенения стреловидного крыла с

профилем GLC-305. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 249-258. DOI: 10.15514/ISPRAS-2021-33(5)-15

Благодарости: Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-29-13016.

Features of the implementation of an efficient parallel computation algorithm for modeling the icing of a swept wing with a GLC-305 airfoil

[1,2] *K.B. Koshelev, ORCID: 0000-0002-7124-3945 <koshelevkb@mail.ru>*
[2] *A.V. Osipov, ORCID: 0000-0001-9223-4274 <a.osipov@ispras.ru>*
[2] *S.V. Strijhak, ORCID: 0000-0001-5525-5180 <s.strijhak@ispras.ru>*

[1] *Institute for Water and Environmental Problems SB RAS*
1, Molodyoznaya St., Barnaul, 656038, Altai Krai, Russia
[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia

Abstract. The paper considers the possibility of the ICELIB library, developed at ISP RAS, for modeling ice formation processes on the surface of aircraft. As a test example to compare the accuracy of modeling the physical processes arising during the operation of the aircraft, the surface of a swept wing with a GLC-305 profile was studied. The possibilities of an efficient parallelization algorithm using a liquid film model, a dynamic mesh, and the geometric method of bisectors are discussed. The developed ICELIB library is a collection of three solvers. The first solver iceFoam1 is intended for preliminary estimation of the icing zones of the fuselage surface and aircraft's swept wing. The change in the geometric shape of the investigated body is neglected, the thickness of ice formation is negligible. This version of the solver has no restrictions on the number of cores when parallelizing. The second version of solver iceDyMFoam2 is designed to simulate the formation of two types of ice, smooth ("Glaze ice") and loose ("Rime ice"), for which the shape of ice often takes on a complex and bizarre appearance. The effect of changing the shape of the body on the icing process is taken into account. The limitations are related to the peculiarities of the construction of the mesh near the boundary layer of the streamlined body. Different algorithms are used to move the front and back edges of the film, which are optimized for their cases. The performance gain is limited and is achieved with a fixed number of cores. The third version of solver iceDyMFoam3 also allows you to take into account the effect of changes in the surface of a solid during the formation of ice on the icing process itself. For the case of smooth ice formation, the latest version of the solver is still inferior in its capabilities to the second one with complex ice surface shapes. In the third version, a somewhat simplified and more uniform approach is still used to calculate the motion of both boundaries of the ice film. The estimation of the calculation results with the data of the experiment from M. Papadakis for various airfoils and swept wing for the case of "Rime ice" is carried out. Good agreement with the experimental results was obtained**.**

Keywords: gas-droplet flow; wing; simulation; solver; grid; ice accretion; liquid film; parallel simulation; coordinates; grid; thickness of ice; domain; decomposition; averaging; distribution on cores

For citation: Koshelev K.B., Osipov A.V. Strijhak S.V. Features of implementation an efficient parallel computation algorithm for modeling the ice accretion of a swept wing with a GLC-305 airfoil. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 249-258 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–15

Acknowledgements. The reported study was funded by RFBR, project number № 19-29-13016.

1. Введение

Вопросы изучения процессов обледенения являются актуальными в связи с использованием авиационной техники, различного оборудования, ветроэлектрических установок в северных широтах и в сложных климатических условиях на территории РФ. На поверхности элементов

самолета, вертолета, кабелей линий электропередач, лопастей ветроэлектрической установки (ВЭУ) могут образовываться различные формы льда, которые приводят к снижению аэродинамического качества и к снижению подъемной силы для обтекаемой поверхности. Для борьбы со льдом используют различные противобледенительные системы, но их не всегда устанавливают, например, на крыле самолета и лопасти ВЭУ. Поэтому важно иметь экспериментальные установки, технологии обработки данных летного эксперимента и расчетные эффективные коды для изучения процессов образования льда на поверхности при изменении физических параметров (скорость, температура, водность, диаметр капель, время в потоке капель) в широком диапазоне изменении исследуемых величин.

2. Математическая модель

В ИСП РАН был разработан тематический решатель iceFoam с использованием Эйлер-Лагранжевой модели для моделирования процесса обледенения [1]. В дальнейшем были разработаны три новые версии решателя iceFoam на основе Эйлерового подхода для газовой фазы и Лагранжевого подхода для моделирования капелек воды и модели пленки воды на поверхности профиля поверх слоя льда. Во всех трех версиях решателей используется модель пленки воды и слоя льда, которая связана с моделью по теории мелкой воды [1,2].

Первая версия решателя iceFoam1 предназначена для начальной оценки мест обледенения произвольного 3D профиля. Данный решатель полностью использует все возможности технологии параллельного программирования MPI для проведения эффективных вычислений средствами открытого пакета OpenFOAM v1912. Главным ограничением этого решателя является предположение, что толщина слоя льда достаточно мала и изменением поверхности профиля при обледенении можно пренебречь. Профиль со льдом и профиль без льда полагаются совпадающими. Для проведения декомпозиции пространственной области пленки используется разработанная утилита extrudeToFilmCellDist. Данная утилита позволяет обеспечить однообразие декомпозиции доменов в газовой области и в прилегающей пленке. Данная версия решателя не имеет ограничений на количестве ядер. С целью оценки масштабирования решателя выполнялись расчет на 12, 24, 48 вычислительных ядрах. С помощью данного решателя можно проводить быстрый расчет.

Вторая версия решателя iceDyMFoam2 предназначена для моделирования обледенения как рыхлым льдом (Rime ice), так и гладким льдом (Glaze ice), для которого зачастую форма льда принимает сложный и причудливый вид. В этой версии решатель достаточно эффективно позволяет учитывать влияние изменения поверхности профиля при обледенении на сам процесс обледенения. Данный решатель требует использования структурированной сетки вблизи поверхности профиля и наличия только одного пространственного домена для пленки и примыкающей к ней эйлеровой сетки. Опыт вычислений показал, что это требование ограничивает максимальное количество ядер, не более 12. При дальнейшем же увеличении количества используемых ядер эффективность параллельных растет слабо или даже падает при 24 ядрах. Для упрощения декомпозиции пространственной области газа разработана утилита addTwoLayersTo0. Данная утилита позволяет задать расчетный домен на нулевом процессоре для пленки и для двух прилегающих слоев эйлеровой сетки газовой фазы потока.

Третья версия решателя iceDyMFoam3 также позволяет учитывать влияние изменения поверхности профиля при обледенении на сам процесс обледенения. Как и первая версия, этот решатель полностью использует все возможности технологии параллельной библиотеки MPI для проведения эффективных параллельных вычислений. Параллельная реализация границы движения льда потребовала использовании низкоуровневых функций на языке программирования C++ открытого пакета OpenFOAM. В этом случае будут обязательно узлы, которые попадут на границу нескольких доменов.

В случае использования подхода с динамической сеткой потребовалось собирать информацию об узлах, которые хранятся в нескольких доменах или на разных расчетных вычислительных ядрах. Для таких узлов после локальных вычислений внутри одного домена выполнялась операция по определению нового положения узла. Далее делалось осреднение по всем доменам, участвующих в расчетах данного узла. Осредненные значения положение узлов, которые записаны были больше чем в одном домене, обратно распределялись по всем необходимых вычислительным ядрам. Таким образом, обеспечивалось совпадения движущихся узлов на разных доменах.

Для случая образования наледи гладкого типа третья версия решателя iceDyMFoam3 пока уступает по своим возможностям второй версии решателя при сложных формах поверхности льда, так как во второй версии для движения передней и задней границ пленки используются разные алгоритмы, которые оптимизированы для своих случаев. В третьей же версии пока используется несколько упрощенный и более единообразный подход для расчета движения обеих границ. Для проведения декомпозиции пространственной области пленки используется разработанная утилита extrudeToFilmCellDist.

Все три версии решателя iceFoam и дополнительные утилиты для работы с расчетной сеткой составляют основу тематической библиотеки ICELIB.

Для верификации решателей нами была проведена серия расчетов на вычислительных кластерах ИСП РАН, НИЦ “Курчатовский Институт” для различных профилей (NACA 0012, GLC-305, General Aviation, Business Jet, Commercial Transport) и 3D стреловидного крыла. Расчеты проводились для различных значений диаметров капель, температуры, скорости набегающего потока, водности, времени обледенения в набегающем потоке. Было проведено сравнение расчета с результатами эксперимента по толщине и форме льда [3,4]. Ошибка не превышала более 10%.

3. Особенности реализации параллельного алгоритма

Согласно концепции пакета OpenFOAM, узлы на границах расчетных доменов дублируются в каждом из этих доменов. В случае изменения координат этих узлов необходимо обеспечить их совпадение на разных вычислительных ядрах сервера. Имеется значение допустимого рассогласования координат таких узлов. Обычно это значение 0.0001 м.

Для расчета изменения границы льда геометрическим методом биссектрис необходима информация о положении всех граней, которым принадлежит узел. Однако, по умолчанию, такая информация доступна только в рамках текущего домена. Фактически на границе доменов вместо процедуры интерполяции используется процедура экстраполяция. При этом, как показано на рис. 1, перемещение совпадающих узлов из разных доменов происходит неоднообразно.

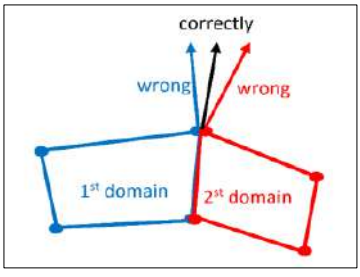


Рис. 1. Распределение по доменам и направление вектора роста льда
Fig. 1. Domain distribution and direction of the ice growth vector

Поэтому для устранения этого недостатка была разработана и запрограммирована следующая процедура:

1. Предварительный шаг. Выявляются номера всех узлов, которые принадлежат более чем одному домену. Этот шаг выполняется один раз перед расчетом, так как топология сетки не меняется.
2. На каждом шаге по времени при расчете движения границы льда:
   2.1. Стандартно выполняется процедура расчета движения льда методом биссектрис.
   2.2. Для каждого узла, принадлежащего более чем одному домену, выполняется осреднение координат по всем ядрам процессоров с использованием функции “reduce”.
   2.3. Осредненное значение координаты присваивается соответствующим узлам также для каждого ядра.

4. Постановка задачи

Для расчетного примера рассматривалась стреловидное крыло с профилем GLC-305 из работы [5]. На рисунке 2 показана геометрия стреловидного крыла с длиной L=1.52 метра и шириной корневой хорды c=0.64 метра, на рисунке 3 показан профиль крыла с координатами точек в безразмерном виде. Расчетная область представляла собой параллелепипед с размерами: длина 7 метров (от -2 до 5 метров, крыло начинается в точке с координатами 0, 0, 0), высота 3 метра (от -1.5 до 1.5 метров), ширина 3 метра (от 0 до 3 метра). Для крыла также заданы углы стреловидности и крутки.

На всех внешних границах, кроме входного сечения, задавалось граничное условие “zero gradient”, но поверхности крыла – условие “no slip”. В одном парселе получилось 350000 капелек, а всего парселей было больше миллиона. Сферические частицы вводились за 0.5 метра от передней корневой кромки крыла в сечении по вертикали от -0.19 до -0.01 метра, чтобы накрыть крыло полностью под углом в 6 градусов, по ширине 1.6 метра.

Ранее подобные численные расчеты проводились с использованием различных собственных кодов для аналогичного стреловидного крыла в работах [6-8].

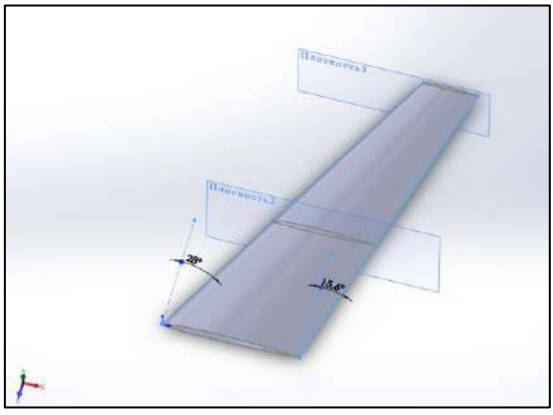


Рис. 2. 3D модель стреловидного крыла
Fig. 2. 3D swept wing model

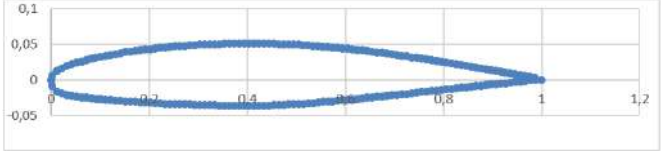


Рис.3 Модель профиля GLC-305
Fig.3.Profile model GLC-305

Расчетная неструктурированная адаптивная сетка была подготовлена с помощью утилиты snappyHexMesh в составе OpenFOAM v19.12 и содержала 1 355 049 узлов. Сетка включала в себя две области: первая для пристеночной подобласти пленки с высотой в одну расчетную ячейку, вторая для внешней области для моделирования газокапельного потока. Характерный размер ячейки около крыла был порядка 4 мм. На рис. 4 изображена сетка с 4 уровнями адаптации.

