ТРУДЫ

ИНСТИТУТА СИСТЕМНОГО ПРОГРАММИРОВАНИЯ РАН

PROCEEDINGS OF THE INSTITUTE FOR SYSTEM PROGRAMMING OF THE RAS

ISSN Print 2079-8156
Том 33 Выпуск 1

ISSN Online 2220-6426
Volume 33 Issue 1

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

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/

Содержание

Table of Contents

DOI: 10.15514/ISPRAS–2021–33(1)–1

Поиск уязвимостей небезопасного использования помеченных данных в статическом анализаторе Svace

[1] *А.Е. Бородин, ORCID: 0000-0003-3183-9821 <alexey.borodin@ispras.ru>*
[1,2] *А.В. Горемыкин, ORCID: 0000-0002-9054-0152 <alexey.goremykin@ispras.ru>*
[1] *С.П. Вартанов, ORCID: 0000-0002-8942-4592 <svartanov@ispras.ru>*
[1, 2] *А.А. Белеванцев, ORCID: 0000-0003-2817-0397 <abel@ispras.ru>*

[1] *Институт системного программирования им. В.П. Иванникова РАН, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Московский государственный университет имени М.В. Ломоносова, 119991, Россия, Москва, Ленинские горы, д. 1*

Аннотация. В статье рассматривается поиск ошибок помеченных данных в исходном коде программ, т.е. ошибок, вызванных небезопасным использованием данных, полученных из внешних источников, которые потенциально могут быть изменены злоумышленником. В качестве основы использовался межпроцедурный статический анализатор Svace. Анализатор осуществляет как поиск дефектов в программе, так и поиск подозрительных мест, в которых логика программы может быть нарушена. Целью является найти как можно больше ошибок при приемлемой скорости и низком уровне ложных срабатываний (< 20–35 %). Для поиска ошибок Svace с помощью компилятора строит низкоуровневое типизированное промежуточное представление, которое подаётся на вход основному анализатору SvEng. Анализатор строит граф вызовов, после чего выполняет анализ на основе резюме. При таком анализе функции обходятся в соответствии с графом вызовов, начиная с листьев. После анализа функции создаётся её резюме, которое затем будет использовано для анализа инструкций вызова. Анализ имеет как высокую скорость, так и хорошую масштабируемость. Внутрипроцедурный анализ основан на символьном выполнении с объединением состояний в точках слияния путей. Для отсеивания несуществующих путей для некоторых детекторов может использоваться SMT-решатель. При этом SMT-решатель вызывается, только если есть подозрение на ошибку. Анализатор был расширен возможностью поиска дефектов, связанных с помеченными данными. Детекторы реализованы в виде плагинов по схеме источник-приёмник. В качестве источников используются вызовы библиотечных функций, получающих данные извне программы, а также аргументы функции main. Приёмниками являются обращение к массивам, использование переменных как шага или границы цикла, вызов функций, требующих проверенных аргументов. Реализованы детекторы, покрывающие большинство возможных типов уязвимостей, для непроверенных целых чисел и строк. Для оценки покрытия использовался проект Juliet. Уровень пропусков составил от 46.31% до 81.17% при незначительном количестве ложных срабатываний.

Ключевые слова: статический анализ; символьное выполнение; анализ помеченных данных; Svace; поиск ошибок; уязвимости

Для цитирования: Бородин А. Е., Горемыкин А. В., Вартанов С. П., Белеванцев А.А. Поиск уязвимостей небезопасного использования помеченных данных в статическом анализаторе Svace. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 7-32. DOI: 10.15514/ISPRAS–2021–33(1)–1

Благодарности. Работа выполнена при поддержке Российского фонда фундаментальных исследований, проект №20-01-00581 А.

Searching for tainted vulnerabilities in static analysis tool Svace

[1] *A.E. Borodin, ORCID: 0000-0003-3183-9821 <alexey.borodin@ispras.ru>*
[1,2] *A.V. Goremykin, ORCID: 0000-0002-9054-0152 <alexey.goremykin@ispras.ru>*
[1] *S.P. Vartanov, ORCID: 0000-0002-8942-4592 <svartanov@ispras.ru>*
[1,2] *Belevantsev A.A., ORCID: 0000-0003-2817-0397 <abel@ispras.ru>*

[1] *Ivannikov Institute for System Programming of the RAS,*
25, Alexander Solzhenitsyn Str., Moscow, 109004, Russia
[2] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation

Abstract. The paper is dedicated to search for taint-based errors in the source code of programs, i.e. errors caused by unsafe use of data obtained from external sources, which could potentially be modified by an attacker. The interprocedural static analyzer Svace was used as a basis. The analyzer searches both for defects in the program and searches for suspicious places where the logic of the program may be violated. The goal is to find as many errors as possible at an acceptable speed and a low level of false positives (< 20-35%). To find errors, Svace with help of modified compiler builds a low-level typed intermediate representation, which is used as an input to the main SvEng analyzer. The analyzer builds a call graph and then performs summary-based analysis. In this analysis, the functions are traversed according to the call graph starting from the leaves. After analyzing the function, its summary is created, which will then be used to analyze the call instructions. The analysis has both high speed and good scalability. Intra-procedural analysis is based on symbolic execution with the union of states at merge points of paths. An SMT solver can be used to filter out infeasible paths for some checkers. In this case, the SMT-solver is called only if there is a suspicion of an error. The analyzer has been expanded to find defects of tainted data using. The checkers are implemented as plugins by using the source-sink scheme. The sources are calls of library functions that receive data from outside the program, as well as the arguments of the `main` function. Sinks are accessing to arrays, using variables as a step or loop boundary, calling functions that require checked arguments. Checkers covering most of the possible types of vulnerabilities for tainted integers and strings have been implemented. The Juliet project was used to assess the coverage. The false negative rate ranged from 46,31% to 81.17% with a small number of false positives.

Keywords: static analysis; symbolic execution; taint analysis; Svace; search for defects; vulnerabilities

For citation: Borodin A.E., Goremykin A.V., Vartanov S.P., Belevantsev A.A. Searching for tainted vulnerabilities in static analysis tool Svace. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 7-32 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–1.

Acknowledgments. This work was supported by the Russian Foundation for Basic Research, project №20-01-00581 A.

1. Введение

В статье описывается реализация поиска ошибок помеченных данных с помощью статического анализатора Svace [1-4].

К наиболее важным особенностям анализатора Svace можно отнести следующее.

- Межпроцедурный анализ на основе резюме, при котором функции обходятся по графу вызовов, начиная с листьев. Каждая функция анализируется только один раз.
- Неконсервативный анализ. За счёт отказа от консервативности анализ получает более высокую точность и производительность.
- Анализ внутри одной функции основан на анализе значений. Анализ отслеживает значения переменных и ячеек памяти и ассоциирует большинство свойств со значениями переменных.

В разд. 2 описывается, какие виды ошибок мы будем искать, в разд. 3 и 4 описывается устройство статического анализатора Svace и модуля SvEng соответственно. Разд. 5 посвящен реализации анализа помеченных данных, и разд. 6 содержит оценку результатов для проектов Juliet и Tizen 6.

2. *Помеченные данные*

В статье рассматриваются ошибки небезопасного использования данных, не контролируемых программой, т.е. полученных из внешних источников. Такие данные могут быть изменены злоумышленником. Если эти данные используются без соответствующей проверки, то в программе присутствует уязвимость.

Неконтролируемыми данными являются данные из файлов, пользовательский ввод, данные, переданные по сети. Мы рассматриваем два вида помеченных данных: помеченные целые числа и помеченные строки.

Ниже перечислены виды уязвимостей, связанных с помеченными данными.

- При использовании для доступа к массиву происходит переполнение буфера, что может позволить злоумышленнику захватить контроль над устройством [5]. По данным национальной базы уязвимостей США (NVD) ошибки подобного рода являются причиной 9,49% всех уязвимостей, занесённых в базу CWE в 2018 году [6].
- Использование в циклах. Если помеченные данные используются в качестве ограничения цикла, то цикл может выполниться большее количество раз, чем предусмотрено. Это может приводить как к излишнему расходованию процессорного времени, так и к другим ошибкам, например, переполнению массива. Если помеченные данные используются в качестве шага итератора цикла, то можно так подобрать данные, чтобы цикл стал бесконечным.
- Использование для некоторых операций, например, для выделения памяти. Злоумышленник может заставить программу выделить излишне много памяти.

Листинг 1 иллюстрирует уязвимости, связанные с целыми числами. Для исправления уязвимости переполнения буфера, необходимо проверить диапазон переменной n, чтобы он принадлежал интервалу [0; 99]. Безопасный диапазон для других видов уязвимостей зависит от логики программы.

```
char buf[100];

int n;
scanf("%d", &n); //n - tainted

//переполнение буфера, если n меньше нуля либо больше 99
buf[n] = 0;

//выделение памяти
char*p = (char*)malloc(n * sizeof(struct Fmt));
int i = 99;

while(i > 0) {
  buf[i] = '0' + (i % 10);
  //бесконечный цикл, если n равно 0
  i -= n;
}
```

Листинг 1. Помеченные целые числа
Listing 1. Tainted integers

Помеченные строки могут как содержать произвольные символы, так и иметь произвольную длину. При копировании такой строки в массив фиксированного размера, может происходить его переполнение.

Фрагмент кода на листинге 2 иллюстрирует уязвимости с помеченными строками.

```
char*p = getenv("aaa");

//потенциальное переполнение буфера
//размер p может быть меньше 10
char x = p[10];

char buf[10];
int n = *((int*)p);
//переполнение буфера
buf[n] = 0;
```

Листинг 1. Помеченные строки
Listing 1. Tainted strings

3. Статический анализатор Svace

Анализатор Svace осуществляет поиск дефектов в исходном коде; при этом поиск происходит как для ошибок, возможных при выполнении программы, так и подозрительных мест, где, возможно, логика программы нарушена. Целью является найти как можно больше дефектов при приемлемом времени работы и высоком качестве результатов.

Инструмент может как пропускать реальные дефекты, так и выдавать ложные срабатывания, не соответствующие реальным дефектам. Для анализируемой программы не требуется никакой подготовки, достаточно, чтобы исходный код компилировался. Время анализа сравнимо со временем компиляции программы. Для больших программ целесообразно использовать Svace во время ночной сборки.

Задача поиска дефектов является алгоритмически неразрешимой [7], т.е. нельзя найти все ошибки определённого типа в произвольной программе и не выдать ложных срабатываний. Для решения этой задачи используются всевозможные компромиссы. Одним из подходов является поиск приближенного решения. Возможна два вида приближений.

- Приближать в сторону отсутствия ложных срабатываний. В этом случае выдаются только истинные сообщения, но возможен пропуск множества дефектов. Типичным примером является выдача ошибок компилятора, для которого недопустима возможность отказаться компилировать корректную программу.
- Поиск всех ошибок определённого типа. Процент ложных срабатываний при этом может быть высоким. Более качественный анализ может достигаться за счёт существенного увеличения времени работы.

Такие приближения называются консервативными, так как они всегда округляют решение в одну сторону. В Svace не используется консервативное приближение. Это позволяет как ускорить анализ программ, так и существенно повысить уровень истинных срабатываний.

В статье [8] утверждается, что консервативный анализ алиасов необходим для оптимизации программ, но является опциональным для анализаторов. Когда авторы потоково- и контекстно-чувствительного анализа [9-10] удалили один шаг, необходимый для обеспечения консервативности, общее время анализа значительно сократилось. В одном случае время уменьшилось от нескольких дней до нескольких минут.

На рис. 1 иллюстрируются возможные подходы. По оси ординат показан процент найденных ошибок, по оси абсцисс процент истинных. Подход, используемый в Svace, позволяет максимизировать площадь прямоугольника, показанного пунктирными линиями.

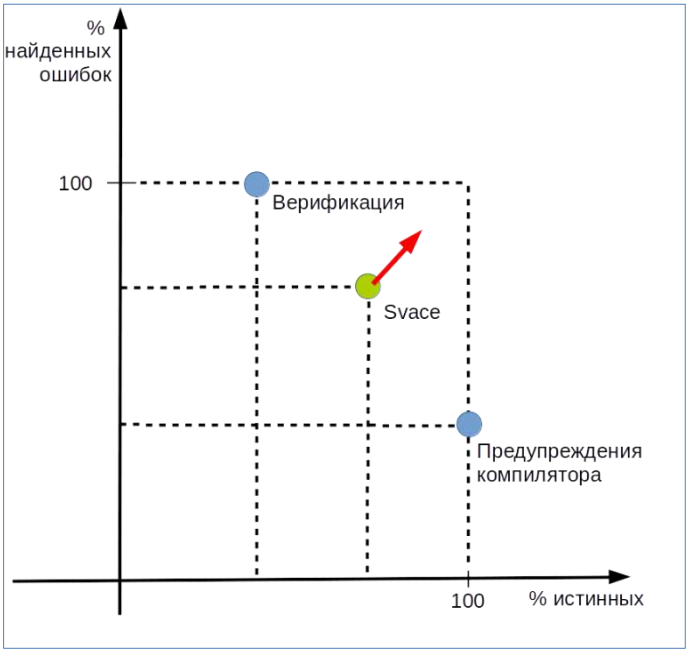


Рис. 1. Подходы к обнаружению ошибок
Fig. 1. Approaches to defect detection

Подобный подход достаточно популярный и не является ноу-хау, используемым в Svace. В статье [11] предлагается термин *soundy*, означающий в целом консервативный анализ, который допускает отказ от консервативности для некоторых конструкций.

2.1 Архитектура Svace

Для анализа программы анализатору требуется исходный код и скрипт сборки. Svace осуществляет поиск дефектов в программах на языках C, C++, Java, Kotlin, Go[1].

Типичная схема анализа показана на рис. 2. Svace перехватывает команды запуска компилятора и компоновки [12]. Затем запускается модифицированный компилятор, который строит абстрактное синтаксическое дерево (АСД) и запускаются детекторы для поиска ошибок на АСД, а также генерируется промежуточное представление программы для последующего анализа. Промежуточное представление подаётся на вход анализатору SvEng[2], выполняющему межпроцедурный анализ.

Анализаторы, построенные на базе АСД, осуществляют проход по узлам АСД и делают относительно простые проверки анализируемых правил. С помощью АСД-анализаторов можно найти подозрительные паттерны на деревьях разборов, различные опечатки, но при этом

- сложно отследить зависимости между переменными,
- сложно выполнить анализ указателей,
- сложно сделать межпроцедурный и межмодульный анализ.

[1] Первый релиз Svace, поддерживающий язык Go, ожидается в январе 2021 г.
[2] Сокращение от **Sv**ace **Eng**ine – движок анализатора Svace.

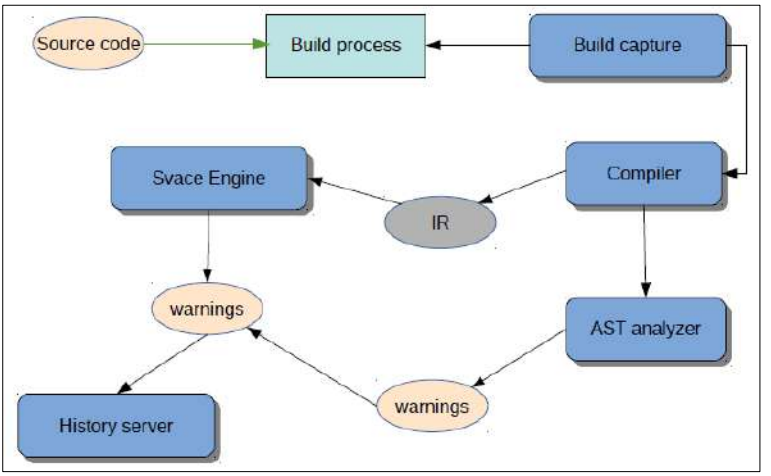


Рис. 2. Схема анализа
Fig. 2. Analysis scheme

Поскольку поиск ошибок помеченных данных, как правило, требует анализа вышеперечисленных свойств, мы не будем реализовывать детекторы помеченных данных на этом уровне.

4. Анализатор SvEng

4.1 Общая схема

Анализатор SvEng осуществляет глубокий межпроцедурный потоково- и контекстно-чувствительный анализ. Наиболее важными являются следующие особенности анализатора SvEng.

- Межпроцедурный анализ на основе резюме, при котором функции обходятся по графу вызовов начиная с листьев. Каждая функция анализируется только один раз. По завершению анализа функции создаётся её резюме, которое затем будет использовано при анализе вызова функции. Резюме содержит описание свойств функции. Анализ сохраняет баланс между компактностью анализа и точностью описания семантики функции.
- Неконсервативный анализ. За счёт отказа от консервативности анализ получает лучшую точность и более высокую производительность.
- Анализ внутри одной функции основан на анализе значений. Анализ отслеживает значение переменных и ячеек памяти и ассоциирует большинство свойств со значениями переменных.
- Запуск всех анализов одновременно. Невысокая стоимость одного детектора.

Дизайн SvEng позволяет находить множество типов дефектов: разыменование нулевых указателей, недостижимый код, переполнение массива, утечки памяти и ресурсов, неправильное использование библиотечных функций, двойные блокировки мьютексов.

Мы реализовали поиск уязвимостей помеченных данных таким образом, чтобы по максимуму использовать возможности анализатора. Детекторы реализованы в виде анализа источник-приёмник, где источником являются функции, возвращающие помеченные данные, а приёмником – операции, в которых эти данные должны быть проверены перед использованием. Анализатор хорошо обнаруживает ошибки, где источник и приёмник не сильно удалены друг от друга на графе вызовов.

Анализ производится по следующем схеме:

1) на вход анализатору подаются файлы с промежуточным представлением;
2) строится граф вызовов;
3) запускается предварительная фаза анализа, на которой доступны легковесные анализы; на этой фазе, в том числе, собирается информация об указателях на функцию и виртуальных вызовах;
4) достраивается граф вызовов для виртуальных функций и вызовов по указателю;
5) запускается основная фаза анализа.

4.2 Промежуточное представление

Для анализа используется собственное промежуточное представление, которое строится уже после запуска анализатора (Svace IR). На вход анализатору подаются файлы в промежуточном представлении, зависящие от анализируемого языка:

- LLVM-файлы для C/C++,
- Java-байт код для Java/Kotlin,
- собственный JSON-формат для Go.

После конвертации исходного кода в представление Svace IR, анализ производится одинаково для всех языков программирования.

Svace IR является низкоуровневым типизированным языком в SSA-форме. Язык имеет процедуры, в том числе возможность их вызова по указателю. Для моделирования виртуальных вызовов в C++ явно строятся виртуальные таблицы. Для моделирования исключений используются конструкции goto.

Листинг 3 содержит пример кода на C. Соответствующий фрагмент на Svace IR приведён на листинге 4.

```
int calc(int f) {
    int a, b;
    int*p;
    a = 1;
    b = 2;
    p = f ? &a : &b;

    int x = *p;
    *p = 3;

    int y = *p;

    return x + y;
}
```

Листинг 3. Код на C
Listing 3. C code

Использование низкоуровневого промежуточного представления имеет как свои плюсы, так и минусы. Недостатки такого представления:

- затруднён анализ высокоуровневых конструкций;
- нет информации об оригинальных синтаксических конструкциях;
- трансформация представления в эквивалентное компилятором.

Основным преимуществом является простота анализа. Семантика точно моделируется компилятором, язык имеет небольшое количество инструкций, которые редко меняются. Значительное количество языковых конструкций являются синтаксическим сахаром

(исключения, конструкторы и деструкторы, циклы и др.) и не требуют добавления новых инструкций в IR.

```
def calc(int f) int {
    a = alloca();
    b = alloca();
    p = alloca ref int();
    store 1, a;
    store 2, b;

    if(f != 0)
        goto true-label;
    else
        goto false-label;

true-label:
    store a, p;
    goto label2;

false-label:
    store b, p;
    goto label2;

label2:
    t1 = load p;
    x = load t1;
    store 3, t1;
    t2 = load p;
    y = load t2;
    t3 = x + y;
    ret t3;
}
```

Листинг 4. Код на Svace IR
Listing 4. Svace IR code

Низкоуровневое IR для задачи поиска помеченных данных представляется хорошим выбором, т.к. для этого вида дефектов важна принципиальная возможность эксплуатации уязвимости, которая не меняется при переходе на низкоуровневый язык. Анализ синтаксических конструкций при этом не очень важен.

4.3 Межпроцедурный анализ

Используется межпроцедурный анализ на основе резюме.

При таком анализе для каждой функции строится резюме – краткое описание поведения функции. Резюме используется для анализа инструкции вызова функции и позволяет избежать повторного анализа тела функции. Резюме создаётся после анализа функции.

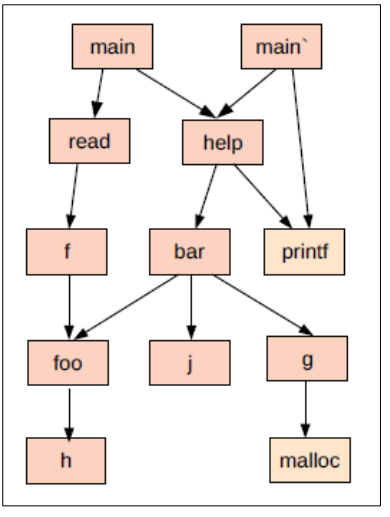


Рис. 3. Пример графа вызовов
Fig. 3. An example of a call graph

На рис. 3 показан граф вызовов для сборки двух программ. Наверху графа вызовов находятся функции `main`, которые вызывают другие функции. Листьями графа являются функции `h` и `j`, анализ начинается с листьев графа. После того как они будут проанализированы, станут доступны их резюме и возможность начать анализ функции `foo`.

Анализ имеет как высокую скорость, так и хорошую масштабируемость. Последняя достигается за счёт того, что резюме создаётся достаточно компактным и не включает все детали поведения функции. Возможно ограничивать размер резюме. В этом случае резюме не будет описывать все интересующие эффекты вызова функции, но анализ вызова функции будет иметь константное время.

Благодаря своим преимуществам, этот анализ является довольно популярным и используется во многих инструментах статического анализа: Prefix [13], Saturn [14], Calysto [15], CSharpChecker [16].

4.4 Предварительная фаза

При использовании только анализа на основе резюме каждая функция посещается ограниченное количество раз. При этом для ряда анализов необходима некоторая информация обо всей программе, либо о функциях, находящихся выше по графу вызовов.

Для получения такой информации была создана предварительная фаза, которая включает следующие виды анализов:

- анализ присваиваний значений указателям на функции,
- анализ заполнения виртуальных таблиц C++,
- анализ иерархии классов для Java,
- анализ значений глобальных переменных.

Эти анализы на вход получают информацию о глобальных переменных и инструкции для каждой функции. Анализ производится параллельно для разных модулей.

Анализ не учитывает порядок инструкций, но доминирующая инструкция всегда анализируется первой. Потоково-нечувствительный анализ был выбран, чтобы не слишком замедлять время общего анализа, т.к. основные свойства будут проанализированы во время основного анализа.

Цель данных анализов – получить необходимую информацию, не затрачивая существенное количество времени.

4.5 Девиртуализация

Цель девиртуализации разрешить вызов процедур для указателей на функции и виртуальных таблиц. Таблица виртуальных функций представляет собой глобальную переменную типа структуры с полем – массивом, содержащим указатели на виртуальные функции, доступ к которым осуществляется через константные значения.

Анализ отслеживает значения переменных, структур и константные индексы массивов.

Для каждого указателя собирается все возможные записи. Результатом анализа для каждой пары «функция – указатель» является множество функций, которые могут быть вызваны, либо пустое множество, если анализ не смог определить всех кандидатов.

Результаты анализа используются двумя способами:

- при достраивании графа вызовов для добавления рёбер,
- при обработке инструкции вызова по указателю.

Анализу на основе резюме нужна информация о графе вызовов. Поэтому после получения информации о виртуальных функциях достраивается граф вызовов. Эта процедура не является сложной, достаточно просто добавить ребро, где **может** быть вызвана функция в результате виртуального вызова. При анализе функции на основном анализе эту информацию

в некоторых случаях можно будет уточнить. Более точный анализ на предварительной фазе добавит меньшее количество лишних рёбер. До тех пор, пока основной анализ в состоянии убрать лишнее ребро, они не влияют на точность анализа. Но время анализа может увеличиваться, а потребление памяти возрастать, т.к. каждое ребро в графе вызовов накладывает ограничения на возможный обход функций и требует наличия резюме.

В основном анализаторе реализован плагин, отслеживающий возможных кандидатов для каждого указателя. Этот плагин получает данные о кандидатах для каждого указателя, а затем используя потоково-чувствительный анализ распространяет эти данные внутри процедуры независимо. В некоторых случаях, используя информацию о значениях переменных и недостижимых инструкциях, анализ может получить более точную информацию, чем та, которая доступна после предварительной фазы.

Для случая, если указателю соответствуют несколько кандидатов, моделируется их условный вызов. Для каждого кандидата применяется резюме, а затем происходит объединение для каждого контекста использования. Таким образом каждое резюме применяется отдельно для каждого контекста.

4.6 Внутрипроцедурный анализ

Для анализа отдельной функции используется символьное выполнение с объединением состояний анализа в точках слияния путей. Для функции строится граф потока управления, и анализ начинает обход графа. С рёбрами графа ассоциируются абстрактные состояния. На каждом шаге для инструкции по абстрактному состоянию на входных рёбрах формируется абстрактное состояние на выходном ребре.

Для сильно связанных компонент (ССК) выполняются несколько итераций. Абстрактные состояния для выходных рёбер ССК сохраняются после каждой итерации. После завершения анализа ССК, эти состояния объединяются. Во время анализа используются несколько эвристик для моделирования всевозможных путей выполнения ССК. В некоторых случаях моделирование происходит некорректно, если эвристики не сработали.

Все дополнительные анализы и детекторы запускаются одновременно, что позволяет уменьшить одновременно и время анализа, и потребляемую память. Время анализа сокращается за счёт того, что общие для всех детекторов свойства анализируются только один раз. А потребление памяти уменьшается, т.к. в большинстве случаев после анализа инструкции, состояние анализа на входном ребре может быть удалено.

Для моделирования значений переменных и ячеек памяти Svace использует абстракцию, названную идентификатором значений. Двум переменным сопоставляется один идентификатор значений, если при выполнении эти переменные будут иметь одинаковые значения (решается задача нумерации значений).

Большинство свойств ассоциируются с идентификаторами значений. Сами свойства также можно описывать с помощью идентификаторов значений. Для описания свойств используются атрибуты. Атрибут обозначает некоторое анализируемое свойство.

Примеры атрибутов:

- Null – значение указателя имеет нулевое значение,
- ValueInterval – значения целочисленной переменной принадлежат отрезку [a;b],
- PointsTo – указатель указывает на ячейки памяти из множества. Сами ячейки памяти обозначаются идентификатором значений, который моделирует адреса ячеек памяти,
- Ness – необходимые условия достижения ребра в графе потоке управления. Представляет собой формулу булевой логики, где в качестве переменных используются идентификаторы значений.

4.7 Использование SMT-решателя

Для реализации чувствительности к путям в Svace используются условные атрибуты, которые выражают свои свойства в виде формул булевой логики над идентификаторами значений. Условные атрибуты, описывающие свойства значений переменных, ассоциируются с идентификаторами значений.