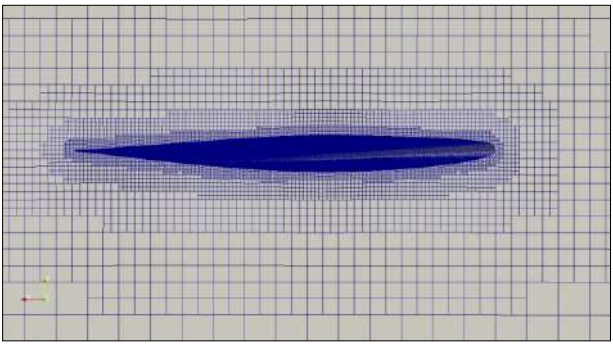
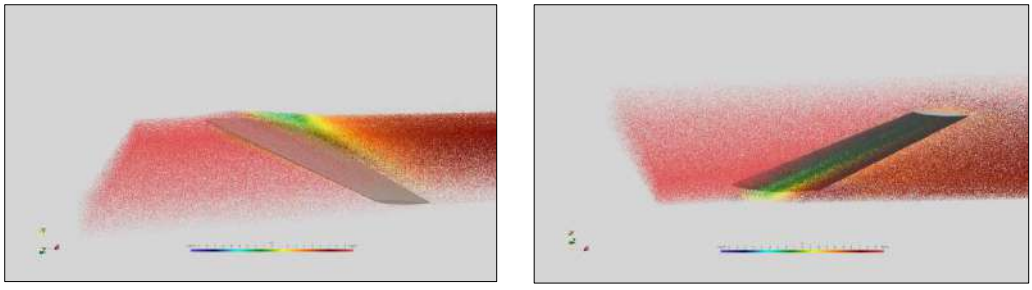


Рис. 4. Сетка с 4 уровнями адаптации
Fig. 4. Mesh with 4 levels of adaptation

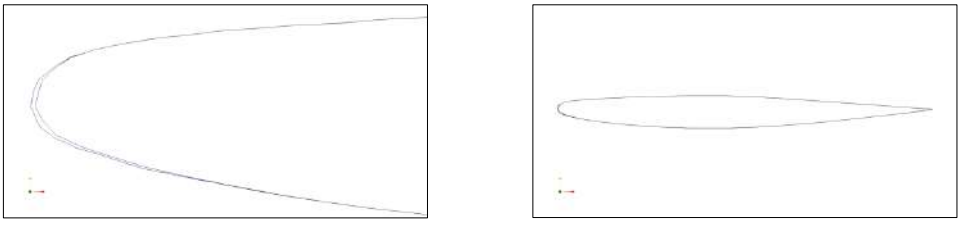
5. Результаты расчета

Расчеты проводились для случая “Rime ice” для значения скорости V=90 м/с, угла атаки в 6 градусов, температура воздуха Tair = 261.87 K, LWC=0.51 г/м^3, диаметр частиц d=14.5 микрон, время обледенения tau=5 минут. Расчетный шаг задавался равным $6 \cdot 10^{-5}$ секунд. Один вариант считался на вычислительном кластере НРС4 НИЦ “Курчатовский Институт” 33 часа на 24 вычислительных ядрах.



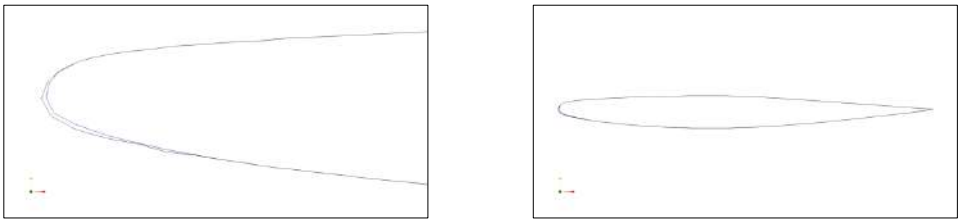
a) *b)*

Рис. 5. Поле скорости и траектории частиц вокруг стреловидного крыла
Fig. 5. Velocity field and particle trajectories around a swept wing



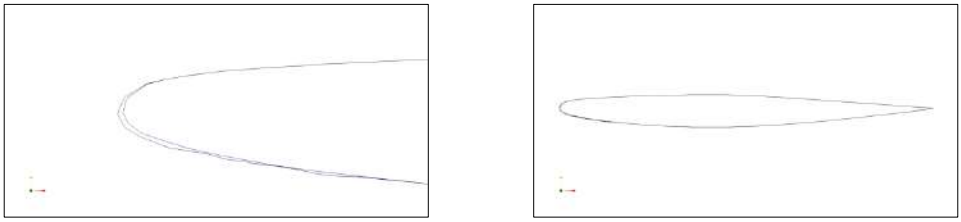
a) *b)*

Рис. 6. Форма льда на крыле в разных сечениях A
Fig. 6. The shape of the ice on the wing in different sections A



a) *b)*

Рис. 7. Форма льда на крыле в разных сечениях B
Fig. 7. The shape of the ice on the wing in different sections B



a) *b)*

Рис. 8. Форма льда на крыле в разных сечениях C
Fig. 8. The shape of the ice on the wing in different sections C

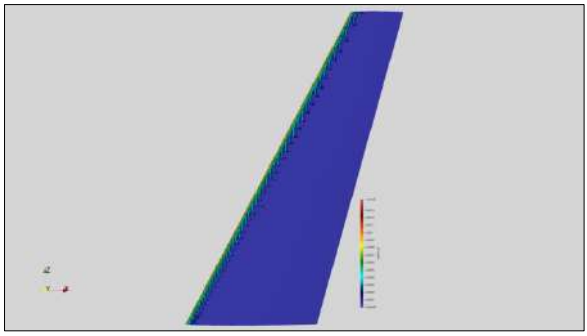


Рис. 9. Форма льда на нижней поверхности крыле
Fig. 9. Ice shape on the underside of the wing

На рис. 5 показана вертикальная скорость капель воды с видом сверху и снизу. Скорость относительно хорды крыла, которая сама наклонена под углом атаки 6 градусов.

На рис. 6a, 7a, 8a передняя кромка крыла до и после обледенения примерно в тех же местах, что и в [5]. На рис. 6b, 7b, 8b весь профиль до и после обледенения в тех же местах. Видно, что к краю крыла обледенение снизу увеличивается.

На рис. 9 представлена форма поверхности льда снизу на крыле. Видно 3D положение поверхности льда. Этот рисунок демонстрирует способность решателя моделирования существенно трехмерных процессов обледенения.

В работе [5] приведены осредненные только 2D графики поверхности льда. При этом можно отметить, что результаты данной работы близки по максимальной толщине льда (примерно 3 мм) к данным эксперимента, по форме льда результаты также качественно совпадают.

6. Заключение

В данной работе представлено описание трех версий решателя iceFoam с поддержкой параллельных вычислений. Решена задача с моделированием обледенения для стреловидного крыла с профилем GLC-305. Для расчетов использовалась версия решателя iceDyMFoam3. Проведено сравнение с результатами эксперимента работы [5]. Получено хорошее совпадение по толщине и форме льда.

Необходима дальнейшая работа по улучшению распараллеливания алгоритма решателя iceDyMFoam3 для расчетов на многоузловом вычислительном кластере с использованием технологии MPI [9,10].

Список литературы / References

[1]. Кошелев К.Б., Мельникова В.Г., Стрижак С.В. Разработка решателя iceFoam для моделирования процесса обледенения. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 217–234 / Koshelev K.B., Melnikova V.G. Strijhak S.V. Development of iceFom solver for modeling ice accretion. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 217–234 (in Russian). DOI: 10.15514/ISPRAS-2020-32(4)-16.

[2]. Bourgault Y., Beaugendre H., Habashi W. Development of a shallow-water icing model in FENSAP-ICE. Journal of Aircraft, vol. 37, no. 4, 2000, pp. 640-646.

[3]. Wright W., Rutkowski A. Validation results for LEWICE 2.0. Technical report CR–1999-208690, NASA, 1999, 674 p.

[4]. Addy, Harold E. Ice Accretions and Icing Effects for Modern Airfoils. Technical report TP-2000-210031, NASA, 2000

[5]. Papadakis M., Yeong H-W. et al. Experimental investigation of ice accretion effects on a swept wing. Technical report PB2005-110681, NASA, 2005, 205 p.

[6]. Zocca M., Gori G., and Guardone A. Blockage and Three-Dimensional Effects in Wind-Tunnel Testing of Ice Accretion over Wings. Journal of Aircraftm, vol. 54, no. 1, 2016, pp. 759-767.

[7]. Pena D., Hoarau Y., Laurendeau E. A single step ice accretion model using Level-Set method. Journal of Fluids and Structures, vol. 65, 2016, pp. 278-294.

[8]. Chang S., Tang H. et al. Three-Dimensional Modelling and Simulation of the Ice Accretion Process on Aircraft Wings. International Journal of Astronautics and Aeronautical Engineering, vol. 3, issue 2, 2018, pp. 1-25.

[9]. Гергель В.П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем. М., Изд-во МГУ, 2010 г., 544 стр. / Gergel V.P. High performance computing for multiprocessor multicore systems. M., Publishing house of Moscow State University, 2010, 544 p. (in Russian).

[10]. Гергель В.П., Сысоев А.В. Высокопроизводительные параллельные вычисления. 100 заданий для расширенного лабораторного практикума. М., ФИЗМАТЛИТ, 2018 г., 248 стр. / Gergel V.P., Sysoev A.V. High performance parallel computing. 100 tasks for an extended laboratory practice. M., FIZMATLIT, 2018, 248 p.

Информация об авторах / Information about authors

Константин Борисович КОШЕЛЕВ – кандидат физико-математических наук, доцент, старший научный сотрудник. Сфера научных интересов: вычислительная гидродинамика, гидрология, геоинформатика.

Konstantin Borisovich KOSHELEV is a candidate of physical and mathematical sciences, associate professor, senior researcher. Research interests: computational fluid dynamics, hydrology, geoinformatics.

Андрей Владимирович ОСИПОВ – инженер. Сфера научных интересов: вычислительная гидродинамика, метод контрольного объема, подвижные сетки, лагранжев подход.

Andrei Vladimirovich OSIPOV – an engineer. Research interests: computational fluid dynamics, finite volume method, dynamic meshes, particles.

Сергей Владимирович СТРИЖАК – кандидат технических наук, ведущий инженер. Сфера научных интересов: вычислительная гидродинамика, многофазные течения, турбулентность, ветроэнергетика, параллельные вычисления.

Sergei Vladimirovich STRIJHAK – a candidate of technical sciences, leading engineer. Research interests: computational fluid dynamics, multiphase flows, turbulence, wind power, parallel computing.

DOI: 10.15514/ISPRAS-2021-33(5)-16

Численное исследование микроклимата помещения с использованием СПО

[1] *А.А. Цынаева, ORCID: 0000-0003-4973-6169 <a.tsinaeva@gmail.com>*
[1,2] *Е.А. Цынаева, ORCID: 0000-0001-5380-5513 <bertyf@mail.ru>*
[1] *Самарский государственный технический университет,*
443010, Россия, г. Самара, ул. Молодогвардейская, д. 244
[2] *Ульяновский государственный университет,*
432063, Россия, Ульяновск, улица Льва Толстого, д. 42

Аннотация. Работа посвящена численному исследованию микроклимата жилого помещения, оборудованного системой управления микроклиматом, обозначаемой термином «умный дом». Численное моделирование микроклимата помещений является актуальной задачей, поскольку эффективное проектирование систем «умный дом» предполагает необходимость прогнозирования теплофизических параметров помещения с целью обеспечения возможности своевременного изменения этих параметров. Математическое моделирование было произведено на базе уравнения сохранения массы (уравнение неразрывности), уравнения сохранения количества движения (уравнение Навье-Стокса) и уравнения энергии. Замыкание системы уравнений осуществлено на основе использования *k-ω sst* модели турбулентности. Численное решение было получено методом конечных объемов с помощью программного кода Code_Saturne, относящегося к свободному программному обеспечению (СПО). Построение сетки выполнено в программном комплексе Salome, также имеющем свободную лицензию. В работе проведена проверка адекватности получаемого численного решения. Для разрешения поля скорости применена схема второго порядка (SOLU), для поля давления – схема (Multigrid), для кинетической энергии турбулентности, ее диссипации и для поля температур использованы автоматические настройки, максимальное количество итераций по каждому циклу задано равным 10000, точность решателя (Solver Precision) принята 10^{-8}. Для получения связанного решения уравнений баланса импульса и неразрывности использован алгоритм SIMPLEC. Верификация математической модели показала, что относительное отклонение температур от значений, полученных другими авторами, составило не более 0,8-1,2%. Таким образом, представленную математическую модель можно признать адекватной. На базе выполненного математического моделирования была получена оценка теплофизических параметров микроклимата в жилом помещении, которое оборудовано системой управления «умный дом». Математическое моделирование поля температур жилого помещения показало, что при использовании систем управления микроклимата значения температур в помещениях, имеющих различное назначение, не выходят за рамки требований комфортности. Математическое моделирование скоростей движения воздуха в различных жилых помещениях не нарушают требований комфортности для таких типов помещений и принимают значения от 0,12 до 0,15 м/с.

Ключевые слова: свободное программное обеспечение; математическое моделирование; микроклимат помещения; умный дом; верификация модели; энергосбережение

Для цитирования: Цынаева А.А., Цынаева Е.А. Численное исследование микроклимата помещения с использованием СПО. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 259-270. DOI: 10.15514/ISPRAS–2021–33(5)–16

Благодарности: Авторы благодарят Самарский государственный технический университет и Ульяновский государственный университет за организацию совместных исследований.

Numerical simulation of indoor microclimate using free software

[1] *A.A. Tsynaeva, ORCID: 0000-0003-4973-6169 <a.tsinaeva@gmail.com>*
[1,2] *E.A. Tsynaeva, ORCID: 0000-0001-5380-5513 <bertyf@mail.ru>*
[1] *Samara State Technical University,*
244, Mododogvardeyskaya st., Samara,443001, Russia
[2] *Ulyanovsk State University,*
42, Lev Tolstoy Street, Ulyanovsk, 432063, Russia

Abstract. The work is devoted to the numerical study of indoor microclimate. Accurately predicting the distributed microclimate inside the residential space equipped with a microclimate control system called the «smart house» allows to save thermal energy significantly. Mathematical modeling was carried out by means of the Navier-Stokes equation, the energy equation, the continuity equation. The system of equations used was closed using the k-w-sst turbulence model. The resulting numerical solution was performed in the Code_Saturn, which has a free license. The Salome free software package was used to build a grid. In this context, a second–order scheme (SOLU) was used to resolve the velocity field, a MULTIGRID scheme was used for the pressure field, automatic settings were used for the kinetic energy of turbulence and her dissipation and for the temperature field, the maximum number of iterations for each cycle was equal to 10000, the Solver Precision accuracy was 10^{-8}. The SIMPLEC algorithm is used to obtain a connected solution of the momentum balance and continuity equations. The paper provides an example of numerical solution verification, which is showed the relative temperature deviation from the values obtained by other authors was no more than 0.8-1.2%. Numerical simulation of the air velocity field in the residential space showed values from 0.12 to 0.15 m/s. Based on the results of the obtained solution, an analysis of the saving of thermal energy was carried out when regulating the supply of heat.

Keywords: free software; mathematical modeling; indoor microclimate; smart home; model verification; energy saving

Forcitation: Tsynaeva A.A., Tsynaeva E.A. Numerical simulation of indoor microclimate using free software. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 5, 2021, pp. 259-270 (in Russian). DOI: 10.15514/ISPRAS–2021–33(5)–16

Acknowledgements. We are grateful to Samara State Technical University and Ulyanovsk State University for their full cooperation.

1. Введение

Математическое моделирование широко используется при исследовании тепломассообменных процессов в энергетике и технике [1,2].

При создании проекта на стадии проектирования или реконструкции систем обеспечения микроклимата [2] численное моделирование является способом снижения затрат времени и материальных средств. Численное решение подобных задач может проводиться на базе свободного программного обеспечения (СПО), например, OpenFoam или Code_Saturne. [3,4]. Эти программные коды могут быть с графическим интерфейсом или без него. Основное достоинство СПО с открытым кодом состоит в том, что пользователь самостоятельно подбирает метод численного исследования в соответствии со своими требованиями и возможностями.

При проектировании и эксплуатации систем обеспечения микроклимата с системой «умный дом» встает задача снижения затрат тепловой энергии при условии сохранения необходимых требований комфортности в помещении [2,5,6]. Повышение эффективности таких систем может быть достигнуто корректировкой эксплуатационных параметров на основе результатов математического моделирования параметров микроклимата помещения [2, 6-8], оборудованного системой регулирования микроклимата [6-9].

Использование численного моделирования позволит повысить эффективность работы системы регулирования микроклимата за счет моделирования различных режимов

управления (режим выходного дня, режим суточного регулирования, ночные режимы и пофасадное регулирование и т.д.) [6-9].

Для проведения численного исследования авторами был выбран программный код Code_Saturne [2]. Математический аппарат Code_Saturne позволяет моделировать гидродинамику и теплообмен энергетических установок [2], исследовать вынужденную и естественную конвекцию [10], теплоотдачу в каналах с неглубокими подковообразными лунками [11].

С учетом особенностей математического моделирования микроклимата жилых помещений исследование проводится в следующем порядке:

1) формулировка математической модели;
2) верификация сформулированной математической модели;
3) математическое моделирование микроклимата помещения, оснащенного системами управления теплопотреблением;
4) обобщенный анализ полученных результатов;
5) разработка решений по повышению эффективности работы систем управления микроклиматом (умный дом) на основе результатов математического моделирования.

2. Формулировка задачи

В работе проводилось численное исследование микроклимата жилого помещения. В качестве рабочей среды принят воздух с начальной температурой $T_0 = 20^0$ С, начальным давлением $p_0 = 101325$ Па.

2.1 Математическая модель

Математическое моделирование систем обеспечения микроклимата и их элементов проводилось в Code_Saturne посредством RANS подхода.

Математическая модель состоит из уравнения сохранения массы (уравнение неразрывности), уравнения сохранения количества движения (Навье-Стокса), уравнения энергии. Уравнения используются в записи из монографии [12]:

$$\begin{aligned}
\rho\frac{du}{dt} &= g_x - \frac{\partial P}{\partial x} + \frac{\partial}{\partial x}\left[\mu\left(2\frac{\partial u}{\partial x} - \frac{2}{3}\mathrm{div}\vec{V}\right)\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)\right];.\\
\rho\frac{dv}{dt} &= g_y - \frac{\partial P}{\partial y} + \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right] + \frac{\partial}{\partial y}\left[\mu\left(2\frac{\partial v}{\partial y} - \frac{2}{3}\mathrm{div}\vec{V}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\right];\\
\rho\frac{dw}{dt} &= g_z - \frac{\partial P}{\partial x} + \frac{\partial}{\partial x}\left[\mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)\right] + \frac{\partial}{\partial y}\left[\mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right)\right] + \frac{\partial}{\partial z}\left[\mu\left(2\frac{\partial w}{\partial z} - \frac{2}{3}\mathrm{div}\vec{V}\right)\right],
\end{aligned} \tag{1}$$

где ρ – значение плотности; u, v, w – проекции вектора скорости потока соответственно на оси x, y, z; t – время; g_x, g_y, g_z – проекции вектора внешней массовой силы на координатные оси; P – давление; μ – значение коэффициента динамической вязкости.

Поскольку в рассматриваемой задаче отсутствуют источники массы, уравнение неразрывности запишется в виде:

$$\frac{d\rho}{dt} + \mathrm{div}\left(\rho\vec{V}\right) = 0\,, \tag{2}$$

уравнение энергии:

$$\rho\frac{de}{dt} = -\mathrm{div}\left(\underline{q''}\right) + q''' - P\mathrm{div}\left(\vec{V}\right) + S, \tag{3}$$

тогда диссипативную функцию S может быть описана выражением [2,12]:

$$S = \mu\left\{2\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial z}\right)^2\right] + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z}\right)^2 + \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)^2 - \right.$$
$$\left. - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)^2\right\}, \quad (4)$$

где e – внутренняя энергия; $\underline{q}''$ – вектор потока теплоты, равный $\underline{q}'' = -\lambda\underline{\nabla}T$ [2,12]; $q''' = 0$ – тепловая мощность внутренних источников теплоты; λ – значение коэффициента теплопроводности; T – температура.

Поскольку в рассматриваемой задаче рабочим телом выступает воздух, то уравнение состояния выражается уравнением Менделеева-Клайперона [12]:

$$\frac{P}{\rho} = \frac{R_m}{m_g} \cdot T, \quad (5)$$

здесь R_m – универсальная газовая постоянная; m_g – молекулярная масса.

Для замыкания системы дифференциальных уравнений применялась *k-ω sst* модель турбулентности [13]. Применение *k-ω sst* модель турбулентности проводилось с учетом опыта множества исследований [8-12].

Уравнения интегрировались по каждому контрольному объему Ω_i сетки [2], а численное решение было получено методом конечных объемов.

Численное решение было получено в программном комплексе Code_Saturne, имеющем интуитивно понятный графический интерфейс. Граничные условия в Code_Saturne могут быть заданы как условия Дирихле или Неймана. Численное решение получено итерационным методом. Для разрешения поля скорости применена схема второго порядка (SOLU), для поля давления – схема (Multigrid), так же, как в работах [10,11], для кинетической энергии турбулентности, ее диссипации и для поля температур использованы автоматические настройки, максимальное количество итераций по каждому циклу задано равным 10000, точность решателя (Solver Precision) принята 10^{-8}. Для получения связанного решения уравнений баланса импульса и неразрывности использован алгоритм SIMPLEC [10,11].

2.2 Верификация математической модели

Для верификации математической модели проводилось сравнение результатов математического моделирования с результатами, полученными другими авторами [6,14] по параметрам микроклимата тестового помещения.

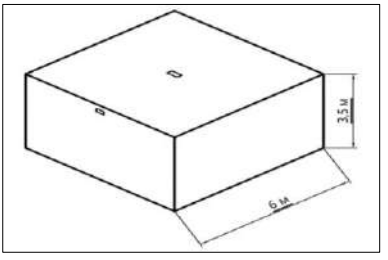


Рис. 1. Геометрия тестового помещения принятого для верификации математической модели
Fig. 1. Geometry of the room for verification of the mathematical model

Физическая постановка задачи верификации:

– помещение имеет размеры 6 м × 6 м × 3,5 м;

– в помещении установлены приточная и вытяжная решетка, для осуществления циркуляции воздуха;

– в помещении в определенных точках размещены 4 человека;

– общая сумма явных тепловыделений от людей составляет 300 Вт.

По имеющимся параметрам [6,14] была построена геометрия тестового помещения, представленная на рис. 1.

При помощи программного комплекса Salome, исходя из геометрии помещения, была создана неструктурированная расчетная сетка.

Для построения расчетной сетки в Salome использовались инструменты Tetrahedron (3D), Netgen 3D Parameters (диапазон размеров 0÷0.2 м), Netgen 1D-2D, Netgen 2D Parameters (диапазон размеров 0÷0.2 м). Полученная расчетная сетка представлена на рис. 2.