Формулы имеют вид, показанный на рис. 4, где *Val* – идентификаторы значений, *Const* – константы. Для каждой литеральной формулы *Atom* определена инструкция отрицания, возвращающая другую литеральную формулу. В формулах могут использоваться конъюнкции, дизъюнкции от других формул, а также отрицание для литеральных формул.

$$\bowtie ::= > \mid < \mid = \mid \neq \mid \leq \mid \geq$$
$$\oplus ::= + \mid - \mid * \mid /$$
$$\mathbf{Op} ::= \mathbf{Val} \mid \mathbf{Const}$$
$$\mathbf{Atom} ::= \mathbf{True} \mid \mathbf{False} \mid \mathbf{Op} \bowtie \mathbf{Op} \mid \mathbf{Op} = \mathbf{Op} \oplus \mathbf{Op}$$
$$\mathbf{Conj} = \mathbf{Atom} \mid \mathbf{Conj} \wedge \mathbf{Conj}$$
$$\mathbf{CFormula} ::= \mathbf{Conj}$$
$$\mathbf{SFormula} ::= \mathbf{Conj} \wedge \overline{\mathbf{Conj}}$$
$$\mathbf{FFormula} ::= \mathbf{Atom} \mid \mathbf{FFormula} \wedge \mathbf{FFormula} \mid \mathbf{FFormula} \vee \mathbf{FFormula}$$

Рис. 4. Используемые формулы
Fig. 4. Used formulas

SMT-решатель запускается только перед выдачей предупреждения для того, чтобы отсеять несуществующие пути. Таким образом SMT-решатель запускается не больше, чем количество неотфильтрованных предупреждений. В общем случае используется следующая формула:

$$Ness, \wedge\ Error_i, (v),$$

где $Ness$ необходимое условие достижимости для ребра i, $Error_i, (v)$ – условие ошибки для значения, моделируемого идентификатором v.

4.8 Анализ потока данных

Для анализа недостижимого кода был реализован консервативный анализ потока данных (АПД[3]) [17]. Этот анализ запускается перед началом анализа каждой функции в основном анализе. Анализ помечает недостижимые рёбра на графе потока управления. Основная причина, по которой он был реализован – недостаточная точность основного анализа. Недостижимое ребро сильно влияет на все остальные анализы, поэтому неконсервативность в данном случае может приводить к значительно худшим результатам.

Позже были добавлены другие анализы:

- анализ интервалов,
- анализ исключений,
- анализ живых переменных.

Помимо получения консервативных результатов, этот анализ позволяет получить данные о функции перед запуском основного анализа. Поскольку в основном анализе все детекторы запускаются одновременно, возникает проблема предварительного получения свойств, которая решается АПД.

4.9 Анализ сверху вниз на предварительной фазе

Рассмотрим пример, показанный на листинге 5. Функция `f` получает помеченные данные и передаёт их в функцию `g`. Если функция `g` небезопасно использует свой аргумент, то надо

[3] Под АПД в статье мы будем подразумевать движок анализа, основанный на анализе потока данных.

выдать предупреждение. Для этого в анализе на основе резюме, надо протащить информацию о небезопасном использовании аргумента `y` через резюме.

В общем случае, сделать это не тривиально. При анализе функции `g` ничего не известно о контекстах, где она будет использована. Поэтому не известно надо ли сохранять информацию о небезопасном использовании. Резюме является компактным, сохранение всей возможной информации лишит анализ основных преимуществ.

```
void f() {
    int x = input();
    g(x);
}

void g(int y) {
    // Известен контекст, в котором переменной y
    // было передано помеченное значение
}
```

Листинг 5. Мотивация для предварительной фазы
Listing 5. Motivation for preliminary phase

Для решения описанной проблемы мы добавили анализ на предварительной фазе на основе АПД. На этой фазе процедуры обходятся в произвольном порядке; в рамках контекста процедуры отслеживаются помеченные данные и для каждого вызова процедуры в этом контексте запоминаются аргументы, которые являются помеченными. Таким образом, сохранённая информация относится только к рассмотренным контекстам.

Идея этого подхода в некотором смысле схожа с динамическим анализом, при котором происходит исследование свойств конкретного пути программы и все выводы которого остаются справедливыми только в рамках этого пути. При таком анализе производится исследование свойств, которые не обладают всеобщностью и справедливы для некоторого контекста вызова или цепочки вызовов функций.

Таким образом, предварительная фаза собирает информацию о контекстах использования функции. Данные о помеченных аргументах функции используются во время основного анализа функции. Соответствующие атрибуты, описывающие помеченные данные, устанавливаются на основе результатов предварительной фазы. Для конкретных детекторов ситуация выглядит так, как будто аргумент является результатом вызова функции, возвращающей помеченные данные. При этом возникает проблема, что информация о помеченных данных может попасть в резюме, что не является корректным. Резюме описывает результат вызова функции для произвольного контекста вызова, а предварительный анализ собирает данные для некоторых контекстов. Для решения этой проблемы был выбран следующий способ. Функция, у которой некоторые аргументы являются помеченными согласно предварительному проходу, анализируется два раза:

1) используются данные от предварительного прохода, резюме не создаётся,
2) обычный анализ без использования данных от предварительного прохода с созданием резюме.

Например, если для функции `input` известно, что она всегда является источником помеченных данных, то при анализе функции `f`, можно сделать вывод о существовании контекста вызова функции `g`, в котором её аргумент имеет помеченное значение. На основании этого, верным будет сообщить об ошибке. Основная фаза принимает решение на основе только анализа функции `g`.

4.10 Спецификации в Svace

Для анализа библиотечных функций, поведение которых известно, используются спецификации. Спецификация в Svace это ещё одно определение функции, написанное на анализируемом языке. Спецификация описывает поведение функции в компактной форме. Помимо конструкций языка, спецификации могу содержать вызовы специальных предопределённых функций, называемых спец-функциями.

Дистирбутив Svace содержит спецификации для широкоиспользуемых библиотек. Пользователь может добавлять свои собственные спецификации.

Для примера разберём спецификацию для функции `strcat` из стандартной библиотеки C. Исходный код спецификации:

```
char *strcat(char *s, const char *append) {
    char d1 = *s;
    char d2 = *append;
    sf_set_trusted_sink_ptr(s);
    sf_set_trusted_sink_ptr(append);
    sf_append_string(s, append);
    sf_vulnerable_fun("This function is unsafe, use strncat instead.");
    sf_null_terminated(s);
    return s;
}
```

В данном коде содержатся следующие спецфункции:

- `void sf_set_trusted_sink_ptr(const void* str)` – данная спецфункция показывает, что ее аргумент `str` должен быть из надежного источника, иначе данный указатель может вызвать уязвимость,
- `void sf_vulnerable_fun(const char*const reason)` – данная спецфункция показывает, что текущая функция небезопасна и имеет safe-аналоги,
- `void sf_append_string(char* dst, const char* src)` – данная спецфункция показывает, что выполнено добавление строки `src` к `dst`,
- `void sf_null_terminated(char *p)` – данная спецфункция показывает, что строка `p` заканчивается нулевым символом.

При анализе данной спецификации детекторы могут извлечь следующие свойства:

строки `s, append` были разыменованы, это может быть полезно для детекторов, которые ищут разыменование нулевых указателей,

строки `s, append` должны быть из надежного источника, информация используется детекторами помеченных данных,

функция `strcat` опасна и нужно использовать ее безопасную замену `strncat`,

строка `s` заканчивается нулевым символом.

5. Детекторы помеченных данных в SvEng

5.1 Плагины и детекторы

Анализ в SvEng разделён на ядро и плагины. Ядро анализа отслеживает граф указателей, выполняет сильные и слабые обновления ячеек памяти и вызывает обработчики для соответствующих ситуаций. Все дополнительные анализы реализуются внутри плагинов в виде обработчиков инструкций, оперирующих не переменными, а соответствующими идентификаторами значений. Дополнительные анализы получают на вход абстрактное состояние на входном ребре инструкции, и формируют абстрактное состояние на выходном

ребре. Детекторы также реализуются в плагинах и выдают предупреждения на основе входного абстрактного состояния и передаточной функции обрабатываемой инструкции.

Всем дополнительным анализам доступна информация на входных рёбрах и недоступна на выходных. Различные анализы и детекторы не должны влиять друг на друга во время анализа инструкции. В текущей версии анализаторы и детекторы запускаются по очереди[4], но результаты должны быть такими же, как если бы они запускались параллельно.

В абстрактном состоянии для описания свойств используются атрибуты. Атрибуты могут ассоциироваться с идентификаторами значений и с рёбрами графа потока управления для некоторого абстрактного состояния. Атрибут обозначает некоторое анализируемое свойство. Атрибуты должны иметь функцию объединения двух атрибутов $\sqcup$. Эта функция используется при объединении состояний в точках слияния путей. Атрибуты позволяют разделять абстрактное состояние между дополнительными анализами.

Атрибут может иметь произвольную структуру, но некоторые виды атрибутов используются достаточно часто. Можно выделить следующими виды атрибутов:

- двоичные атрибуты,
- тернарные атрибуты,
- интервальные атрибуты,
- условные атрибуты,
- множество идентификаторов значений.

Каждый вид атрибутов может также иметь трассу – односвязный список из пар «точка в программе – краткое текстовое описание события». Трассы меняются при распространении атрибутов и используются в момент выдачи предупреждений, чтобы показать дополнительные точки в программе, позволяющие лучше понять, в чём ошибка.

Двоичные атрибуты. Имеют два значения: *true* и *false*. Значение *true* означает, что некоторая переменная точно имеет анализируемое свойство, значение *false* всё остальное, т.е. либо переменная не имеет это свойство, либо недостаточно информации. В зависимости от функции объединения эти атрибуты могут быть двух видов:

- or-атрибуты – результат будет истинным, если хотя бы один аргумент является истинным,
- and-атрибуты – результат будет истинным, если оба аргумента является истинными.

Тернарные атрибуты могут принимать следующие значения:

true – для переменной выполняется свойство в данной точке для всех путей[5], проходящих через неё,

maybe – существует путь, возможно, недостижимый, для которого свойство выполняется,

false – остальные случаи: данное свойство либо не выполняется, либо недостаточно информации.

Функция объединения атрибута:

- *true* $\sqcup$ *false* = *maybe*,
- *maybe* $\sqcup$ *false* = *maybe*,
- *true* $\sqcup$ *maybe* = *maybe*.

[4] Распараллеливание запуска детекторов потенциально позволяет немного ускорить анализ, но при этом значительно усложняет внутрипроцедурный анализ из-за необходимости синхронизации. Поэтому выбор для распараллеливания был сделан на уровне графа вызовов: отдельные функции могут анализироваться параллельно, но анализ внутри функции последовательно.

[5] Для случая статического анализа учитываются все пути, которые анализ посчитал достижимыми. Чем точнее анализ, тем больше недостижимых путей он сможет отсеять.

Фактически, тернарный атрибут является объединением or и and двоичных атрибутов. Значение *true* будет, если оба двоичных атрибута имеют истинное значение. Значение *maybe*, если or-двоичный атрибут имеет истинное значение, а and-атрибут не имеет, и *false* в остальных случаях.

Интервальные атрибуты связывают переменную с некоторым целочисленным интервалом, который описывает произвольное свойство. Например, возможный размер выделенной памяти для указателя или же значение, которое может принимать целочисленная переменная. Интервал может принимать следующие значения: $[a, b] - a \leq b;\ a, b \in [MIN\_INT + 1, MAX\_INT - 1]$, данная запись означает, что свойство у переменной имеет значение в пределах интервала от a до b. Значения MIN_INT и MAX_INT зарезервированы для обозначения бесконечностей $-\infty$ и $+\infty$.

Цепочка интервалов представляет собой несколько интервалов. Цепочка интервалов позволяет моделировать интервалы с выколотыми точками.

Условные атрибуты хранят в себе формулу, описывающую выполнение некоторого свойства (см. 4.7). Выполнимость формулы проверяется SMT-решателем перед выдачей предупреждения. Данная формула состоит из следующих конъюнкций:

- условие достижимости, данная формула содержит условия при которых точка, где выдается срабатывание, будет достижима; данную формулу хранит атрибут $Ness$.
- условия помеченных данных, данная формула содержит условие, при котором указатель или значение целочисленной переменной будет получено из ненадежного источника; данную формулу отслеживают атрибуты $TaintedPtrIf$ (для ненадежного указателя) и $TaintedIntIf$ (для ненадежного значения целочисленной переменной), которые будут описаны позже,
- некоторое дополнительное свойство, зависящее от детектора, например, для обращения к массиву, проверяется что значение индекса меньше нуля, либо больше размера массива.

5.2 Межпроцедурное распространение атрибутов

Ядро анализа при создании резюме определяет какие идентификаторы значений туда попадут и для каждого вызывает обработчик *annotate*. В момент применения резюме ядро анализа сопоставляет формальные аргументы фактическим и вызывает обработчик *apply*. Для того, чтобы сделать атрибут межпроцедурным, достаточно подписаться на эти два обработчика, и реализовать соответствующую логику в зависимости от семантики атрибутов.

Для тернарных, двоичных, интервальных атрибутов резюме формируется путем передачи свойств без изменений (обработчик *annotate*).

Условные атрибуты – прежде, чем передать формулу в резюме, она упрощается:

1) добавляем к условию конъюнкцию с условием достижимости,
2) все атомарные условия, которые содержат в себе идентификаторы, добавленные ядром в резюме, сохраняются в специальное множество,
3) берем не посещенное простое условие из формулы; если оно не содержится в списке, то преобразуем его в $False$[6], иначе помечаем условие как посещенное,
4) для получившейся формулы применяем правила поглощения ($False \wedge Cond = False$, где Cond – некоторое условие),

[6] Округление может быть как в сторону лжи, так и в сторону истины. Выбор округления зависит от семантики. Для атрибутов, которые влияют на то, что свойство считается помеченным, чтобы не выдать ложное срабатывание за счёт пропуска потенциального дефекта, округление идёт в сторону лжи. Для атрибутов, которые влияют на санитизацию, наоборот, округление будет в сторону истины. Для анализатора, имеющего цель найти больше ошибок за счёт понижения процента истинных срабатываний, округления имеет смысл производить в противоположную сторону.

5) долучившаяся формула сохраняется в резюме.

Далее будем считать, что все стандартные атрибуты являются межпроцерурными и используют описанный выше механизм создания резюме, если не сказано иного.

5.3 Используемые атрибуты

Опишем вспомогательные атрибуты, предоставляющие нужную информацию для множества детекторов, которые используют помеченные данные.

Атрибут $MustTaintedInterval$ хранит интервал возможных значений помеченной переменной. Значения из этого интервала должны быть проверены перед использованием. Функция объединения атрибута: пересечение интервалов ($[10,20] \cap \emptyset = \emptyset$, $[10,20] \cap [10,11] = [10,11]$).

Атрибут $MightTaintedInterval$ хранит максимально возможный помеченный интервал. Функция объединения атрибута: объединение с пустым интервалом ($[10,20] \sqcup \emptyset = [10,20]$) и пересечение с непустым ($[10,20] \sqcup [10,11] = [10,11]$).

Со значением помеченной переменной также связаны атрибуты, которые показывают осуществлялась ли проверка значения данной переменной:

$MinIsTainted$ – двоичный or-атрибут, указывает на наличие проверки нижней границы. По умолчанию значение *true* (проверка отсутствовала)

$MaxIsTainted$ – двоичный or-атрибут, указывает на наличие проверки верхней границы. По умолчанию значение *true* (проверка отсутствовала)

Эти атрибуты нужны, чтобы не выдавать бесполезное срабатывание для случаев, когда переменную сравнили с некоторым параметром функции:

```
char* allocate(int max) {
    unsigned int n;
    scanf("%d", n);

    if (n > max) {
       printf ("parameter too big, use %d", max);
       return 0;
    }

    return malloc(n);
}
```

Атрибут подавляет выдачу предупреждений для случаев, когда неизвестна точная безопасная граница. Например, для выделения памяти из кучи нет возможности определить такое ограничение статически. Для доступа к массиву с известным размером, такая граница известна, и атрибут не влияет на выдачу предупреждения.

5.4 Целочисленные помеченные значения

Значений целочисленных переменных могут контролироваться злоумышленником. Все реализованные детекторы являются подвидом источник-приёмник, где в качестве источников используются функции получения данных из внешних источников, а приёмников – операции, где данные необходимо проверить.

Источником являются все данные, которые получены извне программы (файл, сеть, пользовательский ввод). В большинстве случаев эти данные в Svace получаются из спецификаций. Исключением являются параметры функции `main` – `argc` и `argv`. Svace с помощью атрибута $ArgvVarAttr$ связывает `argv` с `argc`, данный атрибут позволяет детекторам избегать ложных срабатываний связанных с использованием параметров `main`. Например, когда указатель `argv` смещают, используя `argc`.

Приёмником являются:

- использование как индекса массивов; перед использованием надо проверить диапазон,
- библиотечные функции,
- использование как шага цикла либо как ограничение цикла
- использование как индекса для указателя; хотя неизвестен его точный размер, это не значит, что можно использовать любой.

Особенностью помеченных целочисленных переменных является то, что проверка их диапазона может производиться довольно сложным образом с использованием битовой арифметики.

Кроме этого, некоторые переменные могут быть связаны друг с другом какой-либо операцией, в этом случае проверяться может только одна переменная. При реализации анализа важно учитывать такие взаимосвязи. Для анализа взаимосвязей между переменными используется SMT-решатель: строятся формулы, описывающие эти взаимосвязи, а затем вызывается SMT-решатель, который определит может ли формула иметь решение.

В Svace реализованы следующие типы детекторов для поиска помеченных целых чисел:

- TAINTED_ARRAY_INDEX доступ к массиву по непроверенному индексу,
- TAINTED.INT_OVERFLOW – приёмником является потенциальное целочисленное переполнение,
- TAINTED_INT.PTR – доступ к указателю с помощью смещения,
- TAINTED_INT – доступ к функции, где входные параметры должны быть проверены,
- TAINTED_INT.LOOP – приёмником помеченных данных является использование переменной как ограничения цикла, либо как шага цикла.

Для перечисленных детекторов могут использоваться суффиксы, имеющие следующее значение:

- .MIGHT – не на всех путях к опасному использованию данных эти данные помеченные,
- .COND – приёмник находится в вызываемой функции и не достижим на всех путях внутри этой функции.

Например, TAINTED_ARRAY_INDEX.MIGHT – доступ к массиву в качестве индекса, где не все пути содержат помеченные данные.

5.4.1 Предупреждение TAINTED_INT

В коде может встретиться ситуация, когда программист использует переменную, значение которой получено из внешнего источника, в таких функции как `strncpy`, `malloc` или в качестве условия выхода из цикла. Так как злоумышленник может передать любое значение, то использование таких переменных может повлечь за собой уязвимости: бесконечный цикл, переполнение массива.

```
void test(int fd) {
    int sizeBuf;
    //sizeBuf получен из ненадежного источника
    int ret = recv(fd, &sizeBuf, sizeof(sizeBuf), 0);
    if(ret<0)
        return;

    if (sizeBuf < 0) {
        return;
    }
    //использование помеченной переменной в calloc
    char*x = calloc(1, sizeBuf);//TAINTED_INT
```

```
}
```

В данном примере размер выделенной памяти зависит от помеченной переменной, *sizeBuf* может быть очень большим, что повлечет за собой выделение чрезмерно больших объемов памяти, которая не будет использоваться или же *sizebuf* может быть равен нулю, тогда обращение к *x* может вызвать ошибку переполнения массива.

Детектор ищет ситуации, когда помеченные целочисленные переменные передаются в функции, где они могут вызвать уязвимость.

Для идентификации помеченных целочисленных переменных используется атрибут $TaintedIntIf$, который хранит в себе формулу, при выполнении которой переменная будет содержать помеченное значение. Функция объединения атрибута: конъюнкция формул с веток.

Для межпроцедурного распространения свойств об использовании данных, которые должны быть из надёжного источника, используется атрибут $TrustedIntSinkFlag$. Фактически, этот атрибут при применении резюме создаёт ещё один приёмник, для которого может быть выдано предупреждение. Заметим, что анализ спецификаций является частным случаем межпроцедурного анализа. При обработке спецификаций для функций типа *malloc* используется этот атрибут.

Условия выдачи срабатывания:

- после вызова функции ее аргумент имеет следующие свойство: T$rustedIntSinkFlag$ – *true* или же переменная используется в условиях цикла,
- переменная имеет не пустой атрибут $MustTaintedInterval$ или $MightTaintedInterval$,
- у переменной не проверяли нижнюю и верхнюю границу; атрибуты $MinIsTainted$ и $MaxIsTainted$ для переменной имеют значение *false,*
- конъюнкция формулы из $TaintedIntIf$ с условием, что переменная лежит в интервале из $MustTaintedInterval$ или $MightTaintedInterval$ выполнима.

Использование атрибутов $MustTaintedInterval$ и $MightTaintedInterval$ является лишним для логики программы, но позволяют оптимизировать запросы к SMT-решателю.

5.4.2 Предупреждение TAINTED_INT.LOOP

Является подвидом TAINTED_INT для уязвимостей, связанных с циклами.

Выдаётся для двух случаев.

- Помеченное значение используется для ограничения количества итераций цикла. Ошибка заключается в том, что цикл может выполняться излишне много итераций. Для определения, что переменная ограничивает количество итераций цикла используется информация о графе потока управления. Обработчик условных инструкций проверяет, что инструкция принадлежит сильно связанной компоненте и входное ребро является ребром входа в эту компоненту.
- Помеченное значение используется как шаг цикла. В этом случае, если злоумышленнику удастся установить такое значение, чтобы переменная не менялась на разных итерациях, в программе будет бесконечный цикл. Определение шага цикла реализовано более сложно. На первом этапе на анализе потока данных вычисляются инварианты цикла, т.е. такие переменные, которые имеют одно и то же значение на всех итерациях. Для всех остальных переменных проверяется, что они используются в арифметических инструкциях, их диапазон допускает нежелательное значение[7] и условные инструкции, проверяющие значения этих переменных.

[7] Чаще всего это 0. В некоторых ситуациях к ошибке может дополнительно приводить целочисленное переполнение.

Возможны ситуации, когда переменная используется в цикле в вызываемой функции. Для анализа используется атрибут $LoopBoundFlag$, который устанавливается в истину в тех случаях, когда анализ посчитал переменную ограничителем количества итераций цикла. Далее этот атрибут распространяется межпроцедурно, и если в момент применения резюме формальный аргумент имеет этот атрибут, а фактический – атрибут $MustTaintedInterval$, то будет выдано предупреждение.

5.4.3 Предупреждение TAINTED_INT.PTR

Если использовать помеченную целочисленную переменную без каких-либо проверок в качестве смещения указателя, то можно выйти за пределы выделенной памяти, так как значение помеченной переменной может быть произвольным.

```
void test(int fd, int *ptr) {
    int index;
    //значение index помечено
    int ret = recv(fd, &index, sizeof(index), 0);
    //использование помеченной переменной index как смещение
    ptr[index] = 3;//TAINTED_INT.PTR
}
```

В данном примере значение переменной *index* может быть любым, поэтому возможна ошибка переполнения буфера.

Детектор ищет ситуации, когда помеченные целочисленные переменные не проверяются и используются в качестве смещения указателя.

Условия выдачи срабатывания:

инструкция получения доступа к указателю,

- смещение имеет непустой $MustTaintedInterval$ или $MightTaintedInterval$,
- у смещения не проверяли нижнюю и верхнюю границу; атрибутов $MinIsTainted$ и $MaxIsTainted$ у смещения имеют значение *false*,
- размер выделенной памяти для указателя неизвестен, или же $MustTaintedInterval$ или $MightTaintedInterval$ интервал может превышать его размер.

5.5.4 Предупреждение TAINTED_ARRAY_INDEX

Предупреждение во многом похоже на TAINTED_INT.PTR с той разницей, что оно выдается при работе с массивами для которых статических анализатор знает размер.

```
void test(int fd) {
    int ptr[6];
    int index;
    //значение index получено из ненадежного источника
    int ret = recv(fd, &index, sizeof(index), 0);
    //использование помеченного значения в качестве индекса
    ptr[index] = 3;//TAINTED_ARRAY_INDEX
}
```

В данном примере нам известен размер массива *ptr*, и значение *index* может быть больше 5.

Детектор ищет ситуации, когда происходит доступ к массиву по индексу, где индекс получен из ненадежного источника и может иметь значение больше, чем размер массива. При этом размер массива известен анализатору.

Для определения размера массива используется атрибут $BufferSizeAttrVal$, который хранит возможный размер массива в виде интервала.

Условия выдачи срабатывания:

- доступ к массиву по индексу,
- нам известен размер массива; интервал из $BufferSizeAttrVal$ не пустой и не $[-\infty, +\infty]$,
- индекс имеет не пустой $MustTaintedInterval$ или $MightTaintedInterval$ атрибут,
- для индекса составляется формула с условием, что его значение может превышать интервал из $BufferSizeAttrVal$, эта формула выполнима.

5.4.5 Ошибки с целочисленным переполнением

Так как значение переменной, пришедшей из внешнего источника, может быть любым, то если осуществлять арифметические действия с данной переменной без предварительной проверки, то может произойти целочисленное переполнение и ее значение будет некорректным. Это может привести как к уязвимостям, так и к нарушению логики выполнения программы. Для таких ситуаций выдаётся предупреждение TAINTED.INT_OVERFLOW.

```
void test(int fd) {
    int ptr[6];
    int index;
    //значение index получено из ненадежного источника
    int ret = recv(fd, &index, sizeof(index), 0);
    //целочисленное переполнение
    index += 1; //TAINTED.INT_OVERFLOW
    if(index > 4) {
        return;
    }
    ptr[index] = 3;
}
```

В данном примере *index* может иметь максимальное значение для переменной типа *int*, поэтому при добавлении единицы может произойти целочисленное переполнение и ее значение станет некорректным.

Детектор проверяет ситуации, когда переменная из ненадежного источника участвует в арифметических операциях: сложение, умножение, вычитание. При помощи интервала из атрибута $MustTaintedInterval$ проверяется, возможно ли переполнение.

Условия выдачи срабатывания:

- инструкция сложения, вычитания, умножения,
- атрибут $MustTaintedInterval$ у одного из аргументов инструкции не пустой.
- интервал из $MustTaintedInterval = [-\infty; +\infty]$ или он может переполнить тип второго аргумента.

5.5 Помеченные строки

В некоторых ситуациях необходимо, чтобы указатель содержал в себе проверенные данные. Например, при открытии файла с помощью open или же когда осуществляется объединение или копирование строк. В случае `strcpy` и `strcat`, если строка `src` помечена, то её размер может быть больше чем у строки `dst`, что вызовет переполнение буфера.

```
void test(int fd, int *ptr, int size) {
    //env получен из ненадежного источника
    char* env = getenv("PATH");
    char *buf = malloc(size);
    //строка в env может быть любого размера
    //поэтому возможно переполнение buf при копировании
```

```
    strcpy(buf, env);//TAINTED_PTR
}
```

В данном примере размер строки в `env` может быть любым, поэтому возможна ошибка переполнения буфера при ее копировании в `buf`.

Детектор ищет ситуации, когда ненадежный указатель используется в функциях, где он может вызвать уязвимость.

Для идентификации испорченного указателя используются следующие атрибуты:

$TaintedPtrIf$ – атрибут хранит условия, при которых указатель будет содержать помеченные данные. Представляет собой формулу логики высказываний. Функция объединения атрибута: конъюнкция формул с веток.

$TaintedPtr$ – тернарный атрибут, показывает, содержит ли указатель помеченные данные.

С помощью тернарного атрибута $TrustedPtrSinkFlag$ детектор находит указатели, которые использовались в опасных функциях, где помеченное происхождение указателя может вызвать уязвимость (`open`, `strcpy`, `strcat` и т.д.).