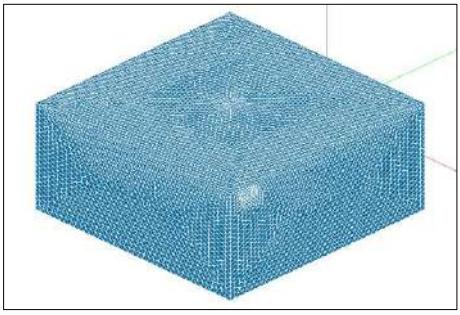


Рис. 2. Расчетная сетка для тестового помещения
Fig. 2. Calculation grid of the room

По результатам проведенного анализа сетки построенная расчетная сетка не имеет дефектов по ортогональности ячеек, по смещению центра, по градиенту наименьших объемов, по соотношению и т.д.

В качестве начальных условий моделирования приняты следующие теплофизические свойства рабочей среды (воздуха в помещении): плотность, ρ = 1,2041 кг/м^3; молекулярная вязкость, $\mu = 18.1 \cdot 10^{-6}$ Па·с; изобарная теплоемкость, c_p = 1005 Дж/(кг·К); теплопроводность, λ = 0.0259 Вт/(м·К).

На рис. 3 показано поле температур помещения, использованное для верификации результатов математического моделирования, представленное в доступных источниках [6, 14].

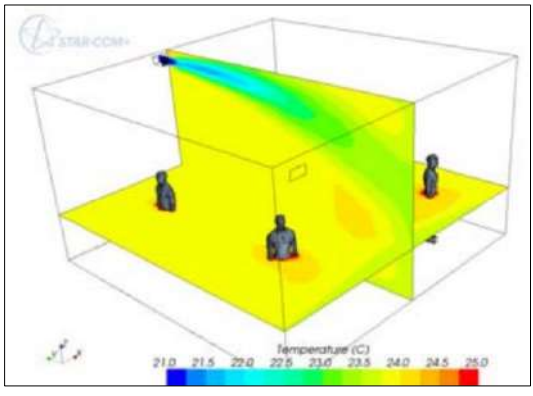


Рис. 3. Поле температур для тестового помещения из [6, 14]
Fig. 3. Temperature field of the test room from [6, 14]

Рис. 4 показывает поле температур помещения в вертикальном продольном сечении, полученное авторами при помощи СПО.

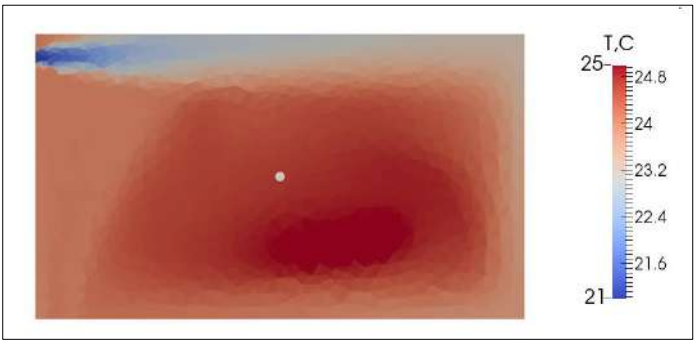


Рис. 4. Поле температур, полученное авторами, для тестового помещения в вертикальном продольном сечении
Fig. 4. The temperature field of the room in the vertical longitudinal section produced by the authors

Анализируя полученные данные, можно сказать, что температура воздуха в характерных точках вертикального продольного сечения составляет на входе 21 ºС, а вблизи человека 24.8 ºС.

При сравнении данные моделирования с результатами, полученными другими исследователями [6,14], видно, что их расхождение составляет 0,3ºС или в относительном выражении 1,2 %.

На рис. 5 показано поле температур помещения в горизонтальном сечении. Люди расположены в соответствии с условиями [6,14].

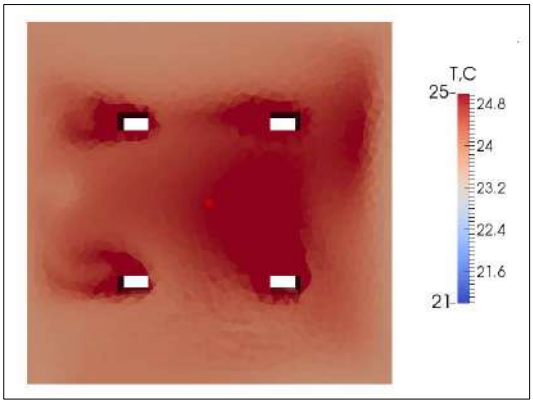


Рис. 5. Поле температур для тестового помещения в горизонтальном сечении, полученное авторами
Fig. 5. The temperature field of the room in horizontal section produced by the authors

Если сравнить значения температур в характерных точках (вблизи людей) с данными, полученными другими исследователями [6, 14], можно увидеть, что температура воздуха в этих точках у авторов получилась равной 24.8 ºС, а в исследовании [6, 14] принимала значение 25 ºС.

Таким образом, разность между значениями температур, полученными в расчете авторов и в исследованиях [6, 14] составляет 0,2 ºС или в относительном выражении 0,8%.

Это говорит о том, что принятая математическая модель адекватна и метод исследования, реализованный с помощью программного кода Code_Saturne, можно использовать для численного исследования микроклимата помещения.

3. Математическое моделирование микроклимата жилого помещения с системой «умный дом»

При обеспечении комфортных условий в помещении снижение затрат тепловой энергии возможно при использовании результатов математического моделирования для прогнозирующего управления системами регулирования микроклимата [6-8, 15], например, «умный дом».

При этом возможно снизить затраты теплоты в некоторые периоды суток, когда потребитель не находится в зданиях общественного назначения или в ночное время в жилых зданиях без нарушения условий комфортности [16].**Физическая постановка задачи**

Объект численного моделирования: жилое помещение площадью 130 м^2, находящееся на 5-м этаже 9-ти этажного дома, оконные проемы помещения выходят на южную и западную стороны света.

План объекта математического моделирования показан на рис. 6.

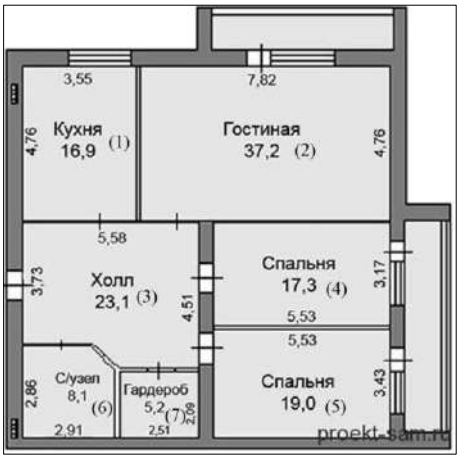


Рис. 6. План объекта математического моделирования
Fig. 6. Plan of the Residential space is an object of numerical simulation

3.2 Геометрия расчетной области и расчетная сетка

Геометрия расчетной области представлена на рис. 7.

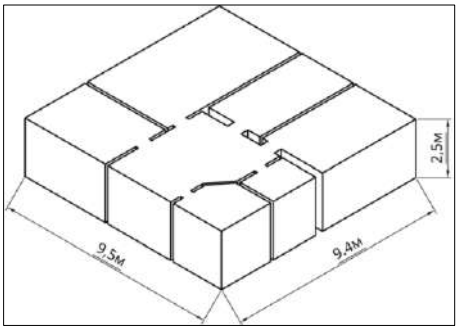


Рис. 7. Геометрия расчетной области
Fig. 7. The geometry of the estimated region

Расчетная сетка была создана при помощи программного комплекса Salome. Расчетная сетка не является структурированной и состоит из пирамидальных ячеек. Расчетная сетка показана на рис.8.

Для построения расчетной сетки применялись следующие инструменты: Tetrahedron (3D), Netgen 3D Parameters (диапазон размеров 0÷0.4 м), Netgen 1D-2D, Netgen 2D Parameters (диапазон размеров 0÷0.4 м)

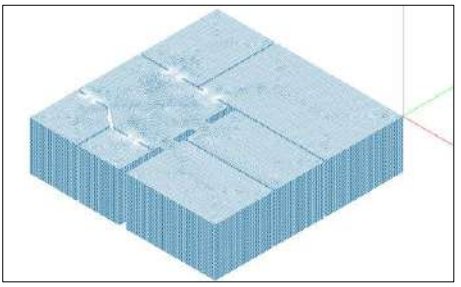


Рис. 8. Расчетная сетка
Fig. 8. Calculation grid

Проведенный анализ качества сетки с помощью встроенных инструментов программного кода Code_Saturne не выявил в ней дефектов.

3.3 Начальные и граничные условия

Начальные и граничные условия задавались в соответствии с физической постановкой задачи. Принимались следующие начальные условия:

- начальная температура среды – T = 20 °C;
- начальное давление – p_0 = 101325 Па.

Температура наружного воздуха бралась по наблюдениям в городе Самара и аппроксимировалась функцией:

$$T_н = -4 - 4{,}5\sin\left(\left(\frac{t}{3600}\right) - 40\right), \quad (6)$$

t – время, с.

Выражение для определения температуры поверхности радиаторов при наличии системы управления получено преобразованием уравнения из работы [17] путем подстановки условий проектирования объекта:

$$T_З = T_в + \Delta t' \frac{T_в - (T_н)^{0.8}}{\delta T} - 0{,}5\theta\left(\frac{T_н}{\delta T}\right), \quad (7)$$

$\delta T = T_{но} - T_в$, $T_{но}$ – проектная температура наружного воздуха для систем отопления, °C, $T_в$ – проектная температура воздуха в помещении, °C, Θ – расчетный перепад температур в местной системе отопления, °C, $\Delta t'$ – температурный напор радиатора, °C.

Согласно физической постановке задачи граничные условия задавались следующим образом: на окнах, например для поверхностей со свойствами «окно 1» (Wall) задано отсутствие шероховатости и температура поверхности, равная минус 31°C, на выходе из помещения задана поверхность «Щель 1» (Outlet), на поверхностях радиаторов задавалась динамически изменяемая во времени температура поверхности «Радиатор 1» согласно уравнению (7), наружные стены заданы, как поверхности без шероховатости с отрицательным тепловым потоком в размере 31 Вт/м^2, задана поверхность «Змеевик» с положительным тепловым потоком в 80 Вт/м^2, на входе в помещение задавался объемный поток воздуха на единицу площади.

4. Результаты численного моделирования и их обсуждение

На рис. 9 показано поле температур жилого помещения, полученное с целью прогнозирующего управления системами регулирования микроклимата при сохранении условий комфортности.

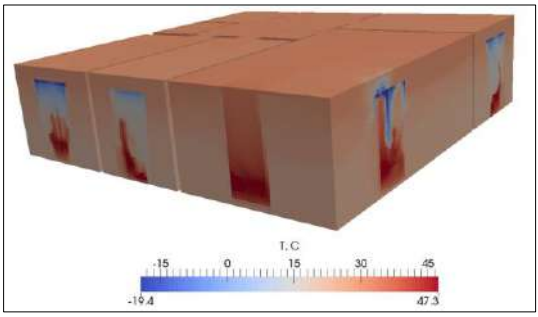


Рис. 9. Поле температур жилого помещения
Fig. 9. The temperature field of the residential space

Значения полученных температур по результатам моделирования представлены в табл. 1.

Табл. 1. Значения температур в жилом помещении
Table 1. Temperature in the residential space

Наименование помещений	**Температура, полученная в результате моделирования**	**Температура согласно нормативных требований [16]**
Кухня (1)	20,8 ^{0}C	19-21 ^{0}C
Гостиная (2)	21,2 ^{0}C	20-22 ^{0}C
Холл (3)	20,7 ^{0}C	19-21 ^{0}C
Спальня (4)	21,6 ^{0}C	20-22 ^{0}C
Спальня (5)	21,7 ^{0}C	20-22 ^{0}C
С/узел (6)	24,8 ^{0}C	24-26 ^{0}C
Гардероб (7)	19,6 ^{0}C	19-21 ^{0}C

Таким образом, математическое моделирование микроклимата позволяет применять прогнозирующее управление системами регулирования микроклимата без потери комфортности по температуре.

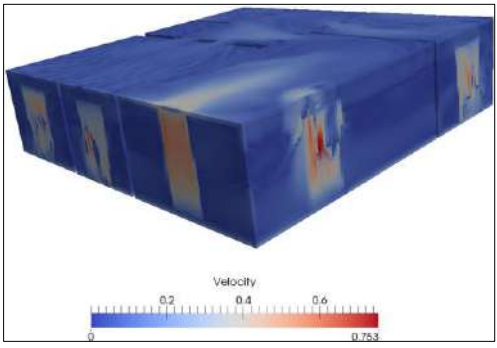


Рис. 10. Поле скоростей движения воздуха в жилом помещении
Fig. 10. The field of air velocity in the the residential space

Поля скоростей движения воздуха в жилом помещении показаны на рис. 10.