Условия выдачи срабатывания:

- атрибут $TaintedPtr$ имеет значение *true* или *maybe*; указатель точно или же возможно получен из ненадежного источника,
- атрибут $TrustedPtrSinkFlag$ имеет значение *true*, указатель использовался в функции, где он может вызвать уязвимость,
- формула в атрибуте $TaintedPtrIf$ выполнима.

Также существуют ситуации, когда помеченный указатель не может вызвать уязвимость. Например, если при использовании `strcpy` известно, что для `dst`-строки выделено памяти достаточно, чтобы скопировать испорченную `src`-строку:

```
void test(int fd, int *ptr, int size) {
    //env содержит помеченные данные
    char* env = getenv("PATH");
    //размер buf зависит от длины строки в env
    char *buf = malloc(strlen(env) + 1);
    //переполнение невозможно
    strcpy(buf, env);//TAINTED_PTR
}
```

В данном примере для массива *buf* выделено достаточно памяти для копирования туда строки *env*.

Для отсеивания таких ситуация при копировании строки детектор узнает идентификатор, который является размером выделенной памяти для строки, после чего проверяет длины каких строк содержит данный идентификатор, если среди этих строк присутствует `dst`-строка, то ошибка не выдается. В примере, для переменной `buf` выделена память размера $(strlen(env) + 1)$, данный размер выделенной памяти включает в себя длину строки `env`, поэтому срабатывание не выдается.

Похожая ситуация есть и со `strcat`. Разница в том, что при присоединении новой строки проверяется включает ли в себя размер выделенной памяти `src`-строку со множеством строк из которых состоит `dst`-строка.

В случае, если помеченную строку сравнивали с другой строкой при помощи функций сравнения (`strcmp`), то срабатывания так же не выдается. Для идентификации таких помеченных строк используется тернарный атрибут $SanitizationInvoked$. Он помечает переменные, которые использовались в функциях сравнения строк.

6. Результаты

6.1 Анализ проектов с открытым исходным кодом

Для оценки процента истинных срабатываний использовался исходный код проекта Tizen 6 [18]. Проект Tizen – это открытая операционная система на базе ядра Linux. Общий размер проанализированного кода с помощью Svace составил более 32 миллионов строк. Результаты анализа приведены в табл. 1. Для оценки было размечено как минимум по 40 предупреждений для каждого вида детекторов. При этом не оценивалось эксплуатируемость ошибки. Предупреждение считалось истинным, если код содержит передачу небезопасных данных в критические операции; проверка достижимости пути из точек входа в программу не производилась.

Детектор TAINTED.INT_OVERFLOW оказался довольно шумным. Возможно, для него требуется доработка, чтобы исключить малополезные предупреждения. Большинство найденных срабатываний выглядят следующим образом:

```
int count;
count = strtol(arg, NULL, base);
```

Функция *strtol* возвращает тип *long*, поэтому потенциально возможна потеря значимой части возвращаемого значения.

Табл. 1. FP rate for Tizen 6
Table 1. Процент ложных срабатываний для проекта Tizen 6

Предупреждение	**Количество предупреждений**	**Процент истинных срабатываний**
TAINTED_ARRAY_INDEX	102	62,5
TAINTED_INT	137	65,5
TAINTED_INT.LOOP	137	76
TAINTED_INT.PTR	82	58
TAINTED_PTR	242	70,5
TAINTED.INT_OVERFLOW	796	85

6.2 Оценка Juliet 1.3

Проект Juliet [19] является тестовым набором для проверки возможностей статических анализаторов. Он включает как корректные тесты, где анализатор должен выдавать данные, так и ошибочные тесты, где анализатор не должен выдавать предупреждение.

Из набора типов ошибок в Juliet были взяты те, которые могут быть связаны с помеченными данными: CWE680, CWE194, CWE195, CWE789, CWE127, CWE124, CWE126, CWE134, CWE400. Данные тесты компилировались с опцией omitgood, которая скрывает все тесты, где ошибка отсутствует. Тесты используют специальную систему именования: имя теста можно разделить на части, каждая часть несет некоторую информацию, например, тип ошибки или же источник и приемник [20]. Основную информацию об используемых функциях в тесте можно получить из части под названием Functional Variant Name. Из получившихся выборки были отсеяны следующие тесты:

- тесты, которые компилируются только для windows; у таких тестов в Functional Variant Name содержатся w32, wchar.
- тесты, которые не содержат помеченные данные; у таких тестов в Functional Variant Name содержится не испорченный источник, а именно, функции: rand, new и другие.

На получившейся выборке было измерено покрытие тестов. Если один из помеченных детекторов выдавал ошибку в тесте, он считался покрытым. Все непокрытые тесты относятся к FN (False Negative). Табл. 2 содержит данные о проценте непокрытых текстов.

Табл. 2. FN rate for Juliet quality
Table 2. Процент пропусков для проекта Juliet

CWE	Количество тестов	Покрытие	FN(%)
CWE680	384	206	46,35
CWE194	816	444	45,59
CWE195	816	444	45,59
CWE789	384	92	76,04
CWE127	240	104	56,67
CWE124	240	104	56,67
CWE126	390	104	73,33
CWE400	624	164	73,72
CWE134	1200	226	81,17
Всего	5094	1888	62,93

Общее покрытие тестов составило 38,32%.

Для данной выборки также было измерено количество ложных срабатываний. Для этого компиляция тестов происходила с опцией omitbad, которая скрывает все тесты с ошибкой. Соответственно любое срабатывание будет ложным. Для тестовой выборки количество ложных срабатываний незначительно и составило 0,47%.

7. Заключение

Был описан межпроцедурный контекстно- и потоково-чувствительный анализ помеченных данных для программ на C, C++, Java, Kotlin и Go для поиска уязвимостей. В анализе используются хорошо известные и проверенные решения, которые можно встретить в других анализаторах.

Общая уникальная схема анализа была разработана за более, чем 10-летний опыт написания статических анализаторов. Получившееся решение не позволяет найти все уязвимости в программе, но процент найденных ошибок выше 38,32% для тестов Juliet.

Список литературы / References

[1] A. Belevantsev, A. Borodin, I. Dudina et al. Design and development of Svace static analyzers. In Proc. of the 2018 Ivannikov Memorial Workshop (IVMEM), 2018, pp. 3-9.

[2] А.Е. Бородин и А.А. Белеванцев. Статический анализатор Svace как коллекция анализаторов разных уровней сложности. Труды ИСП РАН, том 27, вып. 6, 2015 г., стр. 111-134. DOI: 10.15514/ISPRAS-2015-27(6)-8 / A. Borodin, A. Belevancev. A Static Analysis Tool Svace as a Collection of Analyzers with Various Complexity Levels. Trudy ISP RAN/Proc. ISP RAS, vol. 27, issue 6, 2015, pp.111-134 (in Russian).

[3] A. Borodin, A. Belevantsev, D. Zhurikhin, and A. Izbyshev. Deterministic static analysis. In Proc. of the 2018 Ivannikov Memorial Workshop (IVMEM), 2018, pp. 10-14.

[4] В.П. Иванников, А.А. Белеванцев, А.Е. Бородин и др. Статический анализатор Svace для поиска дефектов в исходном коде программ. Труды ИСП РАН, том 26, вып. 1, 2014 г., стр. 231-250. DOI: 10.15514/ISPRAS-2014-26(1)-7 / V. Ivannikov, A. Belevantsev, A. Borodin et al. Svace: static analyzer for detecting of defects in program source code. Trudy ISP RAN/Proc. ISP RAS, vol. 26, issue 1, 2014, pp.231-250 (in Russian).

[5] Aleph One. Smashing the stack for fun and profit. Phrack magazine, vol. 7, issue 49, 1996, pp. 14-16.

[6] National Vulnerability Database – CWE Over Time. 2020. URL: https://nvd.nist.gov/general/visualizations/vulnerability-visualizations/cwe-over-time. Accessed 15.01.2021.

[7] W. Landi. Undecidability of static analysis. ACM Letters on Programming Languages and Systems (LOPLAS), vol. 1, no. 4, 1992, pp. 323-337.
[8] M. Hind. Pointer analysis: haven't we solved this problem yet? In Proc. of the ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, 2001, pp. 54-61.
[9] W. Landi. Interprocedural aliasing in the presence of pointers. PhD Thesis, The State University of New Jersey, 1992, 268 p.
[10] W. Landi, B.G. Ryder. A safe approximate algorithm for interprocedural aliasing. ACM SIGPLAN Notices, vol. 27, no. 7, 1992, pp. 235-248, 1992.
[11] B. Livshits, M. Sridharan, Y. Smaragdakis et al. In defense of soundiness: a manifesto. Communications of the ACM, vol. 58, no. 2, 2015, 44-46.
[12] А. Белеванцев, А. Избышев, Д. Журихин. Организация контролируемой сборки в статическом анализаторе Svace. Системный администратор, вып. 7-8, 2017 г., стр. 135-139 / A. Belevantsev, A. Izbyshev, D. Zhurikhin. Monitoring program builds for Svace static analyzer. System Administrator, issues 7-8, 2017, pp. 135-139 (in Russian).
[13] W.R. Bush, J.D. Pincus, and D.J. Sielaff. A static analyzer for finding dynamic programming errors. Software-Practice and Experience, vol. 30, issue 7, 2000, pp. 775–802.
[14] A. Aiken, S. Bugrara, I. Dillig et al. An overview of the saturn project. In Proc. of the 7th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, 2007, pp. 43–48.
[15] D. Babic and A.J. Hu. Calysto: scalable and precise extended static checking. In Proc. of the 30th international conference on Software engineering, 2008, pp. 211-220.
[16] В.К. Кошелев, В.Н. Игнатьев, А.И. Борзилов. Инфраструктура статического анализа программ на языке C#. Труды ИСП РАН, том 28, вып. 1, 2016 г., стр. 21-40. DOI: 10.15514/ISPRAS-2016-28(1)-2 / V. Koshelev, V. Ignatiev, A. Borzilov, and A. Belevantsev. SharpChecker: static analysis tool for C# programs. Programming and Computer Software, vol. 43, no. 4, 2017, pp.:268–276.
[17] Р.Р. Мулюков, А.Е. Бородин. Использование анализа недостижимого кода в статическом анализаторе для поиска ошибок в исходном коде программ. Труды ИСП РАН, том 28, вып. 5, 2016 г.,, стр. 145-158 / R.R.Mulyukov,A.E.Borodin. Using unreachable code analysis in static analysis tool for finding defects in source code. Trudy ISP RAN/Proc. ISP RAS, 2016, vol. 28, issue 5, 2016, pp. 145-158 (in Russian). DOI: 10.15514/ISPRAS-2016-28(5)-9.
[18] Tizen 6.0 Public M2 Release. URL: https://www.tizen.org/blogs/bighoya/2020/tizen-6.0-public-m2-release-0, accessed 15.01.2021.
[19] P.E. Black. Juliet 1.3 Test Suite: Changes from 1.2. US Department of Commerce, National Institute of Standards и Technology, 2018, 37 p.
[20] Juliet Test Suite v1.2 for C/C++ User Guide. Center for Assured Software, National Security Agency, 2012, 41p.

Информация об авторах / Information about authors

Алексей Евгеньевич БОРОДИН – кандидат физико-математических наук, научный сотрудник. Сфера научных интересов: статический анализ исходного кода программ для поиска ошибок.

Alexey Evgenevich BORODIN – PhD, researcher. Research interests: static analysis for finding errors in source code.

Алексей Вячеславович ГОРЕМЫКИН – студент магистратуры ф-та ВМК, стажер ИСП РАН. Научные интересы: статический анализ исходного кода программ.

Alexey Vyacheslavovich GOREMYKIN – Master's student at the Faculty of Computational Mathematics and Cybernetics, intern at ISP RAS. Research interests: static analysis of the source code of programs.

Сергей Павлович ВАРТАНОВ – научный сотрудник. Сфера научных интересов: символьное исполнение, динамический и статический анализ программ, SMT-решатели.

Sergey Pavlovitch VARTANOV – Researcher. Research interests: symbolic execution, dynamic and static analysis, and SMT solvers.

Андрей Андреевич БЕЛЕВАНЦЕВ – доктор физико-математических наук, ведущий научный сотрудник ИСП РАН, профессор кафедры системного программирования ф-та ВМК МГУ. Область научных интересов – методы оптимизации программ, планирование команд, динамические оптимизации, методы анализа потоков данных и управления, гетерогенные вычислительные системы.

Andrey Andreevich BELEVANTSEV – Doctor of Physical and Mathematical Sciences, Leading Researcher of ISP RAS, Professor of the Department of System Programming of the Faculty of Computer Science of Moscow State University. Research interests: methods of program optimization, command planning, dynamic optimization, methods of data flow analysis and control, heterogeneous computing systems.

DOI: 10.15514/ISPRAS–2021–33(1)–2

Обучение многослойного перцептрона с учителем в задаче распознавания с помощью корреляционного показателя

Н.А. Вершков, ORCID: 0000-0001-5756-7612 <vernick61@yandex.ru>
М.Г. Бабенко, ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>
В.А. Кучуков, ORCID: 0000-0002-1839-2765 <vkuchukov@ncfu.ru>
Н.Н. Кучукова, ORCID: 0000-0002-8070-0829 <nkuchukova@ncfu.ru>

Северо-Кавказский федеральный университет,
355017, Россия, г. Ставрополь, ул. Пушкина, д. 1

Аннотация. В статье рассматривается задача распознавания рукописных цифр с помощью искусственных нейронных сетей прямого распространения (перцептронов) с использованием корреляционного показателя. Предлагаемый метод базируется на математической модели нейронной сети как колебательной системы, аналогичной системе передачи информации. В статье используются теоретические наработки авторов по поиску глобального экстремума функции ошибки в искусственных нейронных сетях прямого распространения. Изображение рукописной цифры рассматривается как одномерный входной дискретный сигнал, представляющий собой смесь «идеального написания цифры» и шума, который описывает отклонение входной реализации от «идеального написания». Для формирования функции ошибки используется широко используемый в системах передачи информации критерий идеального наблюдателя (Котельникова), описывающий вероятность верного распознавания входного сигнала системой передачи информации. В статье проводится сравнительный анализ сходимости обучающей и экспериментально полученной последовательностей на основе корреляционного показателя и широко используемой в задачах классификации функции CrossEntropyLoss с использованием опимизатора и без него. На основе проведенных экспериментов делается вывод о преимуществе предлагаемого корреляционного показателя в 2-3 раза.

Ключевые слова: искусственные нейронные сети; интеллектуальный анализ данных; корреляционная функция; спектральный анализ

Для цитирования: Вершков Н.А., Бабенко М.Г., Кучуков В.А., Кучукова Н.Н. Обучение многослойного перцептрона с учителем в задаче распознавания с помощью корреляционного показателя. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 33-46. DOI: 10.15514/ISPRAS–2021–33(1)–2

Благодарности. Данная работа выполнена при поддержке Российского фонда фундаментальных исследований, проект №20-37-70023, гранта Президента Российской Федерации МК-341.2019.9 и стипендии Президента Российской Федерации СП-2236.2018.5.

Advanced supervised learning in multi-layer perceptrons to the recognition tasks based on correlation indicator

N.A. Vershkov, ORCID: 0000-0001-5756-7612 <vernick61@yandex.ru>
M.G. Babenko, ORCID: 0000-0001-7066-0061 <mgbabenko@ncfu.ru>
V.A. Kuchukov, ORCID: 0000-0002-1839-2765 <vkuchukov@ncfu.ru>
N.N. Kuchukova, ORCID: 0000-0002-8070-0829 <nkuchukova@ncfu.ru>

North-Caucasus Federal University,
1, Pushkin st., Stavropol, 355017, Russia

Abstract. The article deals with the problem of recognition of handwritten digits using feedforward neural networks (perceptrons) using a correlation indicator. The proposed method is based on the mathematical model of the neural network as an oscillatory system similar to the information transmission system. The article uses theoretical developments of the authors to search for the global extremum of the error function in artificial neural networks. The handwritten digit image is considered as a one-dimensional input discrete signal representing a combination of "perfect digit writing" and noise, which describes the deviation of the input implementation from "perfect writing". The ideal observer criterion (Kotelnikov criterion), which is widely used in information transmission systems and describes the probability of correct recognition of the input signal, is used to form the loss function. In the article is carried out a comparative analysis of the convergence of learning and experimentally obtained sequences on the basis of the correlation indicator and widely used in the tasks of classification of the function CrossEntropyLoss with the use of the optimizer and without it. Based on the experiments carried out, it is concluded that the proposed correlation indicator has an advantage of 2-3 times.

Keywords: artificial neural networks; data mining; correlation function; spectral analysis

For citation: Vershkov N.A., Babenko M.G., Kuchukov V.A., Kuchukova N.N. Advanced supervised learning in multi-layer perceptrons to the recognition tasks based on correlation indicator. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 33-46 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–2

Acknowledgements. The reported study was funded by RFBR, project number 20-37-70023, and Russian Federation President Grant MK-341.2019.9 and SP-2236.2018.5.

1. Введение

Подавляющее большинство видов человеческой деятельности связано с накоплением и обработкой больших объемов информации (data mining). Поэтому так много внимания уделяется современным инструментам обработки информации, среди которых достойное место занимают искусственные нейронные сети (ИНС). Теоретической основой ИНС является теорема Колмогорова – Арнольда [1, 2], важнейшим следствием которой является возможность представления функции нескольких переменных в виде суперпозиции функций меньшего числа переменных, т.е. $f(x_1, x_2, \dots, x_n) = \sum_{i=1}^{2n+1} h_i(\sum_{k=1}^{n} \varphi_{ki}(x_k))$, где h_i, φ_{ki} – непрерывные функции, причем φ_{ki} не зависят от *f*. Дальнейшие теоретические разработки имели прикладное значение: примером может служить теорема Хехт-Нильсена [3]. Кроме того, широко известна классическая теорема Вейерштрасса [4] о возможности приближения функции *n* переменных с любой точностью с помощью полинома. А более общая теорема Стоуна [4] утверждает возможность приближения многочленом любого конечного набора функций. Таким образом, можно сделать вывод о том, что с помощью ИНС можно реализовать практически любую, сколь угодно сложную функцию любого количества переменных.

Первые нейронные сети МакКаллока-Питтса не обучались, веса связей нейронов устанавливались заранее. Впервые идею обучения нейронных сетей предложил Дональд Хэбб (Donald Olding Hebb**)** в 1949 году [5]. Большинство современных обучающих алгоритмов основаны на принципах обучения Хэбба. В настоящее время алгоритмы обучения ИНС представлены значительным многообразием и различаются по видам решаемых задач.

Однако основным математическим аппаратом является метод градиентного спуска, опирающийся на дифференциальный анализ слоев ИНС первого и второго порядка. Сейчас, в том числе благодаря достижениям команды Хинтона (Geoffrey E. Hinton) [6], уделяется большое внимание «глубокому обучению» (Deep Learning). Использование многослойных сетей изначально было ограничено сложностями их обучения. Благодаря идеям команды Хинтона стало возможным обучение многослойных ИНС [7]. Другим способом улучшения машинного обучения является применение ансамблей предиктивных моделей, таких как персептрон и деревья решений [8].

Таким образом, современное развитие методов обучения ИНС является скорее интуитивно-алгоритмическим, чем математическим. Применение стандартных алгоритмов облачных вычислений [9] хотя и позволяет упростить обучение моделей и эффективно использовать вычислительные ресурсы, не меняет принципиального подхода к обучению нейронных сетей. И, несмотря на достигнутые успехи и сокращение времени обучения в десятки, а иногда и в сотни раз, на данном направлении остается ряд задач, требующих теоретического осмысления. К ним, в первую очередь, относится задача поиска глобального экстремума целевой функции и конечность алгоритма обучения [10].

Широкое применение ИНС нашли и в теории связи. Так, в [11] обсуждается использование нейронных сетей для реализации быстрых алгоритмов спектральных преобразований. В [12] рассматриваются нейросетевые методы классификации источников сигналов в когнитивных радиосистемах. Однако обратного процесса применения основ теории передачи информации к нейронным сетям в литературе авторы не встречали. Ранее авторы рассматривали моделирование ИНС как систему связи [13].

В этой работе мы проанализировали возможность использования методов, широко используемых в теории передачи информации для распознавания сигнала на фоне шума и сосредоточенных помех [14-18] для поиска экстремума целевой функции при обучении ИНС с учителем. Нами была предложена целевая функция для оценки качества обучения в виде показателя взаимно-корреляционной функции экспериментально полученной и обучающей последовательностей. Предложенный подход поможет в решении задачи преодоления локальных экстремумов с целью поиска глобального.

Статья организована следующим образом. В разд. 2 исследуются существующие методы оценки ошибки обучения и теоретические основы алгоритмов минимизации этой ошибки, а также сходимость обучающей и экспериментально полученной последовательности и сравнению широко используемых в обучении ИНС целевых функций с предлагаемым показателем. В разд. 3 проводится сравнительный анализ предложенного метода с существующими для получения оценки эффективности обучения для задач классификации. В разд. 4 определяются основные направления исследования нейронных сетей как системы передачи информации.

2. Анализ существующих методов определения ошибки обучения

Процесс обучения нейронной сети – это процесс определения весов соединений между нейронами таким образом, чтобы сеть аппроксимировала необходимую функцию с заданной точностью.

Модель обучения с учителем состоит из трех взаимосвязанных компонент [19]: среды, которая характеризуется распределением вероятностей $P(X_i)$ со случайно и независимо появляющимися элементами входного воздействия X_i, учителя, который генерирует желаемый вектор Y_i отклика на входное воздействие X_i и обучаемой машины, т.е. нейронной сети, способной реализовать множество функций отображения вход-выход. При этом ни характеристика среды $P(X_i)$, ни правило классификации $P(Y_i|X_i)$, как правило, неизвестны. Известно только, что обе функции существуют, т.е. существует совместное распределение вероятностей $P(Y_i, X_i) = P(X_i)P(Y_i|X_i)$. Требуется определить функцию отображения $\hat{Y}_i =$

$F(X_i, w)$, т.е. фактический отклик, сгенерированный обучаемой машиной такой, что ожидаемая величина потерь определяется как функционал среднего риска $R(w) = \int L\big(Y_i, F(X_i, w)\big)dP(Y_i|X_i)$, где $L\big(Y_i, F(X_i, w)\big)$ – мера потерь между ожидаемым откликом Y_i и откликом обучаемой машины $\hat{Y}_i$.

2.1 Теоретические основы алгоритма минимизации ошибки обучения

Большой вклад в поиск путей преодоления априорной неопределенности при решении задач приема и обработки информации внес Б. Уидроу (Bernard Widrow) [16]. Он исследовал и обобщил алгоритмы работы с линейным сумматором, который является основой современных ИНС, а также провел анализ используемых функций ошибок и исследовал области применения адаптивной обработки информации. Используя предположение о стационарности входной функции $P(X_i)$ и отклика $P(Y_i|X_i)$, Уидроу сформулировал основные принципы метода градиентного спуска, который на сегодняшний день является основным инструментом обучения ИНС.

Работы отечественных авторов, в частности Я.З. Цыпкина [20], позволили значительно расширить подходы Уидроу для различных классов функций потерь на основе априорной информации о помехах и классах распределений. Я. З. Цыпкин, исследуя оптимальные функции потерь для различных классов распределения вероятности помехи, приходит к выводу, что [20] «… оптимальное решение, минимизирующее средние потери, может быть найдено в крайне редких случаях. Как правило, приходится довольствоваться оценками оптимального решения, минимизирующими эмпирические средние потери». Оптимальная функция потерь может быть определена как $F_{\text{опт}}(\varepsilon) = -\ln(p(\xi))|_{\xi=\varepsilon}$, где $p(\xi)$ – плотность распределения помех [20]. Таким образом, знание плотности распределения помех позволяет определить оптимальную функцию потерь, а значит, и оптимальные средние потери.

А. А. Сикарев и О. Н. Лебедев [15] проводят анализ помехоустойчивости систем передачи информации (СПИ) сложных широкополосных сигналов с целью максимально точного восстановления сигнала на выходе приемника. Для анализа работы системы передачи информации удобно представлять функцию $x(t)$ в обобщенной спектральной форме

$$x_r(t) = \sum_{k=k_{r1}}^{k_{rn}} a_{kr}\varphi_k(t)\,, t \in [t_1, t_n], \tag{2.1}$$

где координатные функции $\varphi_k(t)$ удовлетворяют условию ортогональности

$$\frac{1}{T}\int_0^T \varphi_k(t)\varphi_j(t)dt = \begin{cases} 0, \text{при } k \neq j \\ \frac{1}{T}\int_0^T \varphi_k^2(t)dt, \text{при } k = j \end{cases},$$

а коэффициенты разложения

$$a_{kr} = \frac{1}{\int_0^T \varphi_k^2(t)dt}\int_0^T x_r(t)\varphi_k(t)dt.$$

Для формирования сложных сигналов обычно используют совокупность координатных функций как некоторое подмножество полной ортогональной системы функций: тригонометрических, Лаггера, Лежандра, Эрмита, Уолша, Чебышева и т.п. [15]. Представление (2.1) позволяет более наглядно представить формирование и обработку сложных функций в частотно-временной области. Подобный подход может быть применен и к ИНС.

Отличие и особенности структуры различных вариантов применяемых сложных сигналов в частотно-временной области описываются также и корреляционными функциями. Двумерная корреляционная функция сигналов $x_r(t)$ и $x_l(t)$ выглядит как

$$R_{rl}(\tau,\Omega)=\frac{1}{2T\sqrt{P_r P_l}}\int_{-\infty}^{\infty}x_r(t)x_l^*(t-\tau)e^{j\Omega t}dt$$

Здесь $P_i=\frac{1}{T}\int_0^T x_i^2(t)dt$ – средняя мощность i-того варианта сигнала за период, * - знак комплексного сопряжения (по Гильберту), τ и Ω – сдвиги одного сигнала относительно другого соответственно по времени и частоте.

Полагая, что выход ИНС $\hat{Y}_l=F(X_i,W_j)$ является комбинацией функции Y_l и помехи μ_l, возникающей в связи с некорректно подобранным набором значений W_j, количественное выражение этой меры может быть определено исходя из критерия идеального наблюдателя (Котельникова) как коэффициент взаимного различия (КВР) функции отклика $\hat{Y}(t)$ и целевой функции $Y(t)$ [15, 21]:

$$g_k^2=\frac{l_x^2+l_y^2}{4P_Y P_{\hat{Y}}},\qquad(2.2)$$

где $l_x=\frac{2y\mu}{T}\int_{t_0}^{t_n}Y(t)\hat{Y}(t)dt$ и $l_y=\frac{2y\mu}{T}\int_{t_0}^{t_n}Y(t)\hat{Y}^*(t)dt$ [10]. Для сигналов, ограниченных во времени прямоугольным окном $[0,t_n]$, показатель будет иметь вид, изображенный на рис. 1.

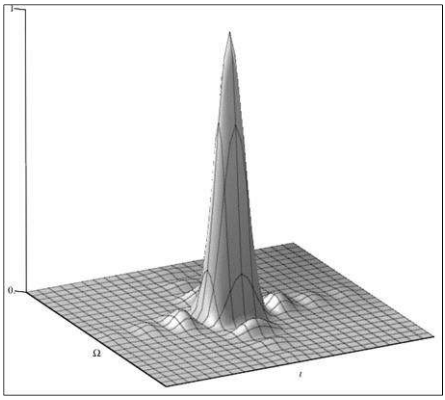


Рис. 1. Зависимость показателя g_k^2 от τ и Ω
Fig. 1. The dependency of g_k^2on τ and Ω

Таким образом, КВР может быть использован в виде функции потерь для обучения нейронной сети.