Значения полученных скоростей движения воздуха по результатам моделирования представлены в табл. 2.

Табл. 2. Значения скоростей движения воздуха в жилом помещении
Table 2. Air velocity in the residential space

Наименование помещений	**Скорость движения воздуха в жилом помещении, полученная в результате моделирования**	**Скорость движения воздуха в жилом помещениисогласно нормативных требований [16]**
Кухня (1)	0,13 м/с	0,15 м/с
Гостиная (2)	0,13 м/с	0,15 м/с
Холл (3)	0,14 м/с	0,15 м/с
Спальня (4)	0,12 м/с	0,15 м/с
Спальня (5)	0,12 м/с	0,15 м/с
С/узел (6)	0,15 м/с	0,15 м/с
Гардероб (7)	0,12 м/с	0,15 м/с

Таким образом, математическое моделирование показало, что применение систем регулирования параметров микроклимата позволяет без потери комфортности экономить около 118,75 Гкал за отопительный период от одного 9-ти этажного дома.

5. Заключение

В ходе данного исследование была выполнена математическая формулировка задачи для моделирования микроклимата жилого помещения с системами управления «умный дом».

Численное моделирование микроклимата проведено с помощью программного кода Code_Saturne с открытым программным кодом.

Проверка адекватности сформулированной математической модели осуществлена на базе сравнения результатов моделирования микроклимата тестового помещения, полученных авторами с результатами других исследователей. Выявлена адекватность сформулированной математической модели и использованных программных средств.

В результате исследования выявлено, что математическое моделирование микроклимата позволяет применять прогнозирующее управление системами регулирования микроклимата без потери комфортности в помещении. Такой подход позволит экономить около 118,75 Гкал за отопительный период с одного 9-ти этажного жилого дома.

Список литературы / References

[1] Z. Zhang, H.-S. Dou et al. Mathematical Modeling of Heat and Mass Transfer in Energy Science and Engineering. Mathematical Problems in Engineering, vol. 2013, Article ID 241646, 3 p.

[2] Ю. Табунщиков, М. Бродач. Математическое моделирование и оптимизация тепловой эффективности зданий. Москва, АВОК-ПРЕСС, 2002 г., 194 стр. / Yu. Tabunshchikov, M. Brodach. Mathematical modeling and optimization of the thermal efficiency of buildings. Moscow, AVOK-PRESS, 2002, 194 p. (in Russian).

[3] Code-Saturne. Available at: http://code-saturne.org, accessed 05.09.2021.

[4] Open Foam. Available at: http://www.openfoam.com/, accessed 05.09.2021.

[5] I. Sultanguzin, H. Toepfer et al. Mathematical modeling and control system of nearly zero energy building. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, vol. 8, no. 2, 2018, pp. 21-24.

[6] Д.М. Денисихина. Оценка теплового комфорта в помещениях на основе анализа результатов математического моделирования. Вестник Томского государственного архитектурно-строительного университета, no. 3, 2015 г., стр. 183-193 / D.M. Denisikhina. Mathematical simulation of room climate parameters. Vestnik of Tomsk state university of architecture and building, no. 3, 2015, pp. 183-193 (In Russian).

[7] N. Fumo, P. Mago, and R. Luck, Methodology to estimate building energy consumption using EnergyPlus Benchmark Models. Energy and Buildings, vol. 42, issue 12, 2010. pp. 2331-2337.

[8] X. Jiang, Z. Jing et al. Modelling and operation optimization of an integrated energy based direct district water-heating system. Energy, vol. 64, 2014. pp. 375-388.

[9] S. Cho, M. Zaheer-uddin. Predictive control of intermittently operated radiant floor heating systems, Energy Conversion and Management, vol. 44, issue 8, 2003. pp. 1333-1342.
[10] M. Nikitin. Modeling of natural convection. In Proc. of the 2nd International Conference on Industrial Engineering, Applications and Manufacturing, 2016, pp. 1-4.
[11] А. Цынаева, С. Разоренов, В. Белая. Численное исследование теплоотдачи в каналах с неглубокими подковообразными лунками. Труды ИСП РАН, том 29, вып. 5, 2017 г., стр. 329-344 / A. Tsynaeva, S. Razorenov, V. Belaya. Numerical modeling of heat transfer of channel with shallow curly dimples. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 5, 2017. pp. 329-344 (in Russian). DOI: 10.15514/ISPRAS-2017-29(5)-16.
[12] Н. Ковальногов. Прикладная механика жидкости и газа. Ульяновск, Изд. УлГТУ, 2010 г., 219 стр. / N. Kovalnogov. Applied Mechanics of Liquid and Gas. Ulyanovsk, Ed. UlSTU, 2010, 219 p. (in Russian).
[13] F. Menter. Two-Equation Eddy-Viscosity Turbulence Models for Engineering Applications. AIAA Journal, vol. 32, № 8, 1994, pp. 1598-1605.
[14] Д.М. Денисихина. Модель человека в задачах расчета распределенных параметров микроклимата в помещении. Известия Казанского государственного архитектурно-строительного университета, no. 2, 2015 г., стр. 192-199 / D.M. Denisikhina. Human shapes for CFD simulation of thermal environment in rooms. News of the Kazan State University of Architecture and Engineering, no. 2, 2015, pp. 192-199 (in Russian).
[15] D.Craft. Local energy management through mathematical modeling and optimization. PhD Thesis. Massachusetts Institute of Technology, 2004, 223 p.
[16] ГОСТ 30494-2011. Здания жилые и общественные. Параметры микроклимата в помещениях / GOST 30494-2011. Residential and public buildings. Microclimate parameters for indoor enclosures (in Russian).
[17] Н. Ковальногов, А. Цынаева, Е. Цынаева. Способ выбора места установки регулирующего устройства в автоматизированных системах управления отоплением. Патент на изобретение RU 2340834 C1, 2008 г. / N. Koval'nogov, A. Tsynaeva, E. Tsynaeva. Method of control system placing in automated heat control systems. Patent for invention RU 2340834 C1, 2008 (in Russian).

Информация об авторах/ Information about authors

Анна Александровна ЦЫНАЕВА – кандидат технических наук, доцент, доцент кафедры «Теплогазоснабжение и вентиляция». Сфера научных интересов: численное моделирование устройств и объектов теплоэнергетики и теплотехники.

Anna Alexandrovna TSYNAEVA – Doctor of Philosophy in Technical Sciences, Associate Professor. She is Associate Professor of the Department of Heat and Gas Supply and Ventilation. Research interests: numerical simulation of devices and objects of thermal power engineering and heat engineering, mathematical modeling and optimization

Екатерина Александровна ЦЫНАЕВА – кандидат технических наук, доцент, доцент кафедры физических методов в прикладных исследованиях Ульяновского государственного университета, доцент кафедры «Теплогазоснабжение и вентиляция» Самарского государственного технического университета. Сфера научных интересов: численное моделирование устройств и объектов теплоэнергетики и теплотехники.

Ekaterina Alexandrovna TSYNAEVA – Doctor of Philosophy in Technical Sciences, Associate Professor.She isAssociate Professor of the Department of Physical Methods in Applied Research of Ulyanovsk State University. She is Associate Professor of the Department of Heat and Gas Supply and Ventilation of Samara State Technical University. Research interests: numerical simulation of devices and objects of thermal power engineering and heat engineering, mathematical modeling and optimization.

DOI: 10.15514/ISPRAS-2021-33(5)-17

Тепловая нагрузка биметаллической оребренной трубки

[1] *Е.С. Байметова, ORCID: 0000-0002-4534-0936 <pushina.e.s@gmail.com>*
[1] *А.Ф. Гиззатуллина, ORCID: 0000-0003-4511-6122 <gialfi@mail.ru>*
[2] *М.Р. Королева, ORCID: 0000-0001-5697-9199 <koroleva@udman.ru>*
[1] *О.В. Мищенкова, ORCID: 0000-0002-8835-1184 <mov@istu.ru>*
[1] *Ф.Н. Пушкарев, ORCID: 0000-0002-7666-5522 <feddokk1great@gmail.com>*
[1] *А.А. Чернова, ORCID: 0000-0001-8579-6279 <alicaaa@gmail.com>*

[1] *Ижевский государственный технический университет имени М.Т. Калашникова, 426000, Россия, г. Ижевск, ул. Студенческая, д. 7*
[2] *Удмуртский федеральный научный центр, 426000, Россия, Ижевск, ул. Т. Барамзиной, д. 34*

Аннотация. Работа посвящена вопросам численного моделирования сопряженной задачи теплообмена в установках замкнутого типа при использовании в качестве рабочих элементов оребренных биметаллических трубок с использованием инструментов пакета openFoam. Моделирование процесса теплопереноса в биметаллических трубках сопряжено с решением задачи определения контактного термического сопротивления на границе металл/металл. Рассматриваемая конструкция биметаллической трубки предполагает обжим медных шайб по поверхности алюминиевой цилиндрической трубы, вследствие чего контактная поверхность трубки является не изотропной по своим свойствам. Предложена математическая модель сопряженного теплообмена для сред воздух/биметалл/хладагент. Показаны особенности организации теплофизических процессов на границе контакта металлов и на границах сопряжения металл/воздух и металл/хладагент. Приведенные распределения температур и тепловых потоков позволяют оценить вклад каждого отдельного ребра в исследуемый процесс теплоотвода от воздушной среды. Показана эффективность рассмотренной технологии изготовления биметаллической оребренной трубки.

Ключевые слова: математическое моделирование; сопряженный теплообмен; биметалл; оребренная трубка

Для цитирования: Байметова Е.С., Гиззатуллина А.Ф., Королева М.Р., Мищенкова О.В., Пушкарев Ф.Н., Чернова А.А. Тепловая нагрузка биметаллической оребренной трубки. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 271-282. DOI: 10.15514/ISPRAS–2021–33(5)–17

Благодарности: Исследование выполнено при финансовой поддержке ИжГТУ имени М.Т. Калашникова в рамках выполнения научного проекта №ЧАА/20-30-07.

Heat load of bimetallic ribbed tube

[1] *E.S. Baimetova, ORCID: 0000-0002-4534-0936 <pushina.e.s@gmail.com>*
[1] *A.F.Gizzatullina, ORCID: 0000-0003-4511-6122 <gialfi@mail.ru>*
[2] *M.R. Koroleva, ORCID: 0000-0001-5697-9199 <koroleva@udman.ru>*
[1] *O.V. Mishchenkova, ORCID: 0000-0002-8835-1184 <mov@istu.ru>*
[1] *F.N. Pushkarev, ORCID: 0000-0002-7666-5522 <feddokk1great@gmail.com>*
[1] *A.A. Chernova, ORCID: 0000-0001-8579-6279 <alicaaa@gmail.com>*

[1] *Kalashnikov Izhevsk State Technical University*
7, Studencheskaya st., Izhevsk, 426000, Russia
[2] *Udmurt Federal Research Center of the Ural Branch of the Russian Academy of Sciences,*
34, T. Baramzinoy st., Izhevsk, 426000, Russia

Abstract. This study is devoted to the problem of numerical modeling of the conjugate heat transfer in a closed-type power installation. The working elements of that are ribbed bimetallic tubes using the openFoam toolbox. The heat transfer process modeling in bimetallic tubes is associated with solving the problem of determining the value of the contact thermal resistance at the metal / metal interface. Considered design of a bimetallic tube involves crimping copper washers on the surface of an aluminum cylindrical tube. Hence, the contact surface of the tube is not isotropic in its properties. A mathematical model of conjugate heat transfer for air / bimetal / coolant medium is proposed. The features of the organization of thermophysical processes at the metal contact interface and at the metal / air and metal / coolant medium are shown. A qualitative comparison of the obtained results with the famous experimental data is carried out. Generalized temperature profiles in the rib longitudinal section are obtained by mathematical modeling. The given distributions of temperature and heat flux make it possible to estimate the contribution of each individual rib to the investigated heat removal process from the air environment. The efficiency of the considered technology of manufacturing a bimetallic finned tube is shown.