2.2 Виды функций потерь, применяемых на практике

В процессе построения и эволюции ИНС основным инструментом оценки достижения цели в процессе обучения стал метод наименьших квадратов (МНК). МНК в общем виде – инструмент математической статистики, позволяющий получать несмещенную оценку приближения получаемых и ожидаемых выходных значений, на основании которой принимается решение об изменении весовых параметров ИНС [22]. МНК применяется для оценки статических моделей. Дело в том, что помеха μ, влияющая на отклик ИНС $\hat{Y}(t)$, обычно полагается гауссовой, а параметрически неопределенное описание функции правдоподобия выглядит как

$$P(Y_i|X_i)=\frac{1}{(2\pi\sigma^2)^{n/2}}\exp\left\{1-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2\right\}.$$

Проблема МНК в том, что матричная алгебра хотя и позволяет создание многопараметрических моделей, но они все являются линейными. Конечно, в общем случае, в качестве аппроксимирующей функции может быть использована квадратичная, экспоненциальная и любая другая. Однако в подавляющем большинстве случаев

применяется именно линейная модель как наиболее простая из перечисленных. Отсюда текущие проблемы с обучением ИНС – множество локальных экстремумов, отсутствие экстремумов на значительном промежутке значений весовых параметров $\{W_k\}$ и т.п. Одной из основных гипотез МНК является предположение о равенстве дисперсий отклонений, т.е. их разброс вокруг среднего (нулевого) значения ряда должен быть величиной стабильной [23]. На практике дисперсии отклонений достаточно часто неодинаковы, то есть наблюдается гетероскедастичность.

Таким образом, устоявшимся подходом для оценки сходимости фактического и желаемого отклика за все время изучения нейронных сетей является широко применяемый в математической статистике МНК, для которого целевой функцией является суммарная квадратичная ошибка

$$E_\Sigma = \sum_n E(n), E(n) = \frac{1}{2}\sum_i e_i^2(n)$$

Здесь $E(n)$ – сумма квадратов ошибок $e_i(n)$ всех нейронов выходного слоя, т.е. $e_i(n) = Y_i - Y_i^k$. При этом математическая форма алгоритма обучения представлена как

$$\frac{\partial E_\Sigma}{\partial w_k} = \sum_n \frac{\partial E(n)}{\partial w_k}$$

которая именуется методом градиентного спуска. Выбирая соответствующим образом величину Δ как величину градиента и опираясь на минимум суммы квадратов ошибок, подбирают вектор изменения значений $\{W_k\}$.

МНК наиболее часто применяется в задачах регрессии, в которых выходные значения ИНС и целевые значения представляют собой непрерывную величину в отличие от задач классификации, в которых число классов дискретно. Поскольку физический смысл целевой переменной при классификации имеет совершенно иную направленность, то предсказываются не сами метки, а их логарифмическое представление. Поэтому в задачах классификации чаще используют Bernoulli loss вида

$$L_{CE}(Y, \hat{Y}) = \log(1 + \exp(-2Y \log \hat{Y}))$$

или Adaboost loss вида

$$L_{CE}(Y, \hat{Y}) = \exp(Y \log \hat{Y}).$$

Для многоклассовой классификации выходные значения ИНС часто интерпретируются как вероятность принадлежности значения к определенному классу. При этом дискретная перекрестная энтропия оценивает векторы Y и $\hat{Y}$ как $L_{CE}(Y, \hat{Y}) = -\sum_{i=1}^m y_i \log(\hat{y}_i)$ [24]. Перекрестная энтропия (Cross Entropy loss) в настоящее время является наиболее часто используемой функцией ошибок для задач классификации.

2.3 Взаимно-корреляционная функция как мера сходства и различия

В процессе обучения ИНС является динамической системой с обратной связью, в которой изменение весовых параметров $\{W_k\}$ осуществляется на основе некоторой целевой функции $\delta(Y, \hat{Y})$, где Y – ожидаемые выходные значения, а $\hat{Y}$ – фактически полученные выходные значения, δ – функция невязки. Изменение параметров $\{W_k\}$, в свою очередь, ведет к изменению значений $\hat{Y}$. В теории управления такое поведение объекта называют параметрической идентификацией модели. Для процесса обучения крайне важна эта обратная связь, т.к. движение значений весовых параметров $\{W_k\}$ в одном направлении чревато вхождением в «область насыщения», когда даже значительные изменения входных данных не вызывают никаких изменений на выходе. Основным достоинством предлагаемого показателя является отсутствие каких-либо требований к виду выходной функции и зависимостью между Y и $\hat{Y}$. Взаимно-корреляционная функция вида $R = \frac{1}{K}\int_{-\infty}^{\infty} y(t) y_k(t -$

$\tau)d\tau$ или $R = \frac{1}{K}\int_{-\infty}^{\infty} y(\Omega)y_k(\Omega - \Delta\Omega)d\tau$ (где K – нормирующий коэффициент) оценивает нормированную взаимную энергию функций $y(t)$ и $y_k(t)$, если они пересекаются на интервале τ (и имеют общий спектр) [21]. Обучающая последовательность $\{X_i, Y_i\}$ может быть представлена не в виде многомерных векторов, а в виде одномерных дискретизированных сигналов $x(t), y(y)$. Как видно из рис. 1, на всем протяжении показатель $g^2 = R^2(\tau, \Omega)$ имеет один значительный локальный максимум и не зависит от передаточной характеристики ИНС.

При анализе сложных сигналов в каналах с помехами, а также при оценке помехоустойчивости таких устройств важной является мера различимости структуры сигналов и воздействующих помех в частотно-временной области [10]. Полагая, что $y_k(t)$ является смесью полезной функции $y(t)$ и помехи $\mu(t)$, возникающей в связи с неудачно подобранным набором значений W_i, количественное выражение этой меры может быть определено как коэффициент взаимного различия функции отклика $y_k(t)$ и целевой функции $y(t)$ вида (2.2). Переходя от КВР для аналоговых сигналов вида (2.2) к дискретному, получим:

$$k = \frac{\sum y_i \hat{y}_i{}^2 + \sum y_i \hat{y}_i{}^{*2}}{4\sum y_i^2 \sum \hat{y}_i^{*2}}. \quad (2.3)$$

Выражение (2.3) оценивает меру сходства ожидаемой последовательности на выходе ИНС с фактической.

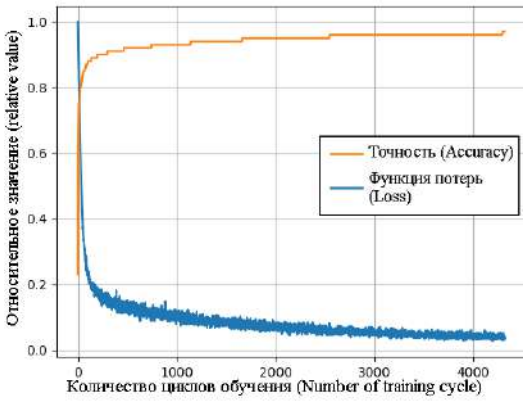
Таким образом, двумерная (комплексная) взаимно-корреляционная функция может служить математической моделью, которая позволяет отслеживать влияние параметров ИНС на отклонение фактических выходных значений от желаемых, а применение квадрата взаимно-корреляционной функции для анализа расхождений ожидаемой функции $\{y_i(t)\}$ и фактически полученной для набора весов $\{W_k\}$ функции $\{y_i^k(t)\}$ позволяет осуществлять оценку для всего обучающего множества.

3. Сравнительная характеристика предлагаемой функции потерь с существующими

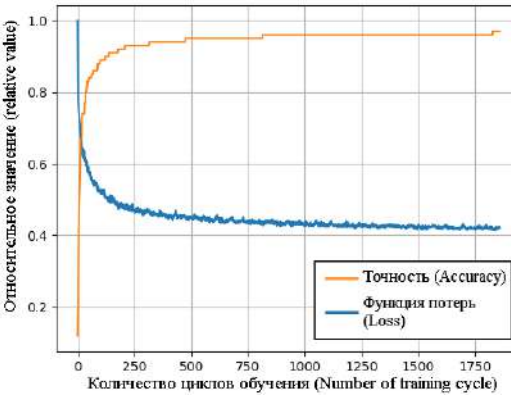
Для проведения сравнительных характеристик предлагаемой функции ошибок (КВР g^2) с наиболее часто применяемыми на практике функциями была использована широко известная база данных MNIST [25]. Эта база является стандартом, который был предложен Национальным институтом стандартов и технологий США для сопоставления методов распознавания изображений с помощью ИНС. Предлагаемая база данных MNIST содержит 60000 изображений для обучения и 10000 изображений для тестирования.

Модуль ИНС был создан на языке Python с использованием библиотеки PyTorch [26]. Для тестирования применялись ИНС прямого распространения с количеством полносвязных скрытых слоев от 1 до 8 и функцией активации ReLU. Для получения сравнительных характеристик была использована функция потерь CrossEntropyLoss, а для реализации показателя «коэффициент взаимного различия, КВР» был написан класс MDCLoss на основе выражения (2.3). В качестве показателя, по которому производилось сравнение, было выбрано количество циклов обучения ИНС до достижения значения ошибки при тестировании 3%. Для всех тестов скорость обучения была установлена 0,1.

В качестве первоначального теста использовалась ИНС с одним скрытым слоем. Обучение проводилось без применения оптимизатора и показало результаты, представленные на рис. 2.

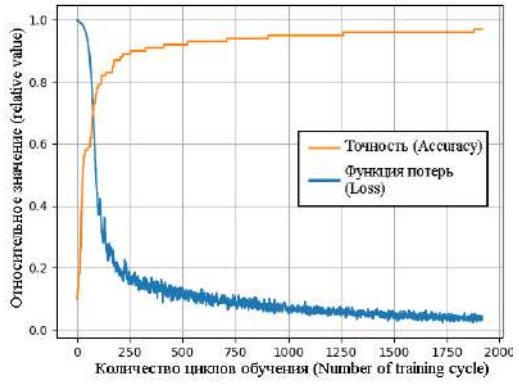


а). Функция CrossEntropyLoss

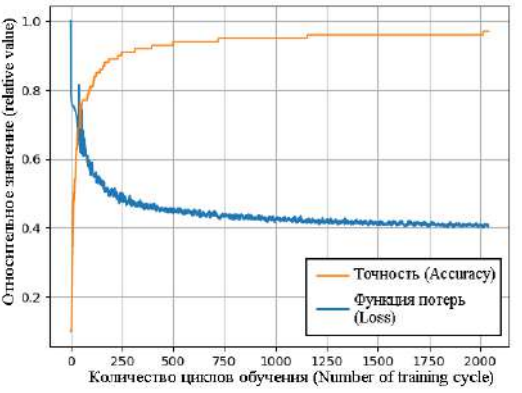


б). Функция KBP

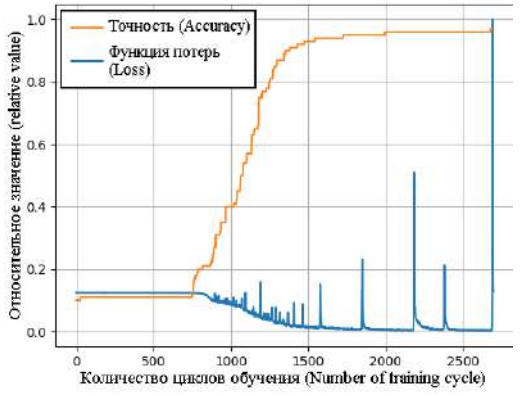
Рис. 2. Сравнительная характеристика обучения ИНС с 1 скрытым слоем
Fig. 2. Comparative characteristic of the ANN training with 1 hidden layer



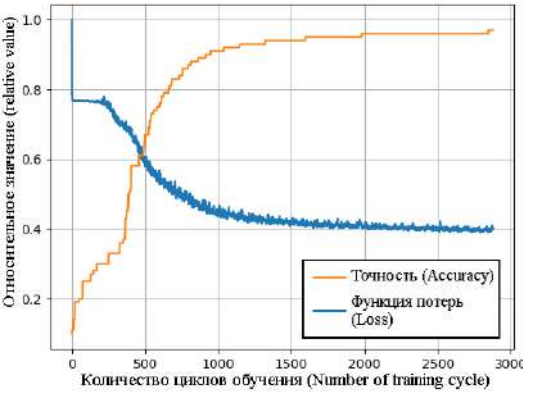
а). Функция CrossEntropyLoss, ИНС 4 скрытых слоя



б). Функция KBP, ИНС 4 скрытых слоя



в). Функция CrossEntropyLoss, ИНС 8 скрытых слоев



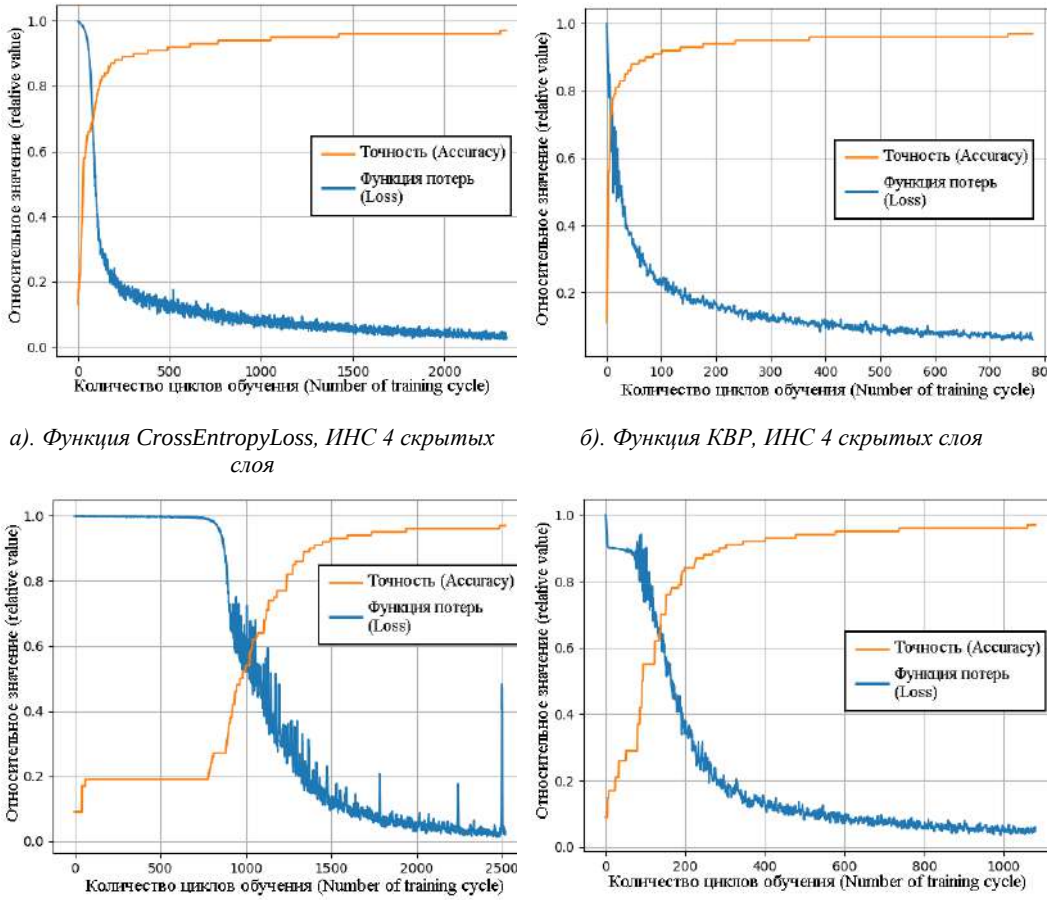
г). Функция KBP, ИНС 8 скрытых слоев

Рис. 3. Сравнительная характеристика обучения ИНС для 4-х и более скрытых слоев без оптимизатора
Fig. 3. Comparative characteristic of the ANN training for 4 or more hidden layers without the optimizer

Здесь и далее на всех рисунках синей линией обозначена динамика относительной функции ошибки, а оранжевой – точность обучения, выраженная $^{\%}/_{100}$. По оси *x* откладывается количество циклов обучения ИНС. Из рис. 2 наглядно видно, что обучение ИНС с использованием предлагаемого показателя КВР произошло почти в 3 раза быстрее, чем с применением функции CrossEntropyLoss. Однако при росте количества скрытых слоев до 4-х или даже до 8-ми это преимущество теряется и количество циклов обучения практически выравнивается как показано на рис. 3.

Для объяснения данного эффекта рассмотрим алгоритм обновления параметров для обычного градиентного спуска

$$W_i = W_{i-1} - \eta \nabla L_{CE}(X_i, W_{i-1}) \quad (3.1)$$



а). Функция CrossEntropyLoss, ИНС 4 скрытых слоя

б). Функция КВР, ИНС 4 скрытых слоя

в). Функция CrossEntropyLoss, ИНС 8 скрытых слоев

г). Функция КВР, ИНС 8 скрытых слоев

Рис. 4. Сравнительная характеристика обучения ИНС для 4-х и более скрытых слоев с оптимизатором

Fig. 4. Comparative characteristic of the ANN training for 4 or more hidden layers with the optimizer

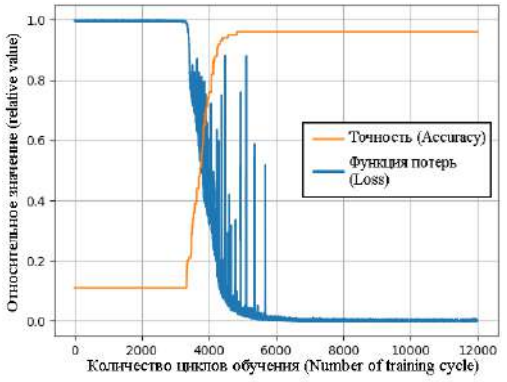
Здесь W_i – значения весов ИНС на текущем шаге, W_{i-1} – значения весов на предыдущем шаге, η – скорость обучения, ∇ - градиент на текущем шаге, L_{CE} – функция потерь. Для одного слоя формула расчета градиента строится с помощью первой производной. А когда речь идет о большом количестве слоев с использованием нелинейных функций, то расчет значения ∇L_{CE} для каждого слоя занимает уже значительное время. Поэтому при росте количества слоев

преимущество предлагаемого показателя уже не выглядит столь очевидным, как в однослойной ИНС.

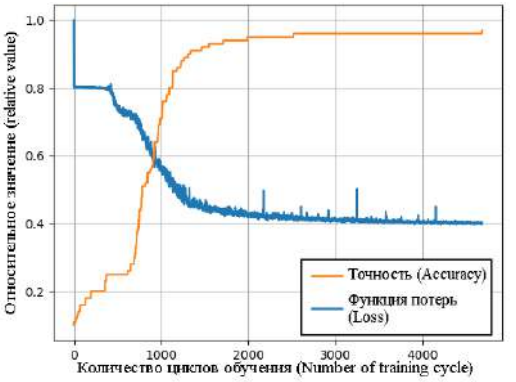
Для повышения эффективности обучения многослойных ИНС применяют оптимизаторы. Суть работы оптимизатора заключается в использовании истории градиента целевой функции. Например, метод Нестерова использует экспоненциальное скользящее среднее. Более совершенные методы, такие как Adagrad или Adadelta позволяют использовать наиболее информативные признаки классификации.

Экспериментально было установлено, что из набора оптимизаторов, входящих в torch.optim, наиболее подходящим для функции КВР является Adadelta. Для того чтобы выдержать одинаковые условия для всех тестов в дальнейшем применялся оптимизатор Adadelta. Сравнительная характеристика обучения ИНС с использованием оптимизатора представлена на рис. 4.

Из рис. 4 видно, что применение функции КВР совместно с оптимизатором Adadelta позволяет получить выигрыш в скорости обучения 2-3 раза. Для того чтобы получить полную картину сравнительного анализа предлагаемой функции ошибки КВР с CrossEntropyLoss, постепенно начали увеличивать количество слоев в ИНС до момента, пока одна из функций ошибки не сможет вывести ИНС из локального минимума за ограниченное количество циклов обучения. Это произошло при числе слоев ИНС равном 10 и представлено на рис. 5.



а). Функция CrossEntropyLoss, ИНС 10 скрытых слоев



б). Функция КВР, ИНС 10 скрытых слоев

Рис. 5. Сравнительная характеристика обучения ИНС для 10-ти скрытых слоев с оптимизатором
Fig. 5. Comparative characteristic of the ANN training for 10 hidden layers with the optimizer

На рис. 5а наглядно видно, что значение функции ошибки колеблется в районе 0,001 – 0,003 при количестве циклов обучения более 10000. При этом достоверность распознавания ИНС набора для тестирования зафиксировано на уровне 96% и не изменяется при количестве циклов обучения 5000 и более. Эксперимент проводился до достижения достоверности распознавания ИНС 97% или 12000 циклов обучения (200 эпох). Завершение эксперимента произошло при достижении предельного количества эпох обучения.

В тоже время применение функции КВР (рис. 5б) позволило достичь уровня достоверности 96% при прохождении менее 3000 циклов обучения (менее 50 эпох). Уровень достоверности 97% был достигнут ИНС при количестве циклов обучения менее 5000.

Таким образом, проведен сравнительный анализ предлагаемого показателя функции ошибок «коэффициент взаимного различия, КВР» с наиболее широко применяемой функцией ошибок в задачах классификации «CrossEntropyLoss» на ИНС прямого распространения. Анализ показал, что применение сравниваемых функций без оптимизатора примерно равнозначно при числе слоев 4 и более. При меньшем количестве слоев преимущество имеет КВР. Сравнительный анализ с использованием оптимизатора Adadelta показал преимущество

в скорости обучения предлагаемого показателя в 2-3 раза на ИНС с числом скрытых слоев менее 10. При увеличении числа скрытых слоев ИНС до 10 функция «CrossEntropyLoss» показала неспособность достичь ожидаемого уровня достоверности ИНС в разумных пределах. В это же время, предлагаемый показатель достиг ожидаемого уровня достоверности в реальные сроки.

Из всего вышесказанного следует, что предлагаемый показатель может быть эффективно использован, как минимум, в задачах классификации с ИНС прямого распространения с числом слоев 4 и более.

4. Заключение и дальнейшие исследования

В этой работе была разработана и исследована модель обучения искусственной нейронной сети как системы передачи информации. Для анализа степени искажений в процессе обучения предлагается использовать комплексный показатель, который можно охарактеризовать как коэффициент взаимного различия обучаемой и фактически полученной последовательностей. Эффективность предлагаемой модели основывается на применении метода сравнения энергетических характеристик сигналов в системах передачи данных [15, 21]. Таким образом, предложенная модель позволит решит задачу поиска глобального экстремума и повысить эффективность обучения ИНС.

Возможные направления дальнейших исследований:

1) разработка эффективного алгоритма изменения весовых показателей для обучения ИНС на основе информационной модели ИНС;
2) изучение особенностей поведения функции КВР при обучении различных ИНС и разработка на этой основе более эффективного оптимизатора, который позволит снизить время обучения.

Решение вышеуказанных задач позволит получить значительный выигрыш при обучении ИНС с учителем, избежать попадания в локальный минимум, а также глубже понять информационные процессы, происходящие при обучении ИНС со значительным количеством слоев. Кроме того, авторы выражают надежду, что подобный подход позволит снизить зависимость структуры ИНС от предметной области, т.к. предлагаемый метод позволяет абстрагироваться от специфических особенностей изучаемой последовательности и сосредоточиться на результатах.

Список литературы

[1]. Колмогоров А.Н. О представлении непрерывных функций нескольких переменных в виде суперпозиций непрерывных функций одного переменного и сложения, Доклады АН СССР, том 114, no. 5, 1957 г., стр. 953-956 / Kolmogorov A.N. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. American Mathematical Society Translations: Series 2, vol. 28, 1963, pp. 55-59.

[2]. Арнольд В.И. О представлении функций нескольких переменных в виде суперпозиции функций меньшего числа переменных. Математическое просвещение, вып. 3, 1958 г., стр. 41-61 / Arnol'd V.I. On the representation of functions of several variables as a superposition of functions of a smaller number of variables. In Vladimir I. Arnold - Collected Works, vol.1. Sringer, 2009, pp.

[3]. Hecht-Nielsen R. Neurocomputing. Addison-Wesley, 1989, 433 p.

[4]. Дзядык В.К. Введение в теорию равномерного приближения функций полиномами. М., Наука, 1977 г., 512 стр. / V.K. Dzyadyk. Introduction to the theory of the uniform approximation of functions by polynomials. Nauka, 1977, 512 p. (in Russian).

[5]. Hebb D.O. The Organization of Behavior. Wiley, 1949, 335 p.

[6]. Hinton G.E. Training Products of Experts by Minimizing Contrastive Divergence. Neural Computation, vol. 14, no. 8, 2002, pp.1771-1800.

[7]. Hinton G.E. Learning Multiple Layers of Representation. Trends in Cognitive Sciences, vol. 11, 2007, pp. 428-434.

[8]. Нужный А.С., Регуляризация Байеса при подборе весовых коэффициентов в ансамблях предикторов. Труды ИСП РАН, том 31, вып. 4, 2019 г., стр. 113-120 / Nuzhny A.S. Bayes regularization in the selection of weight coefficients in the predictor ensembles. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 4, 2019. pp. 113-120. DOI: 10.15514/ISPRAS-2019-31(4)-7 (in Russian).

[9]. García-Hernández L.E., Barrios-Hernande C.J., Radchenko G. et al. Multi-objective Configuration of a Secured Distributed Cloud Data Storage. Communications in Computer and Information Science, vol. 1087, 2019, pp. 78-93.

[10]. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. СПб., Питер, 2018 г., 480 стр. / Nikolenko S., Kadurin A., Arhangel'skaya E. Deep Learning. Piter, 2018, 480 p. (in Russian).

[11]. Дорогов А.Ю. Реализация спектральных преобразовании в классе быстрых нейронных сетей. Программирование, том 29, no. 4, 2003 г., стр. 13-26 / Dorogov A.Y. Implementation of spectral transformations in the class of fast neural networks. Programming and Computer Software, vol. 29, no. 4, 2003, pp. 187–198.

[12]. Аджемов С.С. и др. Использование искусственных нейронных сетей для классификации источников сигналов в системах когнитивного радио. Программирование, том 42, no. 3, 2016 г., стр. 3-11 / Adjemov S.S. et al. The use of artificial neural networks for classification of signal sources in cognitive radio systems //Programming and Computer Software, vol. 42, no. 3, 2016, pp. 121–128.

[13]. Vershkov N.A., Kuchukov V.A., Kuchukova N.N., Babenko M. The Wave Model of Artificial Neural Network. In Proc. of the 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, EIConRus 2020, pp. 542-547.

[14]. Шеннон К. Работы по теории информации и кибернетике. М., Издательство иностранной литературы, 1963 г., 830 стр. / Shannon C. Works on information theory and cybernetics. Izdatel'stvo inostrannoj literatury, 1963, 830 p. (in Russian).

[15]. Сикарев А.А., Лебедев О.Н. Микроэлектронные устройства формирования и обработки сложных сигналов. М.: Издательство «Радио и связь», 1983 г., 213 стр. / Sikarev A.A., Lebedev O.N. Microelectronic devices for the generation and processing of complex signals. Izdatel'stvo «Radio i svyaz'», 1983, 213 p. (in Russian).

[16]. Widrow B. Adaptive sampled–data systems, a statistical theory of adaptation. IRE WESCON Convention Record, vol. 4, 1959, pp. 74-85.

[17]. Айфичер Э.С., Джервис Б.У. Цифровая обработка сигналов: практический подход, 2-е издание. Пер. с англ. М., Издательский дом «Вильямс», 2008 г., 992 стр. / E.C. Ifeachor, B.W. Jervis. Digital signal processing: a practical approach. Pearson Education, 2002, 933 p.

[18]. А.В. Солодов. Теория информации и ее применение к задачам автоматического управления и контроля. М.: Издательство «Наука» Главная редакция физико-математической литературы, 1967 / A.V. Solodov, “Information theory and its application to tasks of automatic control and monitoring”, Nauka, 1967, (in Russian).

[19]. Ерофеева В.А. Обзор теории интеллектуального анализа данных на базе нейронных сетей, Стохастическая оптимизация в информатике, 2015 г., том 11, no. 3, стр. 3-17 / Erofeeva V.A. An Overview of Data Mining Concepts Based on Neural Networks. Stohasticheskaya optimizaciya v informatike, vol. 11, no. 3, 2015, pp. 3-17 (in Russian).

[20]. Цыпкин Я.З. Информационная теория идентификации. М., Наука. Физматлит, 1995 г., 336 стр. / Tsypkin Ya.Z. Information theory of identification. Nauka. Fizmatlit, 1995, 336 p. (in Russian).

[21]. Вершков Н.А., Кучуков В.А., Кучукова Н.Н. Теоретический подход к поиску глобального экстремума при обучении нейронных сетей. Труды Института системного программирования РАН, том 31, вып. 2, 2019 г., стр. 41-52 / Vershkov N.N., Kuchukov V.A., Kuchukova N.N. The theoretical approach to the search for a global extremum in the training of neural networks. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 2, 2019, pp. 41-52 (in Russian). DOI: 10.15514/ISPRAS-2019-31(2)-4.

[22]. Хайкин С. Нейронные сети: полный курс, 2-е издание. М., Издательский дом «Вильямс», 2006 г., 1104 стр. / Haykin S. Neural Networks: A Comprehensive Foundation. Prentice Hall, 1999, 842 p.

[23]. Линник Ю.В. Метод наименьших квадратов и основы математико-статистической теории обработки наблюдений. М., Физматгиз, 1958 г., 334 стр. / Linnik Yu.V. The method of least squares and the foundations of the mathematical-statistical theory of observation processing. M., Fizmatgiz, 1958, 334 p. (in Russian).

[24]. Рао Д., Макмахан Б. Знакомство с PyTorch: глубокое обучение при обработке естественного языка, Пер. с англ. Питер, 2020 г., 265 стр. / Rao D., McMahan B. Natural Language Processing with PyTorch: Build Intelligent Language Applications Using Deep Learning. O'Reilly Media, 2019, 256 p.

[25]. LeCun Y., Cortes C., Burges C.J.C. THE MNIST DATABASE of handwritten digits. Available at http://yann.lecun.com/exdb/mnist/, accessed 10.02.2020.
[26]. PyTorch. Available at https://pytorch.org/, accessed 10.11.2019.

Информация об авторах / Information about authors

Николай Анатольевич ВЕРШКОВ – кандидат технических наук. Сфера научных интересов: модулярная арифметика, нейрокомпьютерные технологии, цифровая обработка сигналов.

Nikolay Anatolievich VERSHKOV – Ph.D. in Engineering Sciences. His research interests include modular arithmetic, neurocomputer technologies, digital signal processing.

Михаил Григорьевич БАБЕНКО – кандидат физико-математических наук. Сфера научных интересов: облачные вычисления, высокопроизводительные вычисления, система остаточных классов, нейронные сети, криптография.

Mikhail Grigoryevich BABENKO – Ph.D. in Physics and Mathematics. His research interests include cloud computing, high-performance computing, residue number systems, neural networks, cryptography.

Виктор Андреевич КУЧУКОВ – младший научный сотрудник. Сфера научных интересов: высокопроизводительные вычисления, система остаточных классов, нейронные сети, цифровая обработка сигналов.

Viktor Andreevich KUCHUKOV – Research Assistant. His research interests include high-performance computing, residue number systems, neural networks, digital signal processing.

Наталья Николаевна КУЧУКОВА – ведущий специалист. Сфера научных интересов: система остаточных классов, нейронные сети, цифровая обработка сигналов.

Natalia Nikolaevna KUCHUKOVA - Leading Specialist. Her research interests include residue number systems, neural networks, digital signal processing.

DOI: 10.15514/ISPRAS–2021–33(1)–3

Изучение видов деятельности на основе машинного обучения в поведенческих контекстах Интернета вещей

[1] *М. Сафьян, ORCID: 0000-0003-4501-9699 <msafyan@gcul.edu.pk>*
[2] *С. Сарвар, ORCID: 0000-0001-9714-6580 <sohail.sarwar@seecs.edu.pk>*
[3] *З.У. Кайум, ORCID: 0000-0003-4230-6895 <zia@aiou.edu.pk>*
[2] *М. Икбал, ORCID: 0000-0002-8438-6726 <miqbal@lsbu.uk>*
[4] *С. Ли, ORCID: 0000-0001-5663-7420 <s.li@uwe.ac.uk>*
[5] *М. Кашиф, ORCID: 0000-0002-5640-9177 <miqbal@lsbu.uk>*

[1] *Правительственный университет колледжа,*
Пакистан, 54000, Пенджаб, Лахор
[2] *Лондонский университет Саут Бэнк,*
Великобритания, SE1 0AA, Лондон
[3] *Открытый университет Аллама Икбала,*
Пакистан, 44000, Исламабад,
[4] *Университет Западной Англии,*
Великобритания, BS16 1QY, Бристоль, Колдхарбор Лейн
[5]*Университет Озёгина,*
Турция, 34794, Стамбул, ул. Орман, 13

Аннотация: Модели изучения видов человеческой деятельности на основе онтологий играют жизненно важную роль в различных областях Интернета вещей, таких как умные дома, умные больницы и т.д. Основными проблемами онтологических моделей являются их статический характер и неспособность к самоэволюции. Модели нельзя разом построить полностью, и нельзя ограничить виды деятельности жителей умного дома. Кроме того, жители непредсказуемы по своей природе и могут выполнять «повседневную деятельность», не указанную в онтологической модели. Это порождает потребность в разработке интегрированной структуры, основанной на единой концептуальной основе (то есть онтологиях видов деятельности), обращаясь к жизненному циклу распознавания видов деятельности и создавая модели поведения в соответствии с распорядком дня жителей. В этой статье предложен процесс эволюции онтологии, в котором изучаются определенные виды деятельности из существующего набора видов деятельности в повседневной жизни. В этом процессе с помощью искусственной нейронной сети изучаются новые виды деятельности, которые не были идентифицированы моделью распознавания, добавляются новые свойства к существующим видам деятельности и выясняется новейшее поведение жителей при выполнении действий. Лучший уровень истинно-положительных срабатываний сети свидетельствует о распознавании видов деятельности с обнаружением искаженных сенсорных данных. Эффективность предложенного подхода очевидна из повышения скорости изучения видов деятельности, обнаружения видов деятельности и эволюции онтологии.

Ключевые слова: интернет вещей; распознавание деятельности; изучение вида деятельности; искусственная нейронная сеть

Для цитирования: Сафьян М., Сарвар С., Кайум З.У., Икбал М., Ли С., Кашиф М. Изучение видов деятельности на основе машинного обучения в поведенческих контекстах Интернета вещей. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 47-58. DOI: 10.15514/ISPRAS–2021–33(1)–3

Machine Learning based Activity learning for Behavioral Contexts in Internet of Things

[1] *M. Safyan, ORCID: 0000-0003-4501-9699 <msafyan@gcul.edu.pk>*
[2] *S. Sarwar, ORCID: 0000-0001-9714-6580 <sohail.sarwar@seecs.edu.pk>*
[3] *Z.U. Qayyum, ORCID: 0000-0003-4230-6895 <zia@aiou.edu.pk>*
[2]*M. Iqbal, ORCID: 0000-0002-8438-6726 <miqbal@lsbu.uk>*
[4] *S. Li, ORCID: 0000-0001-5663-7420 <s.li@uwe.ac.uk>*
[5] *M. Kashif, ORCID: 0000-0002-5640-9177 <miqbal@lsbu.uk>*

[1] *Lahore Government College University,*
Lahore, Punjab, 54000 Pakistan
[2] *London South Bank University,*
London, SE1 0AA England
[3] *Allama Iqbal Open University,*
Islamabad Capital Territory, 44000 Pakistan
[4] *University of West of England,*
Coldharbour Ln, Bristol, BS16 1QY UK
[5] *Özyeğin University,*
Orman Sk. no. 13, İstanbul, 34794 Turkey

Abstract. Ontology based activity learning models play a vital role in diverse fields of Internet of Things (IoT) such as smart homes, smart hospitals or smart communities etc. The prevalent challenges with ontological models are their static nature and inability of self-evolution. The models cannot be completed at once and smart home inhabitants cannot be restricted to limit their activities. Also, inhabitants are not predictable in nature and may perform "Activities of Daily Life (ADL)" not listed in ontological model. This gives rise to the need of developing an integrated framework based on unified conceptual backbone (i.e. activity ontologies), addressing the lifecycle of activity recognition and producing behavioral models according to inhabitant's routine. In this paper, an ontology evolution process has been proposed that learns particular activities from existing set of activities in daily life (ADL). It learns new activities that have not been identified by the recognition model, adds new properties with existing activities and learns inhabitant's newest behavior of performing activities through Artificial Neural Network (ANN). The better degree of true positivity is evidence of activity recognition with detection of noisy sensor data. Effectiveness of proposed approach is evident from improved rate of activity learning, activity detection and ontology evolution.

Keywords: Internet of Things; Activity Recognition; Activity Learning; Artificial Neural Networks

For citation: Safyan M., Sarwar S., Qayyum Z.U., Iqbal M., Li S., Kashif M. Machine Learning based Activity learning for Behavioral Contexts in Internet of Things. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 47-58 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–3.

1. Введение

Эпоха технологических революций ориентирована на человека, поскольку цель внедрения технологий – помочь людям. Одно из многообещающих приложений таких технологий – Интернет вещей (Internet pf Things, IoT). В соответствии с существующими тенденциями, распознавание видов деятельности человека (Activity Recognition, AR) и изучение видов деятельности (Activity Learning, AL) стали естественными механизмами внедрения адаптивных технологий IoT. AR/AL находятся в центре исследований в таких областях, как повсеместные и мобильные вычисления [1], организация вспомогательной жизненной среды

(ambient assisted living) [2], социальная робототехника [3], безопасность на основе видеонаблюдений (surveillance-based security) [4] и контекстно-зависимые вычисления [5].

Изучение видов деятельности [6] основано на распознавании видов деятельности в различных областях и средах. Распознавание видов деятельности основано на онтологических моделях. Чтобы создать онтологическую модель, необходимо проанализировать и обработать всю информацию и данные. Имеются два основных типа методов AR, лежащих в основе изучения видов деятельности: (а) на основе данных (data driven) и (б) на основе знаний (knowledge driven).

Методы, основанные на данных, используют данные сенсоров для разработки моделей видов деятельности с помощью алгоритмов машинного обучения и методов интеллектуального анализа данных. Эти методы позволяют справляться с искажениями, неопределенностью и неполнотой потока данных сенсора, но сталкиваются с проблемами, связанными с недостаточностью данных и отсутствием масштабируемости.

Методы, основанные на знаниях, используют предметные знания в требуемой области для создания моделей деятельности пользователей с помощью методов, основанных на знаниях, таких как управление знаниями и инженерия знаний. Главный недостаток методов, основанных на знаниях, заключается в том, что могут быть построены только статические модели видов деятельности [7]. Распознавание видов деятельности, основанное на знаниях, применимо к повседневной деятельности с определенными временной продолжительностью, местоположением и т.д. (как смоделировано в онтологии).

Онтологические модели удобны для распознавания активности, но проблема состоит в том, что онтологические модели статичны по своей природе и неспособны к самоэволюции. Для обновления этих онтологических моделей использовались нейронные сети для изучения различных видов деятельности при пошаговом выполнении действий.

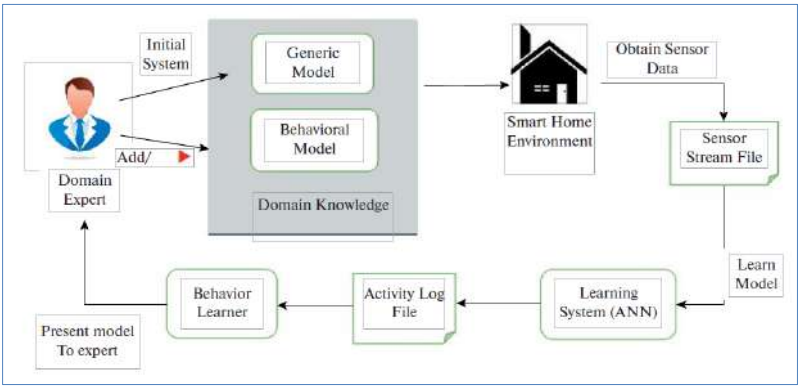


Рис. 1. Процесс активного обучения для получения поведенческой модели
Fig. 1. Activity Learning Process for having Behavioral Model

В этой статье предлагается непрерывный процесс моделирования видов деятельности, где эксперты предметной области предоставляют начальные обобщенные модели видов деятельности с использованием инструментов инженерии знаний [7]. Эти общие модели деятельности интегрируются в среду умного дома. На основе процесса распознавания деятельности создаются журналы деятельности. Для изучения модели поведения человека к содержимому журнала деятельности применяются интеллектуальные методы (искусственная нейронная сеть, Artificial Neural Network, ANN) [8]. Общий процесс изучения видов деятельности для получения поведенческой модели (также называемой полной моделью) представлен на рис.1.

При получении полной модели видов деятельности из обобщенных моделей (предоставленных экспертами предметной области) изучение видов деятельности является

ключевым процессом. Например, если у нас есть вид деятельности под названием «купание», то он состоит из таких видов деятельности, как «включитьДуш» и «взятьМыло».

Это минимальные действия, требуемые для того, чтобы помыться. С другой стороны, другой человек в этой деятельности может использовать шампунь или полотенце. В основе предлагаемого подхода лежит определение минимально необходимых действий (обобщенная модель) для выполнения некоторого вида деятельности, накопление данных, сгенерированных другими жителями, чтобы узнать о возможных других действиях, и получение поведенческой модели того же вида деятельности (с изменчивыми действиями).

Следовательно, система узнает о новых версиях старого вида деятельности. Это позволяет экспертам определять обобщенные виды деятельности на более высоких уровнях абстракции и затем разрабатывать поведенческую модель для создания специализированных знаний.

Оставшаяся часть статьи организована следующим образом: в разд. 2 дается краткий обзор методов распознавания видов деятельности и интерактивного обучения, в разд. 3 описывается предлагаемый фреймворк для активного обучения. Далее в разд. 4 приводятся результаты работы и их оценка. Разд. 5 посвящен итогам работы и потенциальным будущим направлениям исследований.

2. Обзор литературы

В этом разделе представлен краткий исследовательский обзор методов изучения и распознавания видов человеческой деятельности. В последнее время был проведен всесторонний анализ методов изучения видов деятельности наряду с методами моделирования сенсоров на основе машинного зрения и обучения, распознавания изображений и видео, а также мобильного распознавания активности с использованием сенсоров [32–35].

В работе [9] AR на основе данных комбинируется с методами на основе знаний для обработки неполных данных, поступающих от сенсоров. Подход, основанный на знаниях, для параллельной AR представлен Йе (Juan Ye) и др. в [10]. В этом подходе исследуется контекст активации сенсора и используется различие контекстов для кластеризации непрерывной последовательности данных сенсора для AR.

Рибони (Daniele Riboni) и др. [11] предложили фреймворк для представления сенсоров, устройств, активностей и атомарных действий. В их подходе глубокое обучение (Deep Learning, DL) сочетается с вероятностным мышлением (probabilistic thinking). Используемые правила оценки правдоподобия не определяются в семантике и задаются вручную. Предлагаемый подход с использованием DL является статическим по своей. Он позволяет распознавать активности, но не обеспечивает возможности выявления персонализированного поведения человека.

Окейо (George Okeyo) и др. [12] комбинируют формализмы онтологического и темпорального знаний, чтобы обеспечить представление для моделирования составных активностей. В этой статье также описаны правила, позволяющие динамически выводить составные действия. Простые активности, моделируеые в этой статье, по своей природе статичны. Наша работа отличается от [12] [13] двумя аспектами. Во-первых, в нашем случае генерируется полная модель видов деятельности на основе обобщенной модели. Во-вторых, динамически распознаются временные интервалы активностей.

Обширное исследование было выполнено, когда произошла смена парадигм от традиционных систем к умным мобильным устройствам [15][16]. Современные технологии носимых сенсоров включают в себя смартфоны, смарт-кольца, сенсоры на одежде, медицинские сенсоры и специализированные сенсоры, прикрепляемые к различным частям тела.

Окна динамически изменяемого размера для сегментации потоков сенсорных данных используются в [14] и [25]. Вводится механизм распознавания непрерывной активности в

реальном времени с использованием онтологических знаний для последовательных действий.

В обучение с учителем используются размеченные данные для обучения алгоритма, который может систематизировать неразмеченные данные [23]. При использовании для изучения видов деятельности обучение с учителем имеет некоторые особенности, такие как представление данных, преобразование данных из нескольких источников, разделение данных на обучающие наборы, наборы тестов для обучения модели.

Другие методы AR включают скрытые марковские модели [24], наивные байесовские сети [26][27], деревья решений [28], машины опорных векторов [30], [31], [33] и метод K-ближайших соседей [29], [30]. Все эти алгоритмы были бы достаточно хороши, если бы поведение людей было известно заранее.

3. Предлагаемый подход

Здесь подробно описывается предложенная схема, которая изучает поведенческую модель на основе общенной модели, как показано на рис. 2. Моделирование активности на основе онтологий обеспечивается с определенными ограничениями, предполагающими, что полная и обобщенная модели видов деятельности не возможны одновременно.

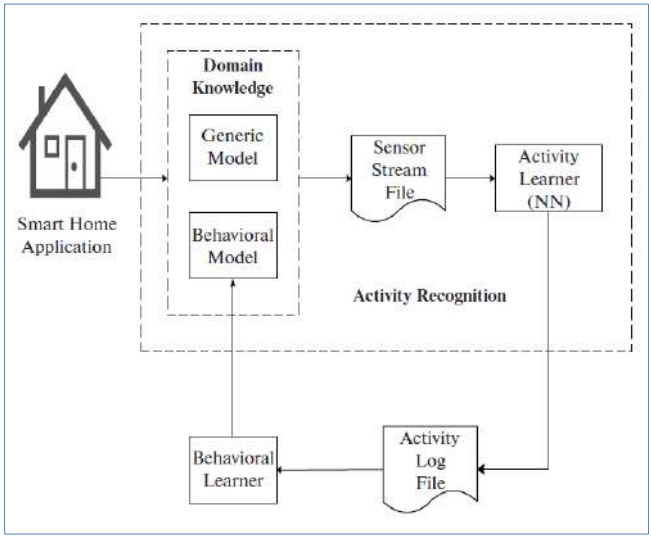


Рис 2: Предлагаемая схема для распознавания активности и обучения
Fig. 2. Proposed Framework for Activity recognition and Learning

Как только информация о человеке из источников (в нашем случае от сенсоров) становится доступной, следующий шаг – представление этих знаний. В зависимости от собранной информации были разработаны два типа моделей активности: обобщенная модель (онтологическая модель) и поведенческая модель (JSON-файл).

3.1. Общая модель (онтологическая модель)

Обобщенная модель видов деятельности может быть представлена через концепты онтологии в виде троек (субъект, предикат и объект). Например, чистка зубов – это вид деятельности, которая выполняется два раза в день, утром и перед сном. Как правило, это деятельность предполагает использование зубной щетки и воды. Обычно это называется основой вида деятельности. Активности определяются как онтологические концепты, а все действия, которые необходимы для завершения активности, как характеристики концепта.

3.2. Поведенческая модель (JSON-файл)

Разработана специализированная поведенческая модель (на основе обобщенной модели видов деятельности). Поведенческие модели развиваются на протяжении процесса изучения видов деятельности. Общие модели представляются файлами онтологий, а поведенческие модели – в виде файлов в формате JSON.

Для распознавания активности был разработан и реализован двухшаговый алгоритм. Первый этап, называемый шагом отображения сенсорных данных в действия, использует информацию о сенсоре из контекстных знаний для преобразования данных активизации сенсоров в действия. На втором этапе, этапе поиска активности, запускается алгоритм распознавания образов с использованием обобщенных и поведенческих моделей. Процесс поясняется на лисинге 3.1.

```
Input: sensor_activation_dataset, domain_knowledge
Output: annotated_dataset
        action_dataset←applyTransformFunction(sensor_activation_dataset,
                                              domain_knowledge)
GAM list ← obtainGAM (domain_knowledge)
for all action ∈ action_dataset do
 if action ∈ BAM then
        activities←obtainActivities(action, BAM list)
     end if
     if action ∈ GAM then
        activities ← obtainActivities(action, GAM list)
     end if
     for all activity ∈ activities do
// Use duration, completion, location criteria
        annotated_dataset←
        findValidActivities(context knowledge)
   end for
end for
return annotated_dataset
```

Листинг 3.1. Алгоритм распознавания активности
Listing 3.1. Algorithm for Activity Recognition

3.3. Отображение действий сенсора

После каждой активации сенсора входные данные из набора данных активации принимаются и преобразуются в знания предметной области. Для каждой активации сенсора проверяется соответствующая модель сенсора в знаниях предметной области. Для каждого сенсора имеется действие, которому он должен быть сопоставлен в файлах знаний предметной области, предоставленных экспертом в предметной области.

Для нашего примера с купанием входные данные от сенсора {shower_sens1, soap_sens1} будут преобразованы в {turn_on_shower(shower), has_soap(soap)}{вкл_душ(душ), иметь_мыло(мыло)}.

3.5. Средство для изучения видов деятельности

В этом подразделе мы описываем и анализируем предлагаемый алгоритм обучения специализированных и поведенческих моделей активности, который мы называем средством для изучения видов деятельности (Activity Learner, AL) (рис. 3).

AL использует результаты из файла Sensor Stream File (SSF), в котором различные последовательности действий для каждого вида деятельности идентифицируются в процессе распознавания активности. Цель AL – научить поведенческую модель видов деятельности на

основе информации, предоставленной SSF. Для обучения используется искусственная нейронная сеть.

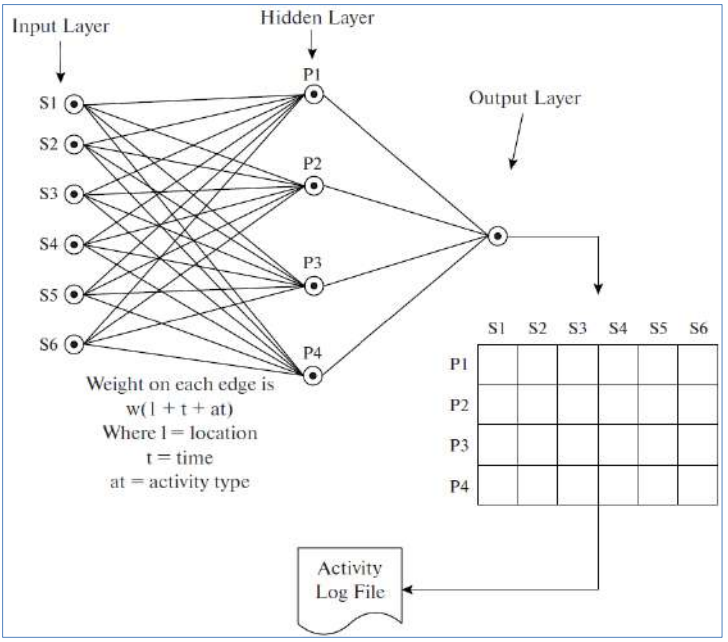


Рис. 3. Модель ANN для изучения активностей
Fig 3. ANN Model for Activity Learning

ANN состоит из взаимосвязанных вершин и взвешенных связей. Многие архитектуры ANN разрабатывались с учетом потребностей различных реальных областей распознавания активности. Архитектура, используемая в предлагаемой работе, представляет собой многослойный перцептрон (Multi-Layer Perceptron, MLP). MLP состоит из входного слоя, скрытого слоя и выходного слоя, где каждый слой состоит из одной или нескольких вершин, представленных на рис. 3 маленькими кружками.

Входной слой получает данные с сенсоров, которые не аннотируются в SSF. Каждый сенсор действует как узел во входном слое. Скрытый слой содержит информацию об обобщенной модели, аннотированную в SSF. Чтобы понять этот сценарий, можно рассмотреть следующий поток сенсорных данных:

```
<2017-09-06T10:23:06 21,
2017-09-06T10:23:11 19,4>
2017-09-06T10:22:56 20,
2017-09-06T10:23:00 18,
2017-09-06T10:23:08 20,
<2017-09-06T10:24:30 25,
2017-09-06T10:24:37 18,7>
2017-09-06T10:24:32 11,
2017-09-06T10:24:59 9,
2017-09-06T10:25:01 26.
```

Сенсоры, не описываемые обобщенной моделью, могут рассматриваться как дополнительные сенсоры или могут быть сенсорами, генерирующими ложные данные. Эти сенсоры отображаются на входной слой. Ребра между входным и скрытым слоями описывают информационный поток. Каждое ребро между входным и скрытым слоями имеет вес, который равен

$$w(l + t + at), \tag{3.1}$$

где l – местоположение, $t -$ время и $at -$ тип активности.

Значение времени t в формуле (3.1) будет 0 или 1 в зависимости от следующего. Пусть $t1$ – время начала активности, $t2$ – время окончания активности, а ts – время активации сенсора. Если значение ts находится между $t1$ и $t2$, т.е.

$$t1 \leq ts \leq t2, \tag{3.2}$$

то значение t будет равно 1, иначе оно будет равно 0.

Значение на выходном уровне сети вычисляется следующим образом:

$$y = \frac{l}{Pl} + \frac{at}{P(at)} + t. \tag{3.3}$$

В (3.3) через l обозначается местоположение входного сенсора, Pl – местоположение выполняемой активности, at – тип активности входного сенсора, $P(at)$ – тип выполняемой активности и t такое же, как выше.

Существует также выходная таблица, которая содержит выходную запись всех вычислений для входного сенсора.

Положительный шум сенсора может быть рассчитан на выходном слое как

$$Noise = T - y, \tag{3.4}$$

где T – заданный выходной параметр каждого сенсора. В нашем сценарии его значение равно 3.

После расчета по всем скрытым и выходным слоям выходная таблица будет заполнена расчетными результатами. Эта выходная таблица содержит все случаи с дополнительными сенсорами, а также сенсорами шума с их соответствующими обобщенными и поведенческими моделями.

4. Результаты и их оценка

Активности повседневной жизни (Activity of Daily Life, ADL) включают набор видов деятельности, которую каждый человек выполняет день ото дня. Некоторые примеры ADL – это заваривание чая, приготовление макарон или стирка одежды. Алгоритм AL применяется к набору данных из SSF. Все строки дополнительных сенсоров отображаются на входном слое, а аннотированные строки – на скрытом слое.

Один из наиболее сложных аспектов заключался в получении исчерпывающего набора данных, охватывающего все обсуждаемые сценарии и пригодного для оценки предлагаемой модели. В нашем исследовании использовался генератор синтетических наборов данных [35].

Если на этапе распознавания активности сенсор, сопоставленный с обобщенной моделью (подраздел 3.2), не активируется, предлагаемая система не может распознать эту активность. Для проверки этого сценария были выбраны две из семи активностей. Чтобы увидеть, как влияет отсутствие данных об активации сенсора на обнаружение соответствующей активности, для этого сенсора назначалась повышенная вероятность сбоя.