Keywords: mathematical modeling; conjugate heat transfer; bimetal; ribbed tube

For citation: Baimetova E.S., Gizzatullina A.F., Koroleva M.R., Mishchenkova O.V., Pushkarev F.N., Chernova A.A. Heat load of bimetallic ribbed tube. Труды ИСП РАН, том 33, вып. 5, 2021 г., стр. 271-282. DOI: 10.15514/ISPRAS–2021–33(5)–17

Acknowlribments. The research was carried out with the financial support of Kalashnikov ISTU within the framework of the scientific project No. ЧАА/20-30-07.

1. Введение

Неослабевающий [1-5] интерес исследователей к теплообменным аппаратам различных конструкций связан с активным внедрением оборудования требующего терморегулирования рабочих сред. Одним из наиболее распространенных видов теплообменных аппаратов [6] являются воздушные теплообменники, состоящие из пакета оребренных трубок. Простота конструкции и возможность управления ее эффективностью за счет компоновки пакета обеспечивает ее широкое распространение. При этом одним из методов дополнительного повышения эффективности теплосъема в таких устройствах является переход от монометалла, используемого в качестве материала для трубок, к биметаллу.

Оребрение гладкой трубы, в случае использования двух различных материалов для несущей трубы и ребер может быть выполнено с помощью ламелей, спиралевидных или кольцевых ребер. При этом, эффективность биметаллической связки достигается только за счет внешнего более теплопроводного слоя. В зависимости от типа оребрения используется тот или иной метод изготовления биметаллических трубок [7]. При спирально-винтовой навивке металлическая лента либо вдавливается в наружную поверхность несущей трубы, либо крепление проводится путём сваривания контактных поверхностей после их разогрева током высокой частоты. В случае изготовления методом накатки на несущую трубку крепится

дополнительная муфта, выполненная из другого материала, на которой методом накатывания выдавливаются рёбра (рис. 1). Для формирования кольцевого поперечного оребрения используется также метод раздачи труб гидравлическим давлением (протягивание дорна) (рис. 2). В этом случае контактная поверхность состоит из двух металлов, например, медных ребер и алюминиевой цилиндрической межреберной поверхности. Такая составная поверхность может обеспечить больший удельный теплосъем, однако ее применение [8, 9] ограничено недостаточной степенью изученности процессов теплопередачи.

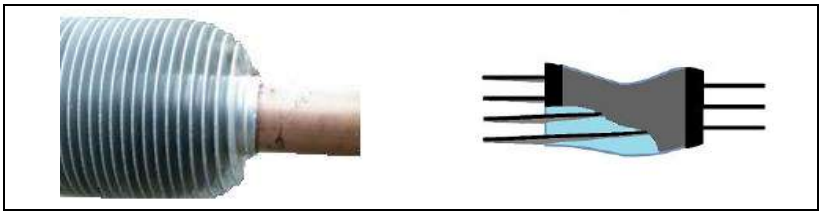


Рис. 1. Биметаллическая оребренная трубка, полученная по технологии напрессовки и накатки
Fig. 1. Bimetallic ribbed tube formed by pressing and knurling method

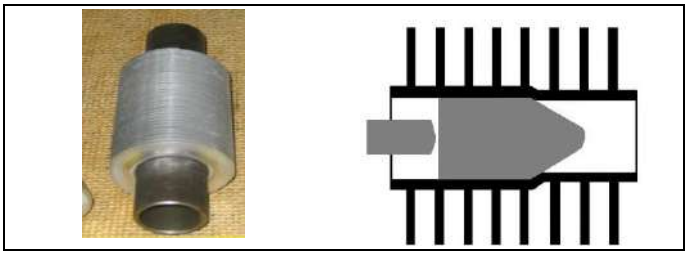


Рис. 2. Биметаллическая оребренная трубка, полученная по технологии раздачи трубы гидравлическим давлением
Fig. 2. Bimetallic ribbed tube formed by hydraulic expansion method

2. Математическая постановка

Процесс охлаждения внешнего потока при помощи биметаллических оребренных элементов, по которым движется хладагент, исследуется в рамках задачи сопряженного теплообмена одной оребренной трубки при задании условий симметрии на боковых гранях расчетной области, моделирующих, по сути, один поперечный ряд оребренных трубок. Геометрическая модель состоит из 44 доменов, характеризующихся различными теплофизическими свойствами (табл. 1). Первые два домена относятся к текучим сплошным средам. Это внешний охлаждаемый поток воздуха (gas), моделирующийся в рамках модели сжимаемой теплопроводной вязкой среды, и охлаждающая жидкость (liquid) – этиленгликоль, описываемая моделью вязкой несжимаемой теплопроводной жидкости. Рассматриваемая биметаллическая трубка выполнена в связке алюминий-медь. Отдельным доменом выступает цилиндрическая алюминиевая трубка (tube). Остальные домены – кольцевые медные ребра, стыкующиеся с несущей трубкой по внутренней поверхности (ribs).

Табл. 1. Теплофизические свойства материалов
Table 1.Thermal and physical properties of the materials

Параметр	Значение	Параметр	Значение
Этиленгликоль (70%)		*Алюминий*	
Плотность	1100 кг/м^3	Плотность	2700 кг/м^3
Теплоемкость	3100 Дж/(кг*K)	Теплоемкость	920 Дж/(кг*K)
Вязкость	0.11 Па*с	Теплопроводность	236 Вт/(м*K)

Теплопроводность	0.32 W/(m*K)		
Воздух		*Медь*	
Теплоемкость	1007 Дж/(кг*K)	Плотность	8930 кг/м^3
Вязкость	18.5 мкПа*с	Теплоемкость	385 Дж/(кг*K)
Теплопроводность	0.025 Вт/(м*K)	Теплопроводность	401 Вт/(м*K)

Несущая трубка толщиной 2 мм выполнена из алюминия. Кольцевые медные ребра толщиной 1 мм и высотой 2 мм, размещены на трубке через каждые 5 мм. Таким образом, на трубке длиной 25 см размещено 41 ребро. Конфигурация скрепления несущей цилиндрической трубки с оребрением соответствует методу раздачи труб гидравлическим давлением (рис. 3).

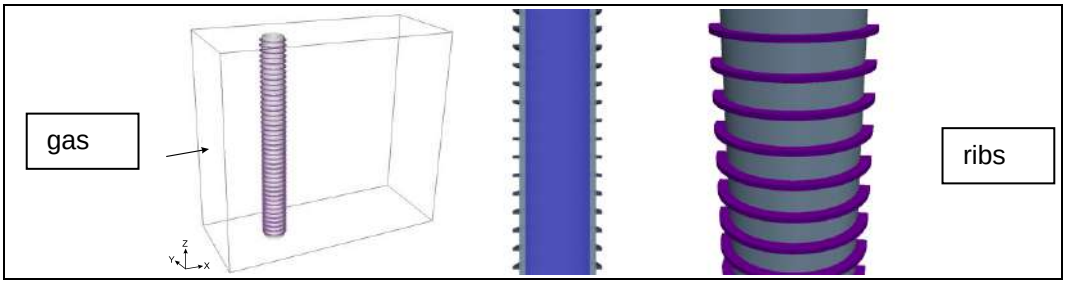


Рис. 3. Расчетная область
Fig. 3. Calculation domain

Моделирование проводится на основе решения уравнений сохранения текучих сред для доменов gas-liquid совместно с уравнением теплопроводности в твердом теле для областей tube-ribs.

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho u_i}{\partial x_i}=0 \tag{1}$$

$$\frac{\partial \rho u_i}{\partial t}+\frac{\partial \rho u_i u_j}{\partial x_j}=-\frac{\partial p}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\mu\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)-\frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}\right)+F_i \tag{2}$$

$$\frac{\partial \rho E}{\partial t}+\frac{\partial \rho H u_j}{\partial x_j}=\frac{\partial u_i \tau_{ij}}{\partial x_j}+\frac{\partial q_j}{\partial x_j}+F_i u_i \tag{3}$$

В системе уравнений (1)-(3) приняты следующие допущения: ρ – плотность среды; u_i - компоненты вектора скорости u; p – давление; μ - динамический коэффициент вязкости; F_i – внешняя массовая сила, задается для хладагента для учета гравитации; $E=C_vT+0.5u_i^2$ – полная удельная энергия среды; $H=E+p/\rho=C_pT+0.5u_i^2=h+0.5u_i^2$ – полная удельная энтальпия; $\tau_{ij}=2\mu S_{ij}-\frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij}$ – тензор вязких напряжений; $S_{i,j}=\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right)$ - тензор скоростей деформаций; $q_j=\lambda\frac{\partial T}{\partial x_j}$ - тепловой поток; λ - коэффициент теплопроводности среды; T – температура.

В случае моделирования движения газообразных сжимаемых сред система уравнений (1) – (3) дополнялась уравнением состояния идеального газа

$$p=\rho RT\,, \tag{4}$$

где R - удельная газовая постоянная, а в случае моделирования несжимаемой жидкости – уравнением постоянства плотности среды

$$\rho = const\ . \tag{5}$$

Процесс переноса тепла в трубке описывается на основе уравнения

$$\rho c_p \frac{\partial T}{\partial t} = \frac{\partial}{\partial x_i} \lambda \frac{\partial T}{\partial x_i}, \tag{6}$$

здесь c_p - удельная теплоемкость материала.

Основные граничные условия, использующиеся при численном моделировании, показаны в табл. 2. Кроме этого на непроницаемых стенках использовалось условие прилипания и условие симметрии на боковых границах домена gas.

Течение хладагента происходит сверху вниз с учетом действия объемной силы тяжести. При этом характеристики жидкости и средняя скорость течения позволяют моделировать движение в ламинарной постановке (число Рейнольдса равно 1000). В то же время течение воздуха с числом Рейнольдса равным примерно 8000 соответствует переходным режимам движения сред и требует учета эффектов турбулентности. Поэтому для расчета параметров в домене gas подключалась модель турбулентности Ментера [10]. Более подробное описание математической постановки исходной задачи сопряженного теплообмена можно найти в работе [11].

Табл. 2. Граничные условия

Table 2. Boundary conditions

Домены/границы	tube	ribs	вход	выход
gas	$\lambda_g \frac{\partial T_g}{\partial n} = \lambda_t \frac{\partial T_t}{\partial n}$ $T_g = T_t$	$\lambda_g \frac{\partial T_g}{\partial n} = \lambda_e \frac{\partial T_e}{\partial n}$ $T_g = T_e$	$u_g = 4$ м/с $T_g = 333$ К	$p_g = 10^5$ Па
liquid	$\lambda_l \frac{\partial T_l}{\partial n} = \lambda_t \frac{\partial T_t}{\partial n}$ $T_l = T_t$	-	$u_l = -1$ м/с $T_l = 213$ К	$p_l = 10^5$ Па
tube		$\lambda_t \frac{\partial T_t}{\partial n} = \lambda_e \frac{\partial T_e}{\partial n}$ $T_t = T_t$	-	-

Расчетная сетка построена с использованием утилит построения сеток blockMesh и topoSet пакета openFoam и содержит более 16 млн призматических элементов (рис.4). Домен gas содержит 11 млн., домен tube – 960 тыс., домен liquid – 3600 тыс. ячеек. Каждое ребро содержит 1920 ячеек. На границы сопряжения газ-металл приходится 172 тыс. граней, металл-жидкость 120 тыс. граней, на границу металл-металл 19 тыс. граней. Сетки на границах сопряжения gas/ribs/tube/solid являются согласованными.

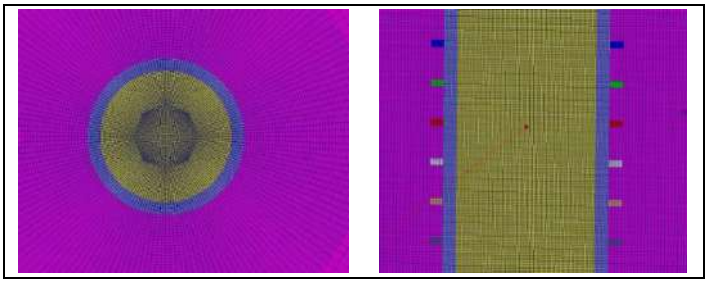


Рис. 4. Расчетная сетка

Fig. 4. Calculation mesh

Численное моделирование проводится на основе решателя chtMultiRegionSimpleFoam – стационарного решателя для моделирования задач сопряженного теплообмена. Для решения уравнения теплопроводности в трубке и ребрах используется стандартный метод сопряженных градиентов PCG. Для разрешения уравнений импульса, энергии, а также уравнений переноса турбулентных характеристик потока используется геометро-алгебраический многосеточный метод GAMG со сглаживанием по методу Гаусса-Зейделя. Используемые схемы приведены в таблице 3.

Исходная область была разделена на части для запуска решателя chtMultiRegionSimpleFoam в параллельном режиме. Использовался метод декомпозиции области scotch с разбивкой на 5 процессов.

Табл. 3. Расчетные схемы
Table 3. Numerical schemes

	Тип схемы	Ограничение
Gradient	cellLimited leastSquares	1
Divergence	bounded Gauss upwind	-
Laplacian	Gauss linear limited corrected	0.33
Interpolation	linear	-
Surface normal gradient	limited	0.5

Подробная валидация численных схем для задач внутренней гидродинамики приведена в работе [12], для задач внешней аэродинамики в работе [13], кроссплатформенная верификация решения задачи сопряженного теплообмена для пакета оребренных трубок представлена в работе [14].

3. Результаты расчета

При обтекании воздушным потоком одиночной оребренной трубки в донной части трубки, в области отрыва потока, образуются симметричные возвратные течения, конфигурация которых определяется отношением диаметра ребра к диаметру трубки. Расчетная структура (рис.5) качественно совпадает с экспериментальной, представленной в работе [15].

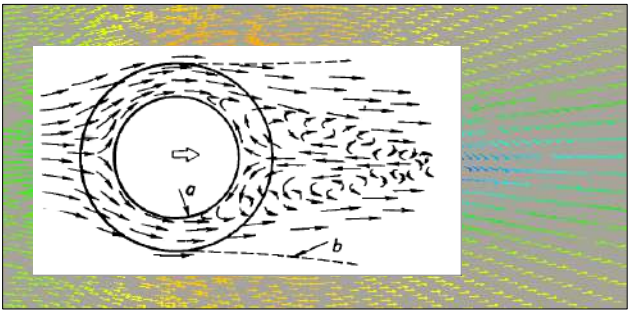
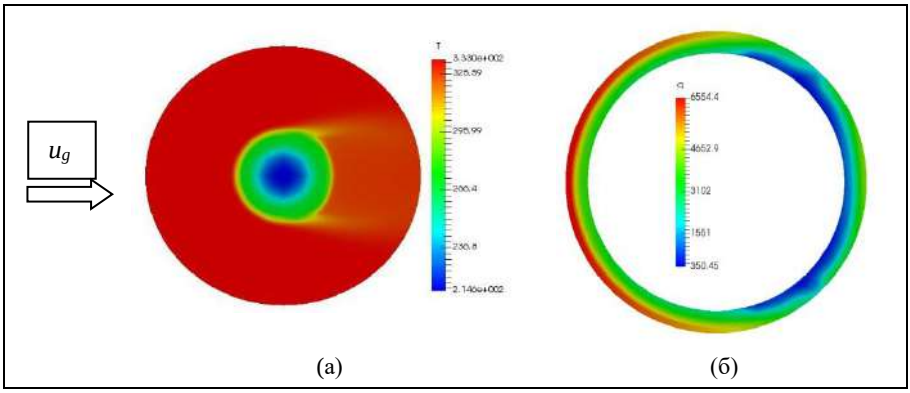


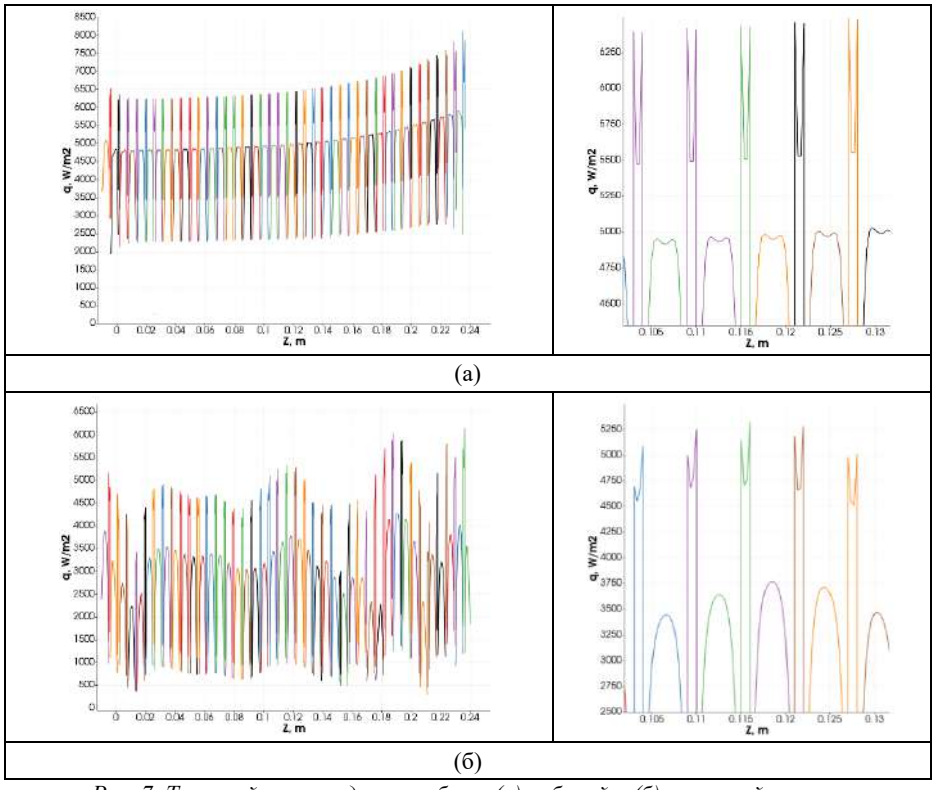
Рис. 5. Сопоставление расчетного и экспериментального [12] векторных полей
Fig. 5. Results comparison with experiment

В области отрыва потока (рис.5, область а) наблюдается незначительное уменьшение температуры (рис.6, а), обусловленное как образованием низкоскоростных возвратных течений, так и вторичным теплосъемом с донной части трубки. Положение области отрыва потока определяет область минимального теплосъема на внешней поверхности оребренной трубки (рис.6,б). Область контакта с набегающим потоком газа на переднюю кромку вплоть до боковой поверхности трубки обеспечивает максимальный теплосъем.



(а) (б)

Рис. 6. Поле температур в сечении между ребрами (а) и тепловой поток на поверхности ребра (б)
Fig. 6. Temperature field between pair of ribs (a) and specific heat flux on the middle rib surface (b)



(а)

(б)

Рис. 7. Тепловой поток вдоль трубки в (а) лобовой и (б) кормовой точках
Fig. 7. Wall heat flux on the tube outer surface: (a) at the front point, (b) at the back point

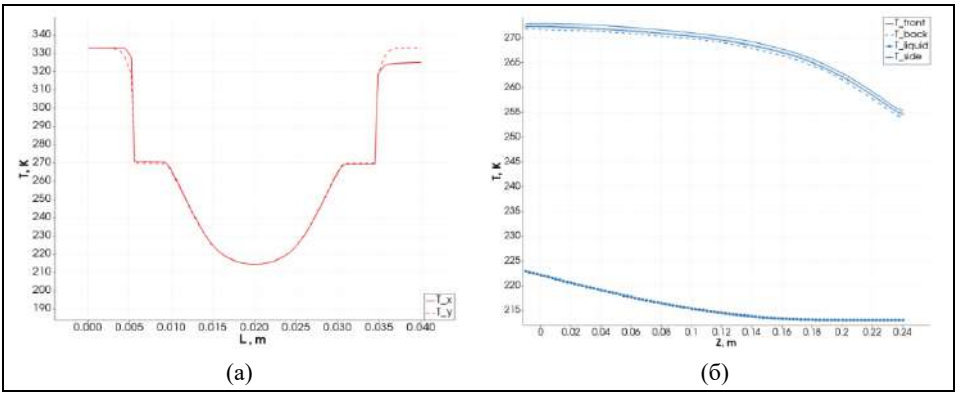
Прогрев ребер и цилиндрической части трубки по длине не равномерен. На рис. 7 показано изменение удельного теплового потока по длине трубки в лобовой и кормовой точках. Вследствие ударного взаимодействия газового потока на переднюю часть трубки, тепловой поток в местах контакта с алюминиевой стенкой имеет ступенчатый характер изменения. На торцах медных ребер виден заметный спад теплового потока, в то время как на кромках наблюдается его резкое увеличение. В среднем, характер распределения удельного теплового потока для всех ребер и межреберных участков трубки одинаков. В отличие от лобовой части, в кормовой точке трубки теплосъем имеет более неравномерный характер (рис. 7б), что

обусловлено режимом течения за трубкой, а именно сформированными возвратными зонами и их неоднородностью по длине трубки. Уровень теплового потока на боковых поверхностях трубки также носит неравномерный характер и изменяется от 1000 до 7000 Вт/м^2.

U-образный профиль изменения температурного потока по высоте ребра может быть связан со срывом теплового пограничного слоя в области острой кромки ребра, при этом из рис.7б видно, что распределение теплового потока по поверхности алюминиевой трубки между медными ребрами имеет косинусоидальную форму, что косвенно свидетельствует о перераспределении тепловых потоков в межреберном пространстве.

Сопоставление температурных профилей, построенных по линиям, направленным вдоль (кривая T_x) и поперек (кривая T_y) направлению движения потока газа и проходящих одновременно по газу, медному ребру, алюминиевой трубке и жидкости (рис. 8) позволяет оценить, как теплопроводность на границах gas/ribs, ribs/tube и tube/liquid в рассматриваемых областях, так и непосредственно оценить скорость и равномерность прогрева ядра потока хладагента. Несмотря на то, что в кормовой части трубки наблюдается самое значительное падение температуры газа, более интенсивно процесс теплоотдачи протекает на боковых поверхностях трубки.

Распределение температуры в трубке и по продольному сечению хладагента представлено на рис. 8б. Показаны профили температуры в лобовой (кривая T_front), кормовой (кривая T_back) зонах, в боковой части (кривая T_side) трубки, а также в ядре потока (кривая T_liquid). Распределение температур по высоте медного ребра, в сравнении с изменением температуры алюминиевой трубки позволяет оценить падение температуры потока газа при первичном взаимодействии с ребром (в лобовой области, соответствующей зоне торможения потока), составляющее от 273 К в верхней части трубки (на входе потока хладагента в трубку) до 255 К (при температуре набегающего потока 325 К и температуре на поверхности трубки от 264 К до 268 К), а падение температуры потока в зоне вторичного взаимодействия (в донной части, в областях возвратных течений) составляет, в среднем, 42 К, при средней температуре воздуха в возвратных течениях – 307 К и изменении температуры на поверхности трубки от 263 К до 267 К.



(а) (б)

Рис. 8. Прогрев стенок трубки и жидкости: а) поперек трубки и б) по длине трубки
Fig. 8. Wall and liquid heating-up: (a) across tube and (b) along tube

Объемное изменение температуры представлено на рис. 9. Эффективное охлаждение набегающего потока газа единичной трубкой обеспечено только на участке до 1/3 длины трубки (то есть, в рассматриваемом случае, примерно, до 0.08 м), далее наблюдается устойчивый прогрев хладагента (рис. 8) с увеличением мощности теплового слоя в трубке до 80% от ее диаметра к выходному сечению трубки (0.25м).