Для остальных активностей шум отказа сенсора отсутствовал. В табл. 1 показаны результаты для сценария отсутствия отказов сенсоров. Табл. 2 соответствует сценарию, когда данные об активации некоторых сенсоров могут теряться. Действия, для которых это возможно, помечены звездочкой (СделатьЧай и СделатьКофе). Как видно из табл. 2, производительность предложенной системы для всех остальных активностей остается прежней. Остаются 100% истинно положительных результатов, в то время как количество ложноположительных и ложноотрицательных результатов значительно сокращается.

Уменьшение количества истинно положительных результатов приводит к появлению ложноотрицательных результатов, поскольку не выявленные действия помечаются как шум. Таким образом, они считаются ложноотрицательными. Не оказывается никакого эффекта на

ложноположительные активности, однако, как и ожидалось, воздействие на истинно-положительные очевидно.

Табл. 1. Результаты для идеального сценария
Table 1. Result for the Ideal Scenario

Виды деятельности	**Число активностей**	**Истинно положительные (%)**	**Ложно-положительные (%)**	**Ложно-отрицательные (%)**
Принимать душ	49	100	0	0
Делать чай	56	100	0	0
Смотреть телевизор	77	100	0	0
Стирать вещи	21	100	0	0
Делать кофе	14	100	0	0
Чистить зубы	56	100	0	0

Табл. 2. Результаты сценария при наличии сбоев сенсоров
Table 2. Results of Sensor Missing Noise Scenario

Виды деятельности	**Число активностей**	**Истинно положительные (%)**	**Ложно-положительные (%)**	**Ложно-отрицательные (%)**
Принимать душ	49	100	0	0
Делать чай*	56	88.5	0	11.5
Смотреть телевизор	77	100	0	0
Стирать вещи	21	100	0	0
Делать кофе*	14	90	0	10
Чистить зубы	56	100	0	0

5. Заключение и дальнейшая работа

В настоящей статье была предложена схема обучения деятельности, основанная на онтологии. Реализация выполняется с помощью аннотации данных, распознавания активности, обучение активности на основе ИНС и аннотированного набора данных сенсора. В результате процесса обучения деятельности были созданы модели поведенческой активности для персонализированных моделей активности жителей умных домов.

Подход к распознаванию и обучению активности основывается на взаимодействии объектов, за которыми наблюдают сенсоры. Одним из ограничений этого подхода является взаимная исключительность обобщенных действий, то есть дополнительные сенсоры могут быть частью нескольких поведенческих действий. Мы надеемся устранить это ограничение с помощью дополнительного сенсора в рамках его точной обобщенной модели активности.

Список литературы / References

[1]. Choudhury T., Consolvo S., Harrison B. et al. The mobile sensing platform: An embedded activity recognition system. IEEE Pervasive Computing, vol. 7, no. 2, 2008, pp. 32-41.

[2]. Philipose M., Fishkin K.P., Perkowitz M. et al. (2004). Inferring activities from interactions with objects. IEEE Pervasive Computing, vol. 3, no. 4, 2004, pp. 50-57.

[3]. Safyan M., Qayyum Z.U., Sarwar S., Iqbal M., Ahmed M. Context-Aware Personalized Activity Modeling in Concurrent Environment. In Proc. of the 10th International Conference on Internet of Things (iThings-17), 2017, pp 977-982.
[4]. Fernández-Caballero A., Castillo J.C., Rodríguez-Sánchez J.M. Human activity monitoring by local and global finite state machines. Expert Systems with Applications, vol 39, no. 8, 2012, pp 6982-6993.
[5]. Chervyakov N., Babenko M., Tchernykh A. et al. AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security. Future Generation Computer Systems, vol. 92, 2019, pp. 1080-1092.
[6]. Chen L., Nugent C., & Okeyo G. An ontology-based hybrid approach to activity modeling for smart homes. IEEE Transactions on Human-Machine Systems, vol. 44, no. 1, 2014, pp. 92-105.
[7]. Poppe R. A survey on vision-based human action recognition. Image and Vision Computing, vol. 28, no. 6, 2010, pp. 976-990.
[8]. Weinland D., Ronfard R., & Boyer E. A survey of vision-based methods for action representation, segmentation and recognition. Computer Vision and Image Understanding, vol. 115, no. 2, 2011, pp. 224-241.
[9]. Azkune G., Almeida A., López-de-Ipiña D., & Chen L. Extending knowledge-driven activity models through data-driven learning techniques. Expert Systems with Applications, vol. 42, no. 6, 2015, pp. 3115-3128.
[10]. Ye J., Stevenson G., & Dobson S. KCAR: A knowledge-driven approach for concurrent activity recognition. Pervasive and Mobile Computing, vol. 19, 2015, pp. 47-70.
[11]. Riboni D., Sztyler T., Civitarese G., & Stuckenschmidt H. Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning. In Proc. of the ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2016, pp. 1-12
[12]. Okeyo G., Chen L., & Wang H. Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes. Future Generation Computer Systems, vol. 39, 2014, pp. 29-43.
[13]. Chen L., Nugent C.D., & Wang H. A knowledge-driven approach to activity recognition in smart homes. IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 6, 2012, pp. 961-974.
[14]. Azkune G., Almeida A., López-de-Ipiña D., & Chen L. (2015). Extending knowledge-driven activity models through data-driven learning techniques. Expert Systems with Applications, vol. 42, no. 6, 2015, pp. 3115-3128.
[15]. Chen L, Nugent C.D., and Wang H. A Knowledge-Driven Approach to Activity Recognition in Smart Homes. IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 6, 2012, pp. 961-974.
[16]. Gellersen H.W., Schmidt A., & Beigl. M. Multi-sensor context-awareness in mobile devices and smart artifacts. Mobile Networks and Applications, vol. 7, no. 5, 2002, pp. 341-351.
[17]. Buettner M., Prasad R., Philipose M., & Wetherall D. Recognizing daily activities with RFID-based sensors. In Proc. of the 11th International Conference on Ubiquitous Computing, 2009, pp. 51-60.
[18]. Gellersen H.W., Schmidt A., & Beigl M. (2002). Multi-sensor context-awareness in mobile devices and smart artifacts. Mobile Networks and Applications, vol. 7, no. 5, 2002, pp. 341-351.
[19]. Ravi N., Dandekar N., Mysore P., & Littman M.L. (2005, July). Activity recognition from accelerometer data. In Proc. of the Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, 2005, pp. 1541-1546.
[20]. Brezmes T., Gorricho J.L., & Cotrina J. Activity recognition from accelerometer data on a mobile phone. Lecture Notes in Computer Science, vol. 5518, 2009, pp. 796-799.
[21]. Dayan P. Unsupervised learning. In Wilson R.A. & Keil F., editors. The MIT Encyclopedia of the Cognitive Sciences. MIT, 1999, 7 p.
[22]. Chen L., Nugent C. Ontology-based activity recognition in intelligent pervasive environments. International Journal of Web Information Systems, vol. 5, no. 4, 2017, pp. 410-430.
[23]. Liao L., Fox D., & Kautz H. Extracting places and activities from gps traces using hierarchical conditional random fields. The International Journal of Robotics Research, vol. 26, no. 1, 2007, pp. 119-134.
[24]. Tapia E.M., Choudhury T., & Philipose M. Building reliable activity models using hierarchical shrinkage and mined ontology. Lecture Notes in Computer Science, vol. 3968, 2006, pp. 17-32
[25]. Yamada N., Sakamoto K., Kunito G. et al. Applying ontology and probabilistic model to human activity recognition from surrounding things. IPSJ Digital Courier, vol. 3, 2007, pp. 506-517.

[26]. Safyan M., Qayyum Z.U., Sarwar S., Iqbal M., Ahmed M. Ontology-driven Semantic Unified Modeling for Concurrent Activity Recognition (OSCAR). International Journal of Multimedia Tools and Applications, vol. 78, 2019, pp. 2073–2104.

[27]. Fong T., Nourbakhsh I., and Dautenhahn K. A survey of socially interactive robots. Journal of Robotics and Autonomous Systems, Vol 42, no. 2, 2003, pp. 143-166.

[28]. Latfi F., Lefebvre B., & Deschen C. (2017). Ontology-Based Management of the Telehealth Smart Home, Dedicated to Elderly in Loss of Cognitive Autonomy. In Proc. of the OWLED: Workshop on OWL: Experiences and Directions, 2027, pp. 3-16.

[29]. Klein M., Schmidt A., & Lauer R. (2007). Ontology-centered design of an ambient middleware for assisted living: The case of soprano. In Proc. of the 30th Annual German Conference on Artificial Intelligence KI, 2007, pp. 2-8.

[30]. Chen L., Nugent C., Mulvenna M., Finlay D., & Hong X. Semantic smart homes: towards knowledge rich assisted living environments. Studies in Computational Intelligence, vol 189, 2009, pp. 279-296.

[31]. Машечкин И.В., Петровский М.И., Царев Д.В., Чикунов М.Н. Методы машинного обучения для задачи обнаружения и мониторинга экстремистской информации в сети интернет. Программирование, том 45, no. 3, 2019 г., стр. 18-37 / Mashechkin I.V., Petrovskiy M.I., Tsarev D.V., Chikunov M.N. Machine Learning Methods for Detecting and Monitoring Extremist Information on the Internet. Programming and Computer Software, vol. 45, no. 3, 2019, pp. 99–115.

[32]. Сарвар С., Кайум З.У, Сафьян М., Икбал М., Махмуд Я. Выявление характерных особенностей программ для борьбы с компьютерным пиратством на основе интеллектуального анализа графов. Труды ИСП РАН, том 31, вып. 2, 2019 г., стр. 171-186. DOI: 10.15514/ISPRAS-2019-31(2)-12 / Sarwar S., Qayyum Z.U., Safyan M., Iqbal M., Mahmood Y. (2019). Graphs Resemblance based Software Birthmarks through Data Mining for Piracy Control. Programming and Computer Software, vol 45, no. 8, 2019, pp. 581–589.

[33]. Бабенко М.Г., Черных А.Н., Червяков Н.И. и др. Эффективное сравнение чисел в системе остаточных классов на основе позиционной характеристики. Труды ИСП РАН, том 31, вып. 2, 2019 г., стр. 187-202. DOI: 10.15514/ISPRAS-2019-31(2)-13 / Babenko M., Tchernykh A., Chervyakov N. et al. Positional Characteristics for Efficient Number Comparison over the Homomorphic Encryption. Programming and Computer Software, vol 45, no. 8, 2019, pp 532–543.

[34]. Augusto J.C. & Nugent C.D. (2004). The use of temporal reasoning and management of complex events in smart homes. In Proc. of the 16th European Conference on Artificial Intelligence, 2004, pp. 778-782.

[35]. García-Hernández L.E., Tchernykh A., Miranda-López V., Babenko M. Multi-objective Configuration of a Secured Distributed Cloud Data Storage. In Proc. of the Latin American High Performance Computing Conference, 2019, pp. 78-93.

Информация об авторах / Information about authors

Мухаммад САФЬЯН – кандидат наук, доцент факультета компьютерных наук. Область его научных интересов включает отображение онтологий, электронное обучение, семантическое распознавание активностей.

Muhammad SAFYAN, PhD, Associated Professor of the Department of Computer Science. His area of interest includes ontology alignment, e-learning and semantic activity recognition.

Сохаил САРВАР – кандидат наук, научный сотрудник. Научные интересы: машинное обучение, Интернет вещей, поиск информации.

Sohail SARWAR, PhD, Research Fellow. Scientific interests: machine learning, IoT, information retrieval.

Зия УЛ КАЙЮМ – кандидат наук, профессор, проректор. Его исследовательские интересы включают искусственный интеллект, инженерию знаний, интеллектуальный анализ данных, семантическую сеть и электронное обучение.

Zia UL QAYYUM, PhD, Professor, Vice Chancellor. His research interests include artificial intelligence, knowledge engineering, data mining, semantic web and e-learning.

Муддессар ИКБАЛ – кандидат наук, старший преподаватель. Его исследовательские интересы включают сетевые технологии 5G, мультимедийные облачные вычисления, мобильные периферийные вычисления, туманные вычисления, Интернет вещей, программно-определяемые сети, виртуализацию сетевых функций, качество взаимодействия, а также облачные инфраструктуры и услуги.

Muddessar IQBAL – PhD, Senior Lecturer. His research interests include 5G networking technologies, multimedia cloud computing, mobile edge computing, fog computing, Internet of Things, software-defined networking, network function virtualisation, quality of experience, and cloud infrastructures and services.

Шанцан ЛИ – кандидат наук, старший преподаватель. Его текущие исследовательские интересы включают в себя сетевую экспертизу, безопасность устройств, беспроводные сенсорные сети, Интернет вещей и облегченную криптографию в IoT.

Shancang LI, PhD, Senior Lecturer. His current research interests include network forensics, device security, wireless sensor networks, Internet of Things, and lightweight cryptography over IoT.

Мухаммад КАШИФ – кандидат наук, научный сотрудник. Область научных интересов: машинное обучение, блокчейн, Интернет вещей.

Muhammad KASHIF, PhD, Researcher. Research interests: Machine Learning, BlockChain, IoT.

DOI: 10.15514/ISPRAS–2021–33(1)–4

Интеллектуальный метод автоматического отслеживания объектов путем интеграции лазерного сканирования и инерциальной навигации

Х.С. Родригес-Киньонес, ORCID: 0000-0002-1830-0226 <julio.rodriguez81@uabc.edu.mx>
Автономный университет Нижней Калифорнии (UABC),
Мексика, 21100, Нижняя Калифорния, Энсенада

Аннотация. За последние годы автономная мобильная навигация была значительные усовершенствована; однако эта технология все еще не может быть эффективно интегрирована с мобильными системами, которые функционируют внутри производственных зданий и жилых домов, а также без абсолютной привязки к местности, обеспечиваемой, например, GPS. По этой причине в настоящей статье представлена методика, позволяющая интегрировать методы обнаружения объектов, такие как YOLO или R-CNN, с системами лазерного сканирования для измерения пространственных координат и инерциальной навигационной системой дли пространственного позиционирования путем привязки к позициям других объектов. В нашем подходе объединяется принцип динамической триангуляции и методика IKZ, которые позволяют получить приемлемое время автономного функционирования мобильного устройства перед тем, как потребуется повторное сканирование окружающих объектов.

Ключевые слова: инерциальные навигационные системы; IKZ; сверточные нейронные сети

Для цитирования: Родригес-Киньонес Х.С. Интеллектуальный метод автоматического отслеживания объектов путем интеграции лазерного сканирования и инерциальной навигации. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 59-64. DOI: 10.15514/ISPRAS–2021–33(1)–4

Intelligent Automatic Object Tracking Method by Integration of Laser Scanner System and INS

J.C. Rodríguez-Quiñonez, ORCID: 0000-0002-1830-0226 <julio.rodriguez81@uabc.edu.mx>
Universidad Autónoma de Baja California,
Ensenada, Mexico, 21100

Abstract. In the last years, autonomous vehicle navigation has had considerable improvements; however, this technology has not yet been able to effectively be integrated with mobile systems that work inside buildings, houses or without the influence of absolute references such as GPS. It is for this reason that this paper provides a methodological proposal that allows the integration of the use of object detection methods such as YOLO (You Only Look Once) or R-CNN (Region Convolutional Neural Networks), with laser scanning systems for the measurement of spatial coordinates and an INS system to obtain spatial positioning through relative references. This method integrates the principle of dynamic triangulation and the methodological proposal IKZ (Inertial Navigation with Kalman and Zero Update) that allow to obtain acceptable mobile autonomy times in which the sector of interest is re-scanned.

Keywords: inertial navigation systems; IKZ; convolutional neural networks

For citation: Rodríguez-Quiñonez J.C. Intelligent Automatic Object Tracking Method by Integration of Laser Scanner System and INS. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 59-64 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–4.

1. Введение

Поиск и отслеживание являются важными задачами для мобильного робота, например, для помощи людям, поиска объектов, автоматической транспортировки и выполнения аварийно-спасательных работ. Использование лазерных сканеров является одним из эффективных подходов к получению пространственных координат отслеживаемого объекта, и для этого полезно применять системы технического зрения (Technical Vision System, TVS) с динамической триангуляцией [1]. Тем не менее, одних этих систем недостаточно для идентификации объектов, представляющих интерес.

Одним из эффективных подходов к решению такого рода задач является использование сверточных нейронных сетей (Convolutional Neural Networks, CNN). В последнее время CNN продемонстрировали отличные результаты в различных областях: распознавание образов, классификация и сегментация изображений [2], [3]. В том числе, CNN успешно используются и в системах трекинга, обеспечивая высокую точность отслеживания.

Однако в последние годы для решения задач обнаружения наиболее часто используется метод YOLO (You Only Look Once) [5]. В нашей работе в качестве метода обнаружения используется именно YOLO, однако предлагаемый подход может быть реализован с использованием и других методов обнаружения объектов.

Вместе с тем, в большинстве современных мобильных систем слежения берутся в расчет только обнаружение интересующего объекта и следование за ним без планирования пути. Эффективным подходом к обеспечению этой недостающей возможности является применение инерциальных навигационных систем (Inertial Navigation System, INS), в которых изменения показаний на коротких промежутках времени датчиков должны быть незначительными, а интересующий объект может использоваться в качестве абсолютного ориентира. С целью получения эффективного метода поиска и слежения предлагается методика интеграции TVS, обнаружения объектов и INS.

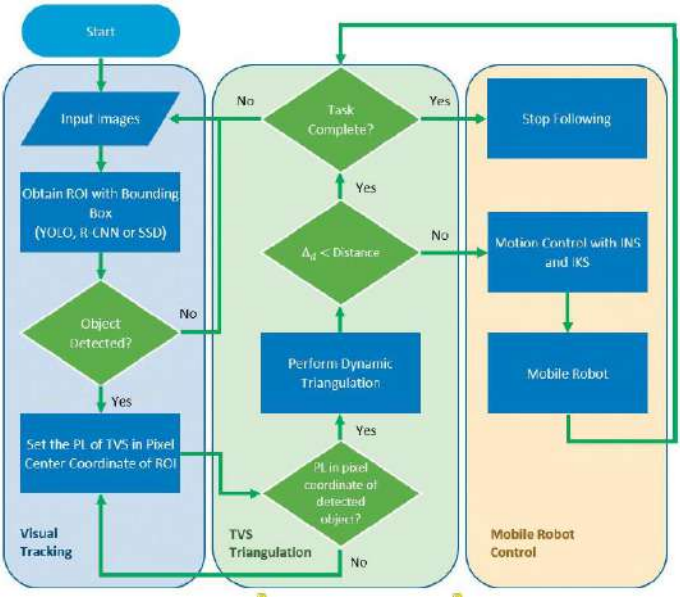


Рис 1. Блок-схема предлагаемого метода
Fig 1. Flow Diagram of the proposed Method

2. Метод

В нашей работе для создания мобильного робота, способного отслеживать определенные объекты или людей, сводится воедино функционирование трех датчиков. Первый датчик – это камера, которая предоставляет изображения RGB для определения местоположения интересующего объекта в зоне съемки. Второй датчик представляет собой систему лазерного сканирования, которая обеспечивает информацию об относительных координатах интересующего объекта, а также информацию о навигационной среде. Третьим датчиком является IMU (Inertial Measurement Unit, инерциальный измерительный блок) с акселерометром, гироскопом и магнитометром, используемый в INS (инерциальная навигационная система) в сочетании с IKZ (Inertial navigation with Kalman and Zero Update, инерциальная навигация с фильтром Калмана и нулевым обновлением) [4].

Такая схема организации системы позволяет определять местонахождение объекта в режиме реального времени, определять расстояние до него, передавать навигационное задание в систему INS-IKZ, что позволяет обнаруживать препятствия с помощью TVS. Рис. 1 демонстрирует, как происходит интеграция информации и принятие решения.

3. Моделирование

Чтобы определить, как себя поведет предложенный подход, необходимо промоделировать различные сценарии работы, для чего учитываются следующие параметры: скорость обнаружения объекта, скорость сканирования с разными угловыми шагами и автономность системы IKZ в секундах. Скорость обнаружения объекта с помощью YOLO V3 может быть оценена по табл. 1, а время позиционирования лазерного сканера с использованием принципа динамической триангуляции – по табл. 2.

Табл. 1. Скорость обнаружения объекта с использованием YOLO V3 [5]
Table 1. Object detection velocity using YOLO V3 [5]

Размер изображения	Yolo V3
320x320 pix	22 мс
416 x 416 px	29 мс
608 x 608 px	51 мс

Табл. 2. Скорость позиционирования лазера при различных угловых перемещениях [6]
Table 2. Positioning Laser Speed at different angular displacements [6]

Угловое перемещение	Yolo V3
2°	10.93мс
5°	52.3мс

Чтобы получить не менее 11 точек на метр, что достаточно для мобильного робота, описанного в работе Басака-Пресиадо (Luis Carlos Básaca-Preciado) и др. [7], при позиционировании лазера предлагается использовать разрешение 5° и 2°. Для демонстрации работы системы в целом представлен алгоритм (листинг 1), описывающий взаимодействие методов обнаружения объектов, триангуляции и управления мобильным роботом.

```
while task is not complete do
 do
  Input Image_Scene
  Call: Position = Yolo (Roi)
 while (Object is not detected)
 Set PL to Position
 Call: distance = Triangulation ()
 if distance is greater than delta then
  do
```

```
    move Mobile to Position
    calculate current_Position = IKZ ()
  while (abs (Position – current_Position) is greater than delta)
 end
end
```

Листинг 1. Взаимодействие систем
Listing 1. Interaction of the systems

Следует отметить, что в каждом методе используется группа датчиков со своим собственным временем отклика (например, в триангуляционных системах применяются позиционирующий лазер и сканирующая апертура, а для систем INS требуется постоянный мониторинг на основе акселерометров и гироскопов), и реализации каждой из этих систем присущи свои собственные проблемы, которые обсуждались в [4], [7-11]. Однако для интеграции всего этого в единую систему, которая позволяет мобильному устройству двигаться к цели, требуется знание различных сценариев.

Было рассмотрено 105 сценариев с использованием различных возможных скоростей из табл. 1, различных углов позиционирования из табл. 2 и различных ситуаций позиционирования, когда система перемещается между 2° и 55° в поле зрения и время сканирования составляет 295-576 мс (в зависимости от величины шага сканирования). Упрощенная таблица этого моделирования показана в табл. 3.

Табл. 3. Упрощенная таблица различных временных сценариев совместного использования обнаружения объектов и сканирования секторов
Table 3. Simplified table of different time scenarios of conjunction use of object detection and sector scanning

Метод	**ΔYolo (мс)**	**Шаг позиционирования лазера**	**Время шага (мс)**	**Число шагов**	**Время (мс)**
YoloV3-320	22	2°	10.93	1	327.93
YoloV3-416	29	2°	10.93	1	334.93
YoloV3-608	51	2°	10.93	1	356.93
⋮	⋮	⋮	⋮	⋮	⋮
YoloV3-608	51	5°	52.4	5	889
YoloV3-320	22	5°	52.4	6	912.4

Как показывает табл. 3, время выполнения как примерно 300 – 1000 мс, и ниже или выше этого диапазона находятся только несколько исключений; гистограмма частот для всех 105 случаев представлена в таблице 4.

Табл. 4. Частотная гистограмма времен обнаружения и сканирования для 105 сценариев
Table 4. Frequency histogram of detection and scanning time for the 105 scenarios

Время в мс	**Частота**
350	5
470	32
590	33
710	16
830	6
950	7
1070	7
1190	7
И свыше...	1

Как упоминалось ранее, только несколько результатов превышают 1000мс, однако это время не является критическим фактором, влияющим на автономность мобильного робота,

поскольку IKZ может точно определять положение мобильного устройства с интервалами до 25 секунд до того, как потребуются абсолютные координаты [4].

4. Заключение

В настоящей статье представлено предложение по интеграции системы обнаружения объектов, системы лазерного сканирования и инерциальной навигационной системы для отслеживания объектов мобильным роботом. В качестве метода обнаружения объектов предлагается использовать YOLO, однако может использоваться разновидность CNN или какой-либо другой метод обнаружения объектов, если его можно применять в реальном времени (или на скорости более 20 кадров в секунду).

Предлагаемая методика чередует обнаружение объектов, сканирование сцены и систему IKZ для определения текущего положения мобильного объекта. Используемый метод IKZ имеет время автономной работы около 20 секунд до появления ухода гироскопа, поэтому за время обнаружения и сканирования обеспечивается своевременная информация для планирования маршрута с использованием IKZ.

Список литературы / References

[1]. Trujillo-Hernández G., Rodríguez-Quiñonez J.C., Ramírez-Hernández L.R. et al. Accuracy Improvement by Artificial Neural Networks in Technical Vision System. In Proc. of the 45th Annual Conference of the IEEE Industrial Electronics Society, *v*ol. 1, 2019, pp. 5572-5577.

[2]. Rohan A., Rabah M., & Kim S.H. Convolutional Neural Network-based Real-Time Object Detection and Tracking for Parrot AR Drone 2. IEEE Access, vol. 7, 2019, pp. 69575-69584.

[3]. Жданов А.Д., Жданов Д.Д., Богданов Н.Н. и др. Проблемы дискомфорта зрительного восприятия в системах виртуальной и смешанной реальностей. Программирование, том 45, no. 4, 2019 г., стр. 9-18 / Zhdanov A.D., Zhdanov D.D., Bogdanov N.N. et al. Discomfort of Visual Perception in Virtual and Mixed Reality Systems. Programming and Computer Software, vol. 45, no. 4, 2019, pp. 147-155.

[4]. Castro-Toscano M.J., Rodríguez-Quiñonez J. C., Hernández-Balbuena D. et al. (2018). Obtención de Trayectorias Empleando el Marco Strapdown INS/KF: Propuesta Metodológica. Revista Iberoamericana de Automática e Informática industrial, vol. 15, num. 4, 2018, pags. 391-403 (in Spanish)..

[5]. Li W., Wei W., Qiang H., & Shi M. Collaborating visual tracker based on particle filter and correlation filter. Concurrency and Computation: Practice and Experience, vol. 31, no. 12, 2019, e4665.

[6]. Reyes-García M., Sergiyenko O., Ivanov M. et al. Defining the Final Angular Position of DC Motor shaft using a Trapezoidal Trajectory Profile. In Proc. of the 28th IEEE International Symposium on Industrial Electronics (ISIE), 2019, pp. 1694-1699.

[7]. Básaca-Preciado L.C., Sergiyenko O.Y., Rodríguez-Quinonez J.C. et al. Optical 3D laser measurement system for navigation of autonomous mobile robot. Optics and Lasers in Engineering, vol. 54, 2014, pp. 159-169.

[8]. Ivanov M., Lindner L., Sergiyenko O. et al. Mobile Robot Path Planning Using Continuous Laser Scanning. In Optoelectronics in Machine Vision-Based Theories and Applications, IGI Global, 2019, pp. 338-372.

[9]. Иванов М.В, Сергиенко О.Ю., Тырса В.В. и др. Интеграция беспроводной связи для оптимизации распознавания окружения и расчёта траектории движения группы роботов. Труды ИСП РАН, том 31, вып. 2, 2019 г., стр. 67-82. DOI: 10.15514/ISPRAS-2019-31(2)-6 / Ivanov M., Sergiyenko O., Tyrsa V. et al. Software Advances using n-agents Wireless Communication Integration for Optimization of Surrounding Recognition and Robotic Group Dead Reckoning. Programming and Computer Software, vol. 45, no. 8, 2019, pp. 557-569.

[10]. Lindner L., Sergiyenko O., Rodríguez-Quiñonez J.C. et al. Continuous 3D scanning mode using servomotors instead of stepping motors in dynamic laser triangulation. In Proc. of the IEEE International Symposium on Industrial Electronics (ISIE), 2015, pp. 944-949.

[11]. Real-Moreno O., Castro-Toscano M.J., Rodrízuez-Ouiñonez J.C. Implementing k-Nearest Neighbor Algorithm on Scanning Aperture for Accuracy Improvement. In Proc. of the 44th Annual Conference of the IEEE Industrial Electronics Society, 2015, pp. 3182-3186.

Информация об авторе / Information about the author

Хулио Сезар РОДРИГЕС-КИНЬОНЕС, кандидат наук, профессор электроники на инженерном факультете. Область научных интересов: автоматизированная метрология, системы стереозрения, системы управления, роботизированная навигация и лазерные 3D-сканеры.

Julio César RODRÍGUEZ-QUIÑONEZ, PhD, Professor of Electronic Topics with the Engineering Faculty. Research interests: automated metrology, stereo vision systems, control systems, robot navigation and 3D laser scanners.

DOI: 10.15514/ISPRAS–2021–33(1)–5

Цифровые двойники в туманных вычислениях: организация обработки данных с сохранением состояния на базе микропотоков работ

[1] *А.Б.А. Алаасам, ORCID: 0000-0002-2084-8899 <alaasamab@susu.ru>*
[1] *Г.И. Радченко, ORCID: 0000-0002-7145-5630 <gleb.radchenko@susu.ru>*
[1,2,3] *А.Н. Черных, ORCID: 0000-0001-5029-5212 <chernykh@cicese.mx>*
[4] *Х.Л. Гонсалес-Компеан, ORCID: 0000-0002-2160-4407 <jgonzalez@tamps.cinvestav.mx>*

[1] *Южно-Уральский государственный Университет,*
454080, Россия, г. Челябинск, проспект Ленина, д. 76.
[2] *Центр научных исследований и высшего образования*
Мексика, 22860, Нижняя Калифорния, Энсенада, ш. Тихуана-Энсенада, 3918
[3] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, г. Москва, ул. А. Солженицына, дом 25
[4] *Центр перспективных исследований Национального политехнического института,*
Мексика, 87130, Сьюдад-Виктория, Тамаулипас

Аннотация. Цифровые двойники процессов и устройств используют информацию, получаемую с датчиков, для синхронизации своего состояния с сущностями физического мира. Концепция потоковых вычислений позволяет эффективно обрабатывать события, генерируемые такими датчиками. Однако необходимость отслеживания состояния экземпляра объекта приводит к невозможности организации цифровых двойников в виде сервисов без сохранения состояния. Еще одной особенностью цифровых двойников является то, что некоторые задачи, реализованные на их основе, требуют способности реагировать на входящие события на скорости, близкой к реальному времени. В этом случае использование облачных вычислений становится неприемлемым из-за высокой временной задержки. Туманные вычисления решают эту проблему, перемещая некоторые вычислительные задачи ближе к источникам данных. Однако сложности в организации обработки состояния на базе контейнеризованных мироксервисных систем создает проблемы в обеспечении бесперебойной работы таких сервисов в условиях туманных вычислений. Таким образом, основная задача исследования заключается в создании методов контейнеризированной обработки потоков данных с сохранением состояния на основе микросервисов, для поддержки развертывания компонентов цифровых двойников на базе туманных вычислительных сред. В рамках этой статьи мы исследуем возможности живой миграции процессов обработки потоков данных с сохранением состояния и способы перераспределения вычислительной нагрузки между облачными и туманными узлами с использованием платформы Kafka и Kafka Stream DSL API.

Ключевые слова: цифровые двойники; микросервисы; микропотоки работ; потоковая обработка данных; контейнеры; Apache Kafka; облачные вычисления; туманные вычисления

Для цитирования: Алаасам А.Б.А., Радченко Г.И., Черных А.Н., Гонсалес-Компеан Х.Л. Цифровые двойники в туманных вычислениях: организация обработки данных с сохранением состояния на базе микропотоков работ. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 65-80. DOI: 10.15514/ISPRAS–2021–33(1)–5

Благодарности: исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 19-37-90073.

Stateful Stream Processing Containerized as Microservice to Support Digital Twins in Fog Computing

[1] *A.B.A. Alaasam, ORCID: 0000-0002-2084-8899 <alaasamab@susu.ru>*
[1] *G. Radchenko, ORCID: 0000-0002-7145-5630 <gleb.radchenko@susu.ru >*
[1,2,3] *A. Tchernykh, ORCID: 0000-0001-5029-5212 <chernykh@cicese.mx>*
[4] *J.L. Gonzalez-Compean, ORCID: 0000-0002-2160-4407 <jgonzalez@tamps.cinvestav.mx>*

[1] *South Ural State University,*
454080, Russia, Chelyabinsk, Lenin Avenue, 76
[2] *Centro de Investigación Científica y de Educación Superior,*
3918, Ensenada-Tijuana Highway, Ensenada, 22860, Mexico
[3] *Ivannikov Institute for System Programming of the Russian Academy of Sciences,*
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia
[4] *Center for Research and Advanced Studies of the National Polytechnic Institute,*
Tamaulipas, Ciudad Victoria 87130, Mexico

Abstract. Digital twins of processes and devices use information from sensors to synchronize their state with the entities of the physical world. The concept of stream computing enables effective processing of events generated by such sensors. However, the need to track the state of an instance of the object leads to the impossibility of organizing instances of digital twins as stateless services. Another feature of digital twins is that several tasks implemented on their basis require the ability to respond to incoming events at near-real-time speed. In this case, the use of cloud computing becomes unacceptable due to high latency. Fog computing manages this problem by moving some computational tasks closer to the data sources. One of the recent solutions providing the development of loosely coupled distributed systems is a Microservice approach, which implies the organization of the distributed system as a set of coherent and independent services interacting with each other using messages. The microservice is most often isolated by utilizing containers to overcome the high overheads of using virtual machines. The main problem is that microservices and containers together are stateless by nature. The container technology still does not fully support live container migration between physical hosts without data loss. It causes challenges in ensuring the uninterrupted operation of services in fog computing environments. Thus, an essential challenge is to create a containerized stateful stream processing based microservice to support digital twins in the fog computing environment. Within the scope of this article, we study live stateful stream processing migration and how to redistribute computational activity across cloud and fog nodes using Kafka middleware and its Stream DSL API.

Keywords: digital twins; microservices; micro-workflows; stream processing; containers; Apache Kafka; cloud computing; fog computing

For citation: Alaasam A.B.A., Radchenko G., Tchernykh A., Gonzalez-Compean J.L. Stateful Stream Processing Containerized as Microservice to Support Digital Twins in Fog Computing. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 65-80 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–5

Acknowledgements. The reported study was funded by RFBR, project number 19-37-90073.

1. Введение

Технологии интернета вещей (Internet of Things, IoT) обеспечивают интеграцию объектов повседневного мира в глобальную вычислительную среду [1]. Из-за широкого распространения технологий IoT физический и цифровой миры становятся все более взаимосвязанными. Всё чаще возникают задачи организации двусторонней связи между этими мирами. Сегодня эта связь может быть организована посредством моделирования виртуальных объектов на основе данных, получаемых с интеллектуальных сенсоров, расположенных на объектах реального мира. Например, в гонках «Формулы 1» поток данных, которые собираются с сотен датчиков, установленных на автомобиле, передается в реальном времени на панель управления и используется для моделирования рабочих характеристик болида в режиме реального времени [2]. Используя эти модели, инженеры

могут вносить корректировки в режим работы автомобиля удаленно, непосредственно в режиме гонки. Такой подход называется «цифровым двойником» (ЦД).

ЦД – это интегрированная мульти-физическая, мульти-масштабная вероятностная симуляция сложного объекта, которая использует наиболее подходящие физические модели, актуальные данные сенсоров и др. для того чтобы получить как можно более достоверное представление соответствующего реального объекта [3]. Реализация моделей ЦД может требовать больших объемов информации и вычислительных ресурсов [4]. Технология *облачных вычислений* позволяет удовлетворять такие требования посредством динамически-масштабируемого предоставления потенциально неограниченного объема вычислительных ресурсов за счет использования технологий виртуализации [5]. *Виртуализация* позволяет разделять физический вычислительный узел на несколько виртуальных машин (ВМ), каждая из которых имеет собственную изолированную операционную систему (ОС) и приложения. Они координируются слоем программного обеспечения под названием *гипервизор*, который обеспечивает поддержку параллельного выполнения нескольких ВМ на одной физической машине [6].

Тем не менее, накладные расходы, связанные с использованием ВМ в облачных вычислениях, могут ограничить эффективность вычислительных ресурсов. Эта проблема может быть решена при помощи технологии контейнеризации [7]. Контейнеризация позволяет запускать независимые контейнеры в виде отдельных процессов непосредственно на ядре базовой ОС, обеспечивая легкую изоляцию процессов внутри контейнеров. Несмотря на сокращение накладных расходов, обеспечиваемое технологией контейнеризации, высокая латентность и возможные перегрузки сетевых каналов не позволяют облачным решениям обеспечить полное соответствие требованиям таких систем как ЦД, чувствительным к временным задержкам и местоположению [8]. Таким образом, облако не может самостоятельно удовлетворить потребности таких приложений.

Технология *туманных вычислений* (fog computing) позволяет решить эту проблему, перемещая часть задач по обработке и хранению данных из облака ближе к *краю сети* (анг. network edge), в так называемые туманные узлы. Системы туманных вычислений поддерживают локальную обработку данных с приемлемой задержкой, обеспечивая предварительную очистку и предобработку данных, перед отправкой в облако [9]. Производительность таких систем критически зависит от способности безопасно и эффективно собирать, передавать и анализировать потоки данных между объектами реального мира и системами обработки данных в режиме реального времени [10].

Процессы, зависящие только от текущего локального состояния, т.е. не от недавнего прошлого или истории всех таких состояний, называются процессами «без сохранения состояния» (stateless). Для удобства читателя в дальнейшем при описании процессов, сервисов и систем, функционирующих в режиме «без сохранения состояния» мы будем называть их «stateless».

Прогностические системы никогда не могут быть реализованы без сохранения состояния, так как для прогнозирования ближайшего будущего они используют не только текущее состояние, но и прошлый опыт [12]. Такие системы принято называть системами «с сохранением состояния» (stateful). Аналогично, в дальнейшем по ходу статьи процессы, сервисы и системы, функционирующие в режиме «с сохранением состояния», мы будем называть «stateful». Stateful-система способна идентифицировать источник входных данных и определить, какие другие данные поступили из того же источника [13]. Разработчики stateful-систем сталкиваются с рядом проблем, включая ограничения масштабируемости и необходимость дополнительных вычислительных затрат на поддержку управления состоянием [14]. Сложность решения этих задач резко возрастает в сложных системах, таких как ЦД, где необходимо обеспечивать stateful-обработку потоков данных между реальными объектами, туманными узлами и облаком.

Сегодня предполагается, что такие сложные распределенные системы должны состоять из набора слабосвязанных независимых компонентов [15]. Такие решения легли в основу «микросервисного» (microservice) архитектурного подхода [16]. Микросервисы – это архитектурный паттерн, который делает акцент на разделении системы на легкие независимые сервисы, каждый из которых отвечает за выполнение отдельной независимой части бизнес-логики приложения [17]. Для обеспечения изоляции микросервисов часто применяются технологии контейнеризации. Однако ограниченная поддержка переносимости состояния в контейнерах приводит к проблемам контейнеризации stateful-сервисов [18]. Кроме того, обработка данных в stateless-режиме является одной из ключевых характеристик микросервисного подхода [16,17]. Stateless-обработка данных позволяет относительно легко добиться таких важных характеристик микросервисных систем, как высокая надежность, высокая доступность, масштабируемость по требованию, балансировка нагрузки [19]. Например, stateless-приложения могут быть развернуты за балансировщиками нагрузки, где их сбои маскируются путем перенаправления трафика на рабочие реплики данного приложения. Но для обеспечения подобных характеристик для stateful-сервисов необходимо использование других подходов [20].

Когда обработка потоков данных требует информации о данных в предыдущих временных сегментах, их реализация в виде stateless-микросервисов становится невозможной. Поддержка сохранения состояния требует применения особых методов для обеспечения высокой надежности и доступности, а приложения такого типа сложнее масштабировать. Решение для управления stateful-сервисами может быть обеспечено посредством платформы обработки потоков данных. Такая система может взять на себя роль «нервной системы» для независимых вычислительных сервисов, в то время как каждый из них выполняет свою часть бизнес-логики.

В этой статье мы представим обзор проблем и требований к решениям по stateful-обработке потоков данных на основе контейнеризированных микросервисов для поддержки развертывания компонентов ЦД на базе туманных вычислений. Кроме того, мы проведем анализ возможности реализации предлагаемого решения с использованием Kafka Streams DSL API[1] для живой миграции контейнеризированных stateful-сервисов обработки потоковых данных.

Статья структурирована следующим образом. В разд. 2 рассматривается проблема обработки данных с сохранением состояния: stateful вычислительной инфраструктуры и stateful-данных. В разд. 3 проводится анализ литературы по теме исследования. В разд. 4 рассматриваются источники данных, и архитектура предлагаемой системы. В разд. 5 рассматривается эксперимент организации живой миграции. В разд. 6 рассматривается эксперимент по распределению вычислительной нагрузки по туманной вычислительной среде. Выводы представлены в разд. 7.

2. Работа с сохранением состояния

Состояние – это любая информация, характеризующая процесс, в течение некоторого внутреннего времени, и может храниться в любом месте иерархии агентов и субагентов, характеризующих процесс [12]. Для выполнения операций с сохранением состояния (stateful-операций) требуется возможность идентификации источника входных данных и определение того, какие еще входные данные были получены из того же источника [13].

Хранение состояния внутри вычислительного сервиса считается узким местом, поэтому состояние должно храниться в отдельном ресурсе [14], например, внешней базе данных, внешней файловой системе, системе кэширования или в промежуточном программном обеспечении для обмена сообщениями.

[1] https://kafka.apache.org

Тем не менее, отдельный ресурс для работы с состоянием приводит к необходимости выделения дополнительных ресурсов, которые могут включать в себя дополнительные вычислительные мощности, память и хранилища данных. Кроме того, такой подход может приводить к проблемам в масштабируемости. В связи с этим, любая stateful-операция сталкивается со сложностью управления состоянием.

Сложность возрастает в условиях туманной вычислительной среды, где важное значение имеет способность живой миграции задач обработки и хранения данных между туманными узлами. *Живой миграцией* называют возможность бесперебойной миграции сервиса между физическими машинами без воздействия на клиентские процессы или приложения. В этом случае вычислительный процесс приостанавливается только на время передачи общего состояния, после чего вычисление возобновляется на целевом узле.

Обеспечение механизма, обеспечивающего правильность обработки, хранения и восстановления состояния, является существенным процессом при реализации stateful-операций. Создание stateful микросервисной системы приводит к целому ряду проблем, наиболее существенными из которых являются предоставление stateful вычислительной инфраструктуры и управление stateful-данными.

2.1 Stateful вычислительная инфраструктура

Сложность проектирования stateful-систем, таких как ЦД, зависит от возможностей базовой вычислительной инфраструктуры. Мы будем называть stateful вычислительной инфраструктурой, такую инфраструктуру виртуализации, которая позволяет хранить и управлять состоянием внутри изолированной вычислительной среды в течение длительного промежутка времени. Технология ВМ обеспечивает возможность создания моментальных снимков и восстановления состояния приложений. Это позволяет осуществлять миграцию ВМ между физическими хостами с минимальным влиянием на запущенные сервисы [21]. Эти технологии применяются во многих платформах управления ВМ, таких как VMware vCenter[1]. Такие снимки позволяют обеспечить миграцию ВМ между вычислительными узлами без существенного прерывания вычислительного процесса и воздействия на их состояние.

К сожалению, такой подход не распространен в технологии контейнеризации. Например, такая миграции не так проста в реализации при использовании платформы Docker[2]. При создании нового контейнера в Docker, новый слой (слой контейнера), доступный для записи, создается поверх слоев, доступных только для чтения (слой изображения), а все изменения, внесенные в контейнер, записываются только на слой контейнера. Методы создания контрольных точек состояния контейнера в настоящее время находятся только в тестовой версии Docker [22]. Это означает, что до сих пор нет непосредственного и оригинального способа организовать живую миграцию докер-контейнеров на другой узел, не потеряв при этом всего состояния.

Существуют несколько методов решения проблемы миграции контейнеров. Например, проект CRIU и его проекты-расширения по-прежнему основаны на экспериментальном режиме Docker[3]. Также CRIU выполняет операцию «PID dance», чтобы восстановить процесс с таким же PID. Эта операция требует привилегированного доступа и обладает низкой производительностью, так как генерирует множество системных вызовов и может привести к возникновению состояния гонки [23]. С другой стороны, контейнеры Linux LXD[4] поддерживают возможность создания снимков и восстановления контейнеров для резервного

[1] https://www.vmware.com/products/vcenter-server.html

[2] https://docs.docker.com/storage/storagedriver/

[3] https://criu.org/Docker

[4] https://ubuntu.com/blog/lxd-2-0-your-first-lxd-container

копирования или миграции. Но, для целей живой миграции, LXD все еще базируется на CRIU[1].

Между тем, некоторые контейнерные платформы, такие как Kubernetes[2], которые изначально строились для поддержки stateless-сервисов, недавно расширили свои модели, включив в них работу с состоянием [12]. Kubernetes предоставляет контроллер StatefulSet для управления stateful-приложениями. Тем не менее, его реализация также ограничена, так как приводит к потере состояния при реализации «rolling-обновлений», сложностям или невозможности применения балансировщиков нагрузки и др. [24]. Ключевой же проблемой в управлении StatefulSet является отсутствие механизмов живой миграции и автоматического создания замены вышедшего из строя StatefulSet [25,26]. Поэтому решение проблемы сохранения состояния при использовании технологии контейнеризации является активной областью исследований. Особенно при обработке stateful-потоков данных, где каждая точка данных может играть решающую роль.

2.2 Stateful-данные

Чтобы организовать управление состоянием, желательно обеспечить постоянное хранение данных о состоянии не только в вычислительных сервисах, но и в других частях системы. Примерами таких систем могут служить внешняя база данных, внешняя файловая система, система кэширования, промежуточное программное обеспечение для обмена сообщениями.

Однако идеальных решений нет, так как любой отдельный ресурс для обработки состояния требует дополнительных серверных ресурсов: дополнительной вычислительной мощности, памяти и места для хранения данных. Например, платформа контейнеризации OpenVZ[3] поддерживает миграцию контейнеров с сохранением состояния с помощью проекта на основе CRIU. OpenVZ использует распределенную систему хранения данных (distributed storage system, DSS) для совместного использования файлов. Однако дополнительные данные, необходимые для репликации на основе DSS, приводят к росту трафика, снижая производительность сети [27]. Таким образом, использование технологий репликации на основе DSS может привести к большим задержкам и ухудшению качества обработки данных на краевых узлах сети.

Проблема усложняется, когда обработка данных ведется не в пакетном, а в потоковом режиме. Потоки данных часто должны обрабатываться последовательно, посредством применения механизмов stateful-обработки временных рядов, таких как скользящее среднее и др. [28].

Для потоковой обработки данных требуются два слоя: *слой хранения* (storage layer, SL) и *слой обработки* (processing layer, PL). SL должен поддерживать быстрый ввод/вывод для больших потоков данных. PL потребляет данные из SL, обрабатывает эти данные и уведомляет SL об обновлении данных. Например, для обеспечения отказоустойчивости, платформа обработки данных Apache Spark[4] поддерживает различные источники входных и выходных данных. Для обеспечения хранения данных она должна быть сконфигурирована с использованием внешней системы хранения, такой как HDFS. Тем не менее, живая миграция для Apache Spark все еще реализуется на базе механизмов миграции ВМ, например, предоставляемых облачной платформой OpenStack [29]. Таким образом, при разработке сложной системы, такой как ЦД, необходимо планировать, как управлять распределением stateful-данных по компонентам

[1] https://lxd.readthedocs.io/en/latest/migration/

[2] https://kubernetes.io/

[3] https://wiki.openvz.org/Virtuozzo_Storage

[4] http://spark.apache.org/docs/latest/streaming-programming-guide.html#checkpointing

системы и подбирать инструменты обработки данных, обеспечивающие возможности их обработки в режиме сохранения состояния.

3. Обзор литературы

В научном сообществе растет интерес к области управления обработкой данных IoT. Например, авторы [30] предложили распределенный алгоритм отслеживания людей для системы, состоящей из камер видеонаблюдения, подключенных к локальным вычислительным узлам, которые, в свою очередь, подключены к центру обработки данных. Обнаружение людей происходит в центре обработки данных, в то время как локальные вычислительные узлы обеспечивают выполнение задач по предобработке видеопотока.

Авторы [31] используют комбинацию хранилища данных Redis[1] с платформой Apache Storm[2] для обеспечения краткосрочного прогнозирования нагрузки и обнаружения выбросов в потоке данных, генерируемых интеллектуальными датчиками измерения энергии. Для поддержки передачи данных авторы использовали библиотеку обмена сообщениями на основе кольцевого буфера LMAX. Авторы [32] также использовали систему Apache Storm, но для реализации алгоритма статистического контроля под названием «CUmulative SUM» (CUSUM) для выявления аномалий в сериях данных мониторинга окружающей среды. Они использовали Apache ActiveMQ[3] в качестве сервиса обмена сообщениями.

Кроме того, наблюдается взрывной рост платформ и различных решений, предлагаемых для анализа потоков данных. Например, платформа Apache Spark обеспечивает обработку данных как в пакетном, так и в потоковом режиме, поддерживает решения задач машинного обучения. Еще одной платформой, нашедшей свое применение для обработки потоков данных, является Apache Kafka. Apache Kafka является отказоустойчивой платформой распределенной потоковой обработки данных. Apache Kafka хранит данные в *темах* (Kafka topics), которые представляют собой названия категорий, в рамках которых публикуются и потребляются сообщения. Kafka Streams DSL API обеспечивает поддержку обработки локального состояния внутри приложения для таких stateful-операций, как подсчет или агрегирование.

Чтобы отслеживать обновление состояния, Kafka поддерживает реплицируемый журнал изменений для каждого хранилища состояний. Каждое приложение, работающее с Apache Kafka должно иметь уникальный идентификатор для поддержки отслеживания состояния. Однако эти возможности привносят определенные ограничения. Например, операции потребления, сохранения состояния и генерации в Kafka Streams DSL могут выполняться только на одном кластере Kafka. Кроме того, кластер Kafka до сих пор не поддерживает географическую репликацию постоянно хранящихся сообщений в нескольких центрах обработки данных. Это приводит к проблемам применения этой технологии в туманной вычислительной среде. В настоящий момент есть два альтернативных решения: MirrorMaker[4] и Confluent Replicator [33]. MirrorMaker является инструментом для зеркалирования данных между кластерами, но, в дополнение к другим ограничениям, он не дает гарантий того, что конфигурация тем Kafka в реплике будет совпадать с конфигурацией в оригинальном кластере. Также, он не предоставляет UI для мониторинга репликации. Confluent Replicator охватывает многие ограничения MirrorMaker, но является проприетарным решением, требующим платной версии платформы Confluent.

[1] https://redis.io/

[2] https://storm.apache.org/

[3] http://activemq.apache.org/

[4] https://docs.confluent.io/4.0.0/multi-dc/mirrormaker.html

4. Исходные данные и процесс обработки

4.1 Данные и инструменты

Для оценки предлагаемого решения обработки stateful-данных мы использовали реальный набор данных, предоставленный DEBS 2012 Grand Challenge[1], собранный из датчиков, установленных на производственном оборудовании. Задержка между двумя последовательными точками данных составляет около 10 мс. В рамках данного исследования мы остановились на реализации первого запроса по обработке данных. Каждый элемент исходных данных состоит из 66 полей. Первый набор операторов обеспечивает обнаружение изменения состояния во входных полях между последовательными точками исходных данных и выдает их вместе с временными метками изменения состояния. Второй набор операторов решает задачу установления корреляции между изменением состояния датчика и изменением состояния клапана, а также рассчитывает временное расстояние между наступлением изменения состояния и выдает эти данные с временными штампами.

Для автоматизации развертывания Apache Kafka и реестра схем в контейнере мы используем Docker-образ, предоставляемый Lenses[2]. Все микросервисы во всех экспериментах разрабатываются с использованием Java 8 с Kafka Streams DSL API и контейнеризированы с помощью Docker.

4.2 Предлагаемый процесс обработки данных

Мы организуем обработку данных в два этапа. В каждом из них Kafka Streams DSL API автоматически синхронизирует локальное хранилище состояния с промежуточными темами Kafka. Первый этап включает в себя следующие основные шаги:

1) вотребление потока данных из исходной темы;
2) группировка потоков: каждая группа отвечает за свой источник данных;
3) агрегация сгруппированного потока: включает в себя те точки данных, в которых изменилось состояние источника данных;
4) встраивание агрегированных данных в поток результирующих данных: мы создаем новое сообщение с обновленной схемой сообщения, которая включает новое значение изменения состояния вместе с меткой времени;
5) передача результата в виде субпотока следующей группе операторов или отправка его в тему Kafka, если первый этап был реализован в отдельном микросервисе.

Второй этап включает в себя следующие шаги:

1) Если оба этапа обработки выполняются в рамках одного микросервиса, обеспечивается потребление данных от первого этапа обработки; если каждый из этапов обработки реализован в отдельном микросервисе, обеспечивается потребление исходного потока из темы, хранящей результаты первого этапа;
2) Перегруппировка полученных сообщений, которые включают в себя только сообщения об обнаруженном изменении состояния для каждого датчика;
3) Агрегация полученного потока для анализа корреляции между изменением состояния датчиков и изменением состояния клапанов;
4) Встраивание агрегированного значения в поток результатов; создание нового сообщения, которое включает обновленную схему сообщений, состоящую из нового результата анализа корреляции между изменением состояния датчиков и изменением состояния клапана вместе с временной меткой;

[1] http://debs.org/grand-challenges/

[2] https://lenses.io/

5) Отправка результата в тему результатов Kafka.

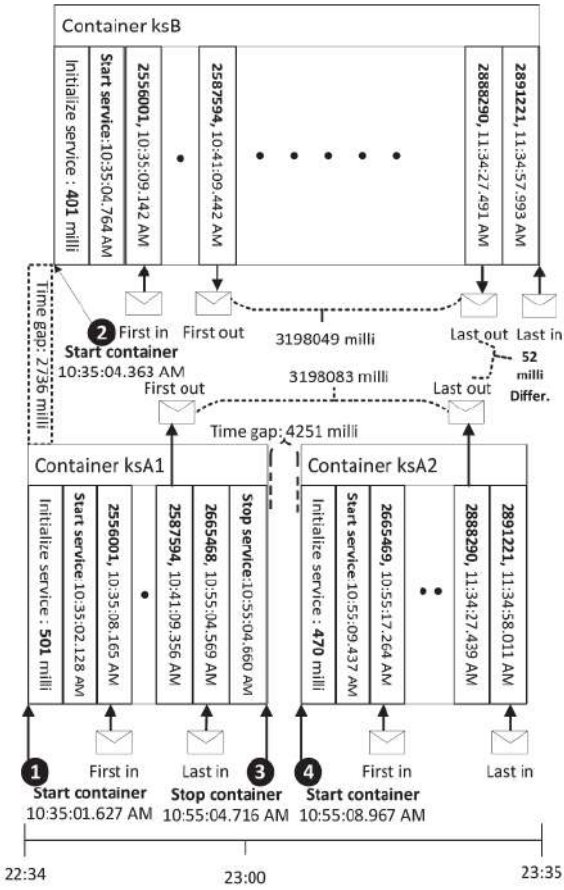


Рис. 1. Эксперимент по живой миграции
Fig. 1. Live migration experiment.