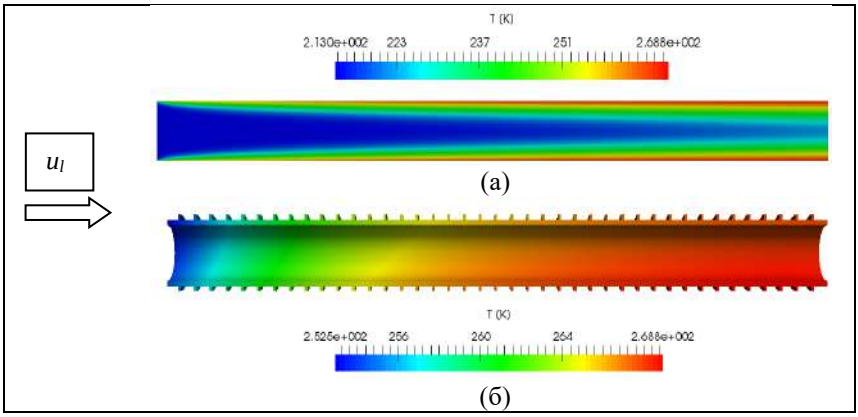


Рис. 9. Прогрев жидкости (а) в продольном направлении и (б) прогрев стенки
Fig. 9. Liquid heating-up (a) and wall heating-up (b)

Сопоставление графических данных на рис. 6-9 позволяет однозначно определить достигаемый первичный перепад температур при взаимодействии потока воздуха с охлаждающим элементов в виде оребренной трубки. Без учета зоны восстановления потока (в том числе восстановления температурного поля) за участком стабилизации течения после области возвратных течений, то есть в зоне лобовая область трубки/донная область трубки, достигаемый перепад температур, для рассматриваемого режима охлаждения, составляет (10-25) К на внешней кромке медных ребер и (1-2) К на поверхности алюминиевой трубки. Такое распределение температур косвенно свидетельствует как о работоспособности предложенной конструкции биметаллической ореберенной трубки, так и об эффективности применения меди в качестве материла для кольцевого оребрения.

Необходимо отметить, что согласно [16] для расчета контактного сопротивления в сложной системе может быть использована следующая зависимость:

$$R_t = R_{t1} + R_{t2} = \frac{l_1}{\lambda_1 S_1} + \frac{l_2}{\lambda_2 S_2},$$

где l - длина участка тепловой цепи, м (высота медного ребра (1) и толщина алюминиевой трубки (2)), S - площадь поперечного сечения участка, м² (площадь кромки медного ребра и площадь поверхности межреберной поверхности алюминиевой трубки). Очевидно, что использование данного типа зависимости для расчета термического сопротивления оребренной трубки крайне нецелесообразно в виду его неудобства для сложной многосоставной системы. Альтернативная форма классического определения термического сопротивления предполагает учет перепада температуры и отводимого теплового потока:

$$R_t = R_{t1} + R_{t2} = \frac{(T_2 - T_1)_1}{q_1} + \frac{(T_2 - T_1)_2}{q2},$$

согласно которому среднее термическое сопротивление системы медное ребро – алюминиевая трубка (для единичного ребра) в рассмотренном случае составляет 0.011 м^2·К/Вт. При этом контактное сопротивление пары медь – алюминий в рассматриваемой системе остается неоцененным. Согласно работе [7], контактное термическое сопротивление может быть оценено аналогичной зависимостью, где в качестве перепада температур берется разница температур вблизи границы контакта, там же оценивается и подводимый тепловой поток. В этом случае, контактное термическое сопротивление на 1 ребро для рассматриваемой системы изменяется в диапазоне (0.0005÷0.002) м^2·К/Вт.

4. Заключение

В работе проведено численное моделирование сопряженного теплообмена между внешним потоком горячего воздуха (температура 333 К) и хладагентом (температура 213 К), текущим по ореберенной биметаллической трубке. Моделирование проводится с использованием инструментов пакета OpenFoam на основе решателя chtMultiRegionSimpleFoam, разработанного для решения задач сопряженного теплообмена.

Особенностью рассматриваемой биметаллической трубки является нестандартный тип оребрения, получаемый методом раздачи труб гидравлическим давлением. Данный тип производства оребренных трубок позволяет избежать наличия дополнительного металлического слоя, утяжеляющего рабочий элемент.

В работе сформулирована математическая модель сопряженного теплообмена для сред воздух/биметалл/хладагент. Показаны особенности организации теплофизических процессов на границе контакта металлов и на границах сопряжения металл/воздух и металл/хладагент и подтверждена эффективность технологии изготовления биметаллической оребренной трубки методом раздачи трубки гидравлическим давлением. Анализ результатов численного моделирования процесса охлаждения потока воздуха, при его взаимодействии с оребренной биметаллической трубкой, во внутренний канал которой подается хладагент, позволил оценить контактное термическое сопротивление на каждое из ребер трубки на границе алюминий/медь. Также получены оценки для термических сопротивлений системы в целом, справедливые для оребренных трубок алюминий/медь, имеющих не изотропную внешнюю поверхность.

Список литературы / References

[1]. Федоров В.А., Мильман О.О. и др. Результаты экспериментально-расчетных исследований воздушного потока в цирктрассах воздушных конденсаторов паротурбинных установок. Вестник МГТУ им. Н.Э.Баумана. Сер. Машиностроение, № 5, 2015 г., стр. 87-105 / Fedorov V. A., Mil'man O. et al. Results of experimental and computational analysis of air flow in circle channels of air-cooled condensers of steam power plants. Herald of the Bauman Moscow State Technical University. Series Mechanical Engineering, № 5, 2015, pp. 87-105 (in Russian).

[2]. Du Toit C.G., Kroger D.G. Modelling of the recirculation in mechanical-draught heat exchangers. N&O JOERNAAL, vol. 9, № 1, 1993, pp. 2-8.

[3]. Zhao W., Wang Q., Liu P. The experimental investigation of recirculation of air-cooled system for a large power plant. Energy and Power Engineering, № 2, 2010, pp. 291-297.

[4]. Xing X., Xianming F. et al. Modeling and Simulation of an Air-cooling Condenser under Transient Conditions. Procedia Engineering, № 31, 2012, pp. 817–822.

[5]. Чернов Н.С. Технико-экономическая оценка эффективности теплообменных аппаратов. Автомобильная промышленность, №3, 2011 г., стр. 33-35 / Chernov N. S. Evaluation of efficiency of heat exchangers (pipe coil) from pipes with edges on an external surface. Avtomobil'naya promyshlennost', № 3, 2011, pp. 33-35 (in Russian).

[6]. Koroleva M.R., Saburova E.A., Chernova A.A. Studying the efficiency of cooling and resistance of ribbed tubular elements. Journal of Physics: Conference Series, vol. 1675, 2020, article no. 012009.

[7]. Кунтыш В.Б, Пиир А.Э. Анализ тепловой эффективности, объемной и массовой характеристик теплообменных секций аппаратов воздушного охлаждения. Химическое и нефтегазовое машиностроение, no. 5, 2009 г., стр. 3-6. / Kuntysh V.B, Piir A.E. Analysis of the thermal efficiency, volume, and weight characteristics of heat exchange sections of air cooling equipment. Chemical and Petroleum Engineering, vol. 45, no. 5-6, 2009, pp. 257-262.

[8]. Дударев В.В., Филатов С.О., Карлович Т.Б. Методика расчета и анализ коэффициента теплопередачи биметаллических ребристых труб аппаратов воздушного охлаждения с неравномерным внешним загрязнением. Энергетика. Известия высших учебных заведений и энергетических объединений СНГ, том 60, № 3, 2017 г., стр. 237-255. / Dudarev V. V., Filatau S. O., Karlovich T. B. The Method of Calculation and Analysis of Heat Transfer Coefficient of Bimetallic Finned

Tubes of Air Cooling Units with Irregular External Contamination. Energetika. Proceedings of CIS higher education institutions and power engineering associations, vol. 60, № 3, 2017, pp. 237-255 (in Russian).

[9]. Бессонный А.Н., Кунтыш В.Б., Калейчик Т.П. Методика расчета тепловой и аэродинамической характеристик калориферов из биметаллических труб со спиральными накатными и навитыми алюминиевыми ребрами. Вестник Международной академии холода, no. 3, 2004 г., стр. 16-20. / Bessonnyj A.N., Kuntysh V.B., Kalejchik T.P. Method for calculating the thermal and aerodynamic characteristics of air heaters made of bimetallic pipes with spiral rolling and wound aluminum fins. Journal of International Academy of Refrigeration, no. 3, 2004, pp. 16-20 (in Russian).

[10]. Menter F.R., Kuntz M., Langtry R. Ten years of industrial experience with the SST turbulence model. In Proc. of the 4th international symposium on turbulence, heat and mass transfer, 2003, 8 p.

[11]. Gizzatullina A., Koroleva M. et al. Numerical investigation of cooling down and aerodynamic resistance processes in ribbed tubular elements. In Proc. of the 2020 Ivannikov Ispras Open Conference, 2020, pp. 142-149.

[12]. Байметова Е.С., Гиззатуллина А.Ф., Пушкарев Ф.Н. Решение задачи сопряженного теплообмена в оребренной трубке с использованием OpenFoam. Химическая физика и мезоскопия, том 23, no. 2, 2021 г., стр. 154-164 / Baimetova E.S., Gizzatullina A.F., Pushkarev F.N. Solving the conjugate heat transfer problem in the ribbed tube with OpenFoam. Chemical physics and mesoscopy, vol. 23, no. 2, 2021, pp. 154-164 (in Russian).

[13]. Gizzatullina A.A., Mishchenkova O.V., Pushkarev F.N. Applying the scalability apparatus to estimate the thermal efficiency of a single finned tube. Journal of Physics: Conference Series, vol. 2057, 2021, article no. 012014.

[14]. Koroleva M.R., Saburova E.A., Chernova A.A. Studying the efficiency of cooling and resistance of ribbed tubular elements. Journal of Physics: Conference Series, vol. 1675, 2020, article no. 012009.

[15]. Жукаускас А.А. Конвективный перенос в теплообменниках. М., Наука, 1982 г., 472 стр. / Zhukauskas A.A. Convective transfer in heat exchange devices. Moscow, Science, 1982, 472 p.

[16]. Бендерский Б.Я. Техническая термодинамика и теплопередача: Курс лекций. М.-Ижевск, Институт компьютерных исследований, 2002 г., 264 стр. / Benderskij B.Ya. Technical Thermodynamics and Heat Transfer: Course of Lectures. M.-Izhevsk, Institute of Computer Science, 2002, 264 p.

Информация об авторах / Information about authors

Елена Сергеевна БАЙМЕТОВА – старший преподаватель кафедры Тепловые двигатели и установки. Сфера научных интересов: вычислительная гидрогазодинамика и интенсификация процессов теплообмена.

Elena Sergeevna BAIMETOVA is a Senior Lecturer at the Thermal Engines and Installations Department. Research interests: computational fluid dynamics and intensification of heat exchange processes

Альбина Фирдавесовна ГИЗЗАТУЛЛИНА – аспирант кафедры Тепловые двигатели государственного технического университета имени М.Т. Калашникова. Сфера научных интересов: математические методы решения задач сопряженного теплообмена.

Albina Firdavesovna GIZZATULLINA, postgraduate student of the Thermal Engines and Installations Department. Research interests: mathematical methods of solving problems of coupled heat exchange.

Мария Равилевна КОРОЛЕВА – кандидат физико-математических наук, доцент, старший научный сотрудник. Сфера научных интересов: математическое моделирование, разностные схемы, газовая динамика, гидродинамика, кавитация, теплообмен.

Maria Ravilevna KOROLEVA – Candidate of Physical and Mathematical sciences, Associate Professor, Senior Researcher. Research interests: mathematical modeling, difference schemes, gas dynamics, hydrodynamics, cavitation, heat exchange.

Ольга Владимировна МИЩЕНКОВА – кандидат физико-математических наук, доцент кафедры Тепловые двигатели и установки. Сфера научных интересов: математическое моделирование, разностные схемы.

Olga Vladimirovna MISHCHENKOVA – Candidate of Physical and Mathematical Sciences, Associate Professor. Research interests: mathematical modeling, difference schemes.

Федор Николаевич ПУШКАРЕВ – аспирант кафедры Тепловые двигатели и установки. Сфера научных интересов: математическое моделирование, нейрокомпьютерные технологии, цифровая обработка данных.

Fyodor Nikolaevich PUSHKAREV is a postgraduate student at the Thermal Engines and Installations Department. Research interests: mathematical modeling, neurocomputer technologies, digital data processing.

Алена Алексеевна ЧЕРНОВА – кандидат технических наук, доцент кафедры Тепловые двигатели и установки. Сфера научных интересов: газодинамика, гидродинамика, конвективный теплообмен, теплопередача, математическое моделирование.

Alena Alekseevna CHERNOVA – Candidate of Technical Sciences, Associate Professor, Department of Thermal Engines and Installations. Research interests: gas dynamics, hydrodynamics, convective heat exchange, heat transfer, mathematical modeling.