5. Эксперимент по живой миграции

Этот эксперимент проверяет возможность восстановления работы после остановки или выхода из строя контейнера, без влияния на финальные результаты обработки данных. Кроме того, он проверяет возможность использования функции синхронизации локального хранилища состояния с промежуточными темами Kafka в качестве основы для переноса вычислительной задачи в новый контейнер, сохраняя при этом непрерывность результатов из предыдущей точки остановки.

5.1 Методология развертывания и тестирования

Тест начинается с инициализации двух контейнеров (`ksA1`, `ksB`) с одним и тем же микросервисом, но каждый с уникальным идентификатором приложения и различными темами вывода (см. точки 1 и 2 на рис. 1). После этого производится запуск источника данных, который генерирует поток данных в течение одного часа. Через 20 минут завершается работа контейнера `ksA1` (см. точку 3 на рис. 1), а новый контейнер (`ksA2`) запускается с тем же микросервисом, той же темой ввода и вывода, и тем же идентификатором приложения, что и `ksA1` (см. точку 4 на рисунке 1). Таким образом, мы

позволяем `ksA2` повторно использовать предыдущие промежуточные темы `ksA1` для получения состояния `ksA1` в `ksA2`. Через 1 час генерация данных прекращается, и мы оцениваем результаты проведенного эксперимента.

5.2 Оценка результатов эксперимента

Корректность обработки данных составила 100% во всех тестах. Оценка корректности обработки данных была проведена путем сравнения результатов контейнера `ksB` с результатами пары `ksA1` и `ksA2`. Вычисляя разницу во времени между первым и последним итоговыми результатами в каждом тесте, мы можем рассчитать среднее итоговое время, требуемое для завершения обработки всех данных. Среднее общее временя выполнения теста, как при обработке данных посредством одного контейнера `ksB`, так и при прерывании контейнера `ksA1` и запуске контейнера `ksA2`, является очень похожим, с разницей +- 52 мс. Средний промежуток времени между остановкой `ksA1` и запуском `ksA2` составляет 4251 мс. Это время имеет важное значение, поскольку это один из накладных расходов при миграции вычислений из одного контейнера в другой. Среднее время между запуском контейнера и запуском сервиса внутри контейнера составляет 475 мс. Это время – также один из накладных расходов, связанных с запуском контейнеризованного сервиса, а также накладные расходы, связанные с остановкой контейнеров и переносом вычислений в другой контейнер.

6. Эксперимент по распределению вычислительной нагрузки в туманной вычислительной среде

В этом эксперименте мы сравниваем два подхода к организации обработки входных данных (см. рис. 2). Первый подход заключается в реализации обоих этапов процесса обработки данных в одном микросервисе (`ks_total`), размещенным в частном облаке ЮУрГУ, которое располагается рядом с источником данных. Во втором подходе, тот же вычислительный процесс разделяется на два микросервиса (`ks1_susu`) и (`ks2_yandex`). `ks1_susu` реализует первый этап обработки данных и развернут в частном облаке в ЮУрГУ в то время как `ks2_yandex` реализует второй этап процесса обработки данных и развернут в публичном облаке компании Яндекс (Яндекс.Облако). Обмен данными между ними организован через два географически разделенных кластера Kafka. Чтобы преодолеть проблемы репликации данных, которые все еще существуют в Kafka, нами был реализован репликатор для организации синхронизации географически разделенных кластеров Kafka. Он разработан на основе концепции микро-потоков работ (анг. Micro-Workflow - MW) [34–36].

На рис. 2 компоненты синей прямоугольной формы в микросервисах `ks_total`, `ks1_susu` и `ks2_yandex` представляют собой вычислительные единицы, представляющие *слой обработки* (PL). Компоненты желтой трубчатой формы (такие как кластеры Kafka и хранилище локальных состояний) представляют собой *слой хранения* (SL). В наших экспериментах SL также распределен: локальное хранилище состояния расположено непосредственно в контейнере, осуществляющем обработку данных. Эксперимент исследует потенциал для повышения параллелизма и возможность развертывания части вычислительной рабочей нагрузки в публичном облаке, в то время как другая часть вычислительной рабочей нагрузки развертывается рядом с источником данных, чтобы обеспечить более быстрое время отклика. Мы сравниваем результаты выполнения обработки данных в рамках системы, состоящей из микросервисов `ks1_susu` и `ks2_yandex` с результатами соответствующих частей обработки данных в `ks_total`.

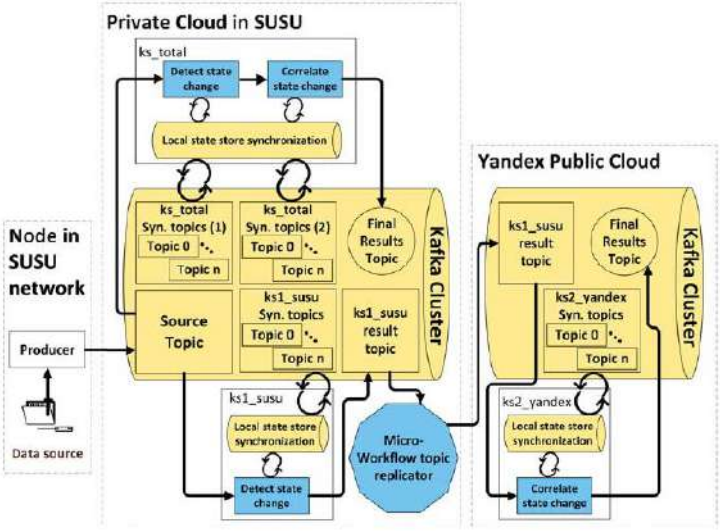


Рис. 2. Детали реализации эксперимента по распределению вычислительной нагрузки
Fig. 2. Implementation details of the computational load distribution experiment.

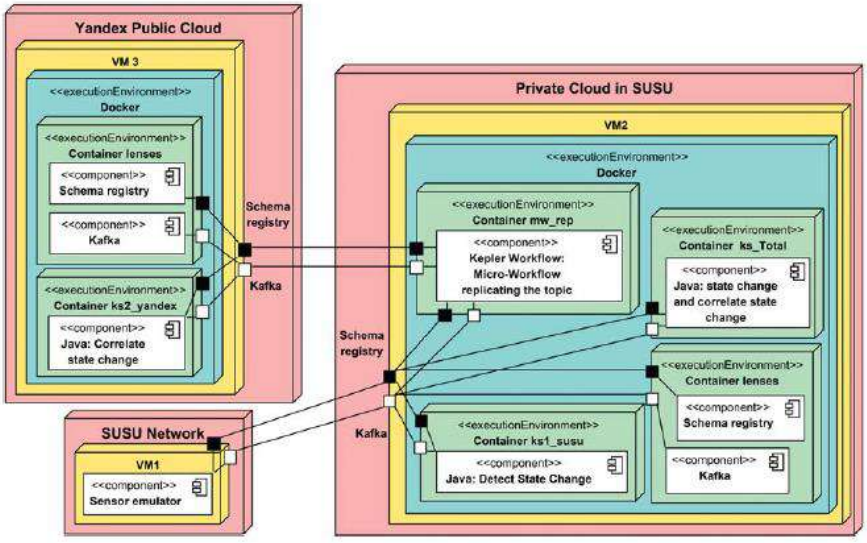


Рис. 3. Размещение компонентов эксперимента по распределению вычислительной нагрузки
Fig. 8. Deployment of the components of the computational load distribution experiment

6.1 Развертывание эксперимента

На рис. 3 показаны детали инфраструктуры, на которой развернут эксперимент.

Эксперимент развернут на трех вычислительных узлах. Узел VM1 (4 ГБ оперативной памяти, двухъядерный процессор Intel Xeon X5680) развернут в сети ЮУрГУ. Здесь находится эмулятор датчика. Узел VM2 (8 ГБ оперативной памяти, восьмиядерный процессор Intel (R) Xeon (R) Gold 6242) развернут в частном облаке в ЮУрГУ. Здесь размещен первый кластер Kafka, контейнер `ks_total`, контейнер `ks1_susu`, а также контейнер `mw_rep`, который включает MW для репликации сообщений из темы в кластере Kafka в кластере VM2 в кластер

Kafka в VM3. Узел VM3 (6 ГБ оперативной памяти, двухъядерный процессор Intel Cascade Lake) развернут в Яндекс.Облаке. Здесь находится второй кластер Kafka и контейнер `ks2_yandex`.

6.2 Оценка результатов эксперимента

Задержка передачи данных между VM1 и VM2 составляет в среднем 2 мс. Задержка между VM2 и VM3 составляет в среднем 30 мс. Продолжительность экспериментов составила 6 часов. К каждом из экспериментов было обработано 1 638 104 исходных сообщения. Промежуток времени между двумя последовательными сообщениями от эмулятора датчика составлял в среднем 14 мс. Сравнение значений всех результатов, полученных `ks1_susu` и `ks2_yandex` с результатами соответствующих частей в `ks_total` показало 100% соответствие результатов обработки данных с точки зрения значений. Оба микросервиса (как `ks1_susu`, так и его соответствующая часть в `ks_total`) получили 1 638 104 входных сообщений и сгенерировали 1 616 результирующих сообщений во время теста.

Зная время первого и последнего результирующего сообщения, мы можем оценить период времени, который потребовался чтобы завершить обработку полученных данных. Рис. 4 показывает, что `ks1_susu` и его соответствующая часть в `ks_total` начать генерацию результатов одновременно, но `ks1_susu` выдает последний результат на 49 мс быстрее.

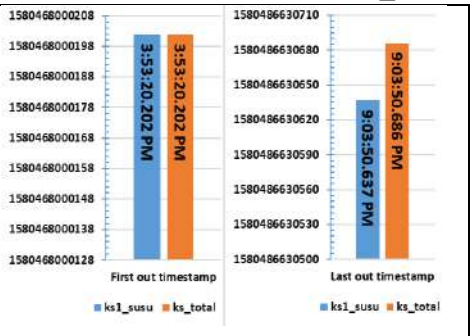


Рис. 4. Временные отметки первых и последних результатов ks1_susu и соответствующей части ks_total
Fig. 4. Time of first and last results of ks1_susu and the corresponding part of ks_total

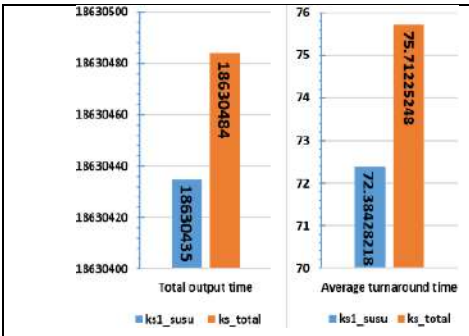


Рис. 5. Итоговое время вывода и время оборота ks1_susu и соответствующая его часть ks_total
Fig. 5. Total output time and turnaround time of ks1_susu and the corresponding part of ks_total

На рис. 5 показано итоговое время вывода и время оборота. *Время оборота* – это средний временной интервал, необходимый для получения одного результирующего сообщения посредством микросервиса. Заметим, что интервал начинается с момента получения второго исходного сообщения, которое участвовало процессе изменения состояния из источника Kafka, и заканчивается в момент отправки результирующего сообщения в соответствующую тему Kafka (в случае `ks1_susu`) или в следующий этап рабочего процесса обработки данных (в случае `ks_total`). `ks1_susu` завершил работу быстрее, чем его соответствующая часть в `ks_total`. Кроме того, среднее время оборота `ks1_susu` было меньше, чем среднее время оборота его соответствующей части в `ks_total`, примерно на 3 мс.

Как `ks2_yandex`, так и его соответствующая часть в `ks_total` получили 1 616 исходных сообщения и сгенерировали 816 результирующих сообщения во время теста. Данные на выходе из `ks1_susu` являются потоком данных, получаемых `ks2_yandex`.

Результаты обработки `ks1_susu` передаются в тему, расположенную в первом кластере Kafka в частном облаке в ЮУрГУ. Это сообщение затем реплицируется репликатором на базе MW в тему во втором кластере Kafka в Яндекс.Облаке, а затем потребляется `ks2_yandex`

для реализации этапа обработки данных «Установление корреляции». В рамках же `ks_total`, входные данные для этого шага обработки являются данными результатов первого шага рабочего процесса, реализованного в том же `ks_total` (см. рис. 2).

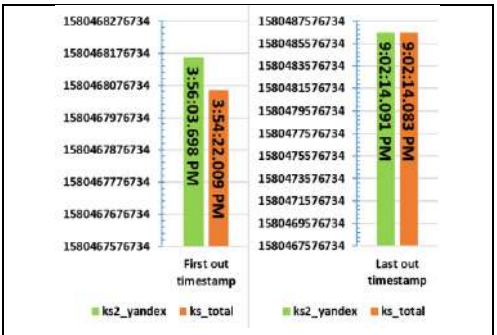


Рис. 6. Временные отметки первых и последних результатов ks2_yandex и соответствующей части ks_total

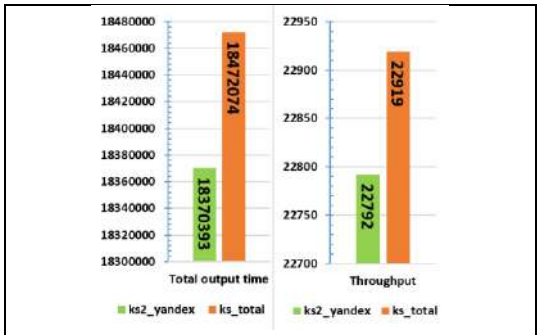


Рис. 7. Итоговое время вывода и пропускная способность ks2_yandex и его соответствующей части в ks_total

При анализе времени получения результатов надо учитывать, что результаты `ks2_yandex` включают дополнительную задержку, связанную с передачей данных между VM2, развернутом в частном облаке в ЮУрГУ, и VM3, развернутом в Яндекс.Облаке. Тем не менее, время задержки постепенно сокращается во время экспериментов, см. рис. 6. Сокращение времени задержки можно объяснить следующим образом: `ks1_susu` генерирует результаты быстрее, чем соответствующая часть `ks_total` (см. рис. 5). Кроме того, `ks2_yandex` генерирует результаты быстрее, чем его соответствующая часть в `ks_total`. Итоговое время на получение всех результатов в `ks2_yandex` составляет на 101 681 мс меньше, чем итоговое время, затраченное на получение результатов в соответствующей части в `ks_total`. Таким образом, `ks2_yandex` имеет более высокую пропускную скорость по сравнению с соответствующей частью в `ks_total` на 127 мс (см. рис. 7).

7. Выводы

Наше исследование показывает важность двух факторов, влияющих на построение обработки потоков данных с сохранением состояния в таких сложных системах, как ЦД: stateful вычислительная инфраструктура и stateful-данные. Stateful вычислительная инфраструктура упрощает сложность и характер вычислительной деятельности за счет способности инфраструктуры хранить и управлять данными внутри изолированной вычислительной системы в течение длительных промежутков времени. Тем не менее, важно планировать, как управлять stateful-данными при реализации микросервисной архитектуры и выбирать инструменты обработки данных, которые обеспечивают сохранение состояния. Кроме того, если система обработки потоков данных не поддерживает репликацию сообщений в географически разделенных центрах обработки данных, это создает проблемы с точки зрения надежности, долговечности и доступности данных.

Результаты нашего эксперимента показывают, что подход, основанный на создании локальных хранилищ состояния в привязке к приложениям на слое обработки и синхронизации промежуточных данных со слоем хранения (как, например Kafka Stream DSL API в наших экспериментах) может быть использован для поддержки живой миграции контейнеризованных приложений с сохранением состояния в туманной вычислительной среде с минимальными накладными расходами (за исключением повышения временной задержки). Если состояние хранится в слое хранения, на задержку влияют два основных

фактора. Во-первых, расстояние между приложением и слоем хранения. Во-вторых, это объем трафика, который генерируется при обмене промежуточными данными между приложением и слоем хранения. Кроме того, результат наших экспериментов показывает, что распределение обработки потока данных на изолированные сервисы повышает параллелизацию.

В качестве направления дальнейших исследований мы планируем развивать MW подход к обработке потоков данных. Важными темами для дальнейших исследований являются исследование репликаторов и подходов обработки потоков данных в географически-разделенных центрах обработки данных для поддержки развертывания компонентов ЦД в туманной вычислительной среде.

Список литературы / References

[1]. A. Tchernykh, M. Babenko, N. Chervyakov et al. Scalable Data Storage Design for Non-Stationary IoT Environment with Adaptive Security and Reliability. IEEE Internet of Things Journal, vol. 7, no. 10, 2020, pp. 10171-10188.

[2]. G. E. Modoni, M. Sacco, W. Terkaj. A Telemetry-driven Approach to Simulate Data-intensive Manufacturing Processes. Procedia CIRP, vol. 57, 2016, pp. 281-285.

[3]. E. H. Glaessgen and D. D. S. Stargel. The Digital Twin Paradigm for Future NASA and U.S. Air Force Vehicles. In Proc. of the 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference - Special Session on the Digital Twin, 2012, pp. 1-14.

[4]. M. Grieves and J. Vickers. Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In Transdisciplinary Perspectives on Complex Systems, Springer, 2017, pp. 85–113.

[5]. Q. Zhang, L. Cheng, R. Boutaba. Cloud computing: state-of-the-art and research challenges. Journal of Internet Services and Applications, vol. 1, no. 1, 2010, pp. 7-18.

[6]. Б.М. Шабанов, О.И. Самоваров. Принципы построения межведомственного центра коллективного пользования общего назначения в модели программно-определяемого ЦОД. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 7-24. DOI: 10.15514/ISPRAS-2018-30(6)-1 / B.M. Shabanov and O.I. Samovarov. Building the Software-Defined Data Center. Programming and Computer Software, vol. 45, no. 8, 2019, pp. 458-466 (in Russian).

[7]. G. I. Radchenko, A. B. A. Alaasam, A. N. Tchernykh. Comparative Analysis of Virtualization Methods in Big Data Processing. Supercomputing Frontiers and Innovations, vol. 6, no. 1, 2019, pp. 48-79.

[8]. J. Luo, L. Yin, J. Hu et al. Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT. Future Generation Computer Systems, vol. 97, 2019, pp. 50-60.

[9]. M. Aazam, S. Zeadally, K. A. Harras. Deploying Fog Computing in Industrial Internet of Things and Industry 4.0. IEEE Transactions on Industrial Informatics, vol. 14, no. 10, 2018, pp. 4674-4682.

[10]. S. Singh, A. Angrish, J. Barkley et al. Streaming Machine Generated Data to Enable a Third-Party Ecosystem of Digital Manufacturing Apps. Procedia Manufacturing, vol. 10, 2017, pp. 1020-1030.

[11]. Y. Qamsane, C. Chen, E. C. Balta et al. A Unified Digital Twin Framework for Real-time Monitoring and Evaluation of Smart Manufacturing Systems. In Proc. of the 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), 2019, pp. 1394-1401.

[12]. M. Burgess. Locality, Statefulness, Causality in Distributed Information Systems Concerning the Scale Dependence Of System Promises. arXiv, 2019, available: http://arxiv.org/abs/1909.09357.

[13]. C. Peiffer and I. L'Heureux. System and method for maintaining statefulness during client-server interactions. United States Patent. US8346848B2, 2013.

[14]. M. Naseri and A. Towhidi. Stateful Web Services: A Missing Point in Web Service Standards. In In Proc. of the International MultiConference of Engineers and Computer Scientists, 2007, pp. 993-997.

[15]. R. Lichtenthaler. Model-driven software migration towards fine-grained cloud architectures. CEUR Workshop Proceedings, vol. 2339, 2019, pp. 35-38.

[16]. S. Newman. Building Microservices: Designing Fine-Grained System. O'Reilly Media, 2015, 280 p.

[17]. James Lewis and Martin Fowler. Microservices. 2014. Available at: https://martinfowler.com/articles/microservices.html, accessed Jan. 11, 2019.

[18]. The Microservice Revolution: Containerized Applications, Data and All, Available at: https://www.infoq.com/articles/microservices-revolution, accessed Dec. 10, 2019.

[19]. C. Fehling, F. Leymann, R. Retter, W. Schupeck, P. Arbitter. Cloud Computing Patterns: Fundamentals to Design, Build, Manage Cloud Applications. Vienna: Springer Vienna, 2014.

[20]. W. Li and A. Kanso. Comparing containers versus virtual machines for achieving high availability. In Proc. of the 2015 IEEE International Conference on Cloud Engineering, 2015, pp. 353-358.
[21]. C. Clark, K. Fraser, S. Hand et al. Live Migration of Virtual Machines. In Proc. of the 2nd conference on Symposium on Networked Systems Design & Implementation, vol. 2, 2005, pp. 273-286.
[22]. docker checkpoint | Docker Documentation, Available at: https://docs.docker.com/engine/reference/commandline/checkpoint, accessed Dec. 11, 2019.
[23]. A. Reber. CRIU and the PID dance. In Proc. of the Linux Plumbers Conference, 2019, pp. 1-4.
[24]. StatefulSets – Kubernetes, Available at: https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/#limitations, accessed Dec. 18, 2019.
[25]. L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, F. Khendek. Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes. In Proc. of the 19th IEEE International Conference on Software Quality, Reliability and Security, 2019, pp. 176–185.
[26]. H. Loo, A. Yeo, K. Yip, T. Liu. Live Pod Migration in Kubernetes. University of British Columbia, Vancouver, Canada. [Online]. Available at : https://www.cs.ubc.ca/~bestchai/teaching/cs416_2017w2/project2/project_m6r8_s8u8_v5v8_y6x8_proposal.pdf
[27]. H. Ohtsuji and O. Tatebe. Network-Based Data Processing Architecture for Reliable and High-Performance Distributed Storage System. Lecture Notes in Computer Science, vol. 9523, 2015, pp. 16–26.
[28]. A. B. A. Alaasam, G. Radchenko, A. Tchernykh. Stateful Stream Processing for Digital Twins: Microservice-Based Kafka Stream DSL. In Proc. of the International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON), 2019, pp. 0804–0809.
[29]. A. Raddaoui, A. Settle, J. Garbutt, S. Singh. High Availability of Live Migration. openstack.org, 2017. [Online]. Available at: http://superuser.openstack.org/wp-content/uploads/2017/06/ha-livemigrate-whitepaper.pdf, accessed Dec. 11, 2019.
[30]. Д.А. Купляков, Е.В. Шальнов, В.С. Конушин, А.С. Конушин. Распределенный алгоритм сопровождения для подсчета людей в видео. Программирование, том. 45, no. 4, стр. 28-35. / D.A. Kuplyakov, E.V. Shalnov, V.S. Konushin, A.S. Konushin. A Distributed Tracking Algorithm for Counting People in Video. Programming and Computer Software, vol. 45, no. 4, 2019, pp. 163–170.
[31]. A. Sunderrajan, H. Aydt, A. Knoll. DEBS Grand Challenge : Real time Load Prediction and Outliers Detection using STORM. In Proc. of the 8th ACM International Conference on Distributed Event-Based Systems, 2014, pp. 294–297.
[32]. S. Trilles, S. Schade, O. Belmonte, J. Huerta. Real-Time Anomaly Detection from Environmental Data Streams. Lecture Notes in Geoinformation and Cartography, vol. 217, 2015, pp. 125–144.
[33]. Apache Kafka's MirrorMaker – Confluent Platform, Available at: https://docs.confluent.io/4.0.0/multi-dc/mirrormaker.html, accessed Apr. 15, 2020.
[34]. G. Radchenko, A. Alaasam, A. Tchernykh. Micro-Workflows: Kafka and Kepler Fusion to Support Digital Twins of Industrial Processes. In Proc. of the IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), 2018, pp. 83–88.
[35]. А.Б.А. Алаасам, Г.И. Радченко, А. Н. Черных. Микро-потоки работ: сочетание потоков работ и потоковой обработки данных для поддержки цифровых двойников технологических процессов. Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика, том 8, no. 4, 2019 г., стр. 100-116 / A. B. A. Alaasam, G. Radchenko, A. Tchernykh. Micro-Workflows: A Combination of Workflows and Data Streaming to Support Digital Twins of Production Processes. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering, vol. 8, no. 4, 2019, pp. 100–116, Nov. 2019 (in Russian).
[36]. A.B.A. Alaasam, G. Radchenko, A. Tchernykh et al. Scientific Micro-Workflows : Where Event-Driven Approach Meets Workflows to Support Digital Twins. In Proc. of the International Conference RuSCDays'18 – Russian Supercomputing Days, vol. 1, 2018, pp. 489–495.

Информация об авторах / Information about authors

Амир Басим Абдуламир АЛААСАМ является аспирантом кафедры системного программирования. Область его научных интересов включает в себя технологии

распределенных вычислительных систем, включая методы поточной обработки данных, облачные и туманные вычисления, системы потоков работ.

Ameer Basim Abdulameer ALAASAM is a postgraduate student of the Department of System Programming. The area of his scientific interests includes technologies of distributed computing systems, including methods of stream data processing, cloud and fog computing, workflow systems.

Глеб Игоревич РАДЧЕНКО – кандидат физико-математических наук, доцент, директор Высшей школы электроники и компьютерных наук, заведующий кафедрой Электронных вычислительных машин. Область научных интересов: технологий распределенной обработки данных, облачные и туманные вычислительные системы, проблемно-ориентированные распределенные вычислительные системы.

Gleb Igorevich RADCHENKO – Candidate of Physical and Mathematical Sciences, Associate Professor, Director of the School of Electronic Engineering and Computer Science, Head of the Department of Computers. Research interests: distributed data processing technologies, cloud and fog computing systems, problem-oriented distributed computing systems.

Андрей Николаевич ЧЕРНЫХ получил степень кандидата наук в Институте точной механики и вычислительной техники РАН. Он является профессором CICESE. В научном плане его интересуют многоцелевая оптимизация распределения ресурсов в облачной среде, проблемы безопасности, планирования, эвристики и метаэвристики, интернет вещей и т.д.

Andrei Nikolaevitch TCHERNYKH received his PhD degree at the Institute of Precision Mechanics and Computer Engineering of the Russian Academy of Sciences. He is holding a full professor position in computer science at CICESE. He is interesting in grid and cloud research addressing multiobjective resource optimization, both, theoretical and experimental, security, uncertainty, scheduling, heuristics and meta-heuristics, adaptive resource allocation, and Internet of Things.

Хосе Луис ГОНСАЛЕС-КОМПЕАН получил степень кандидата наук в области компьютерной архитектуры в Политехническом университете Каталонии в Барселоне. В настоящее время он является штатным профессором-исследователем в Центре перспективных исследований. Его направления исследований: облачные вычисления и контейнерные системы хранения, лингвистические архивные системы, безопасные и федеративные сети хранения.

José Luis GONZÁLEZ-COMPEÁN received his PhD degree in Computer architecture from UPC Universitat Politècnica de Catalunya, Barcelona. Currently he is full time professor researcher at Center of Research and Advanced Studies. His research lines: cloud computing and containerized storage systems, linguistic archival systems, secure and federated storage networks.

DOI: 10.15514/ISPRAS–2021–33(1)–6

Паттерны микросервисной архитектуры: многопрофильный обзор литературы

Х.А. Вальдивия, ORCID: 0000-0003-4270-0462 <zs15011636@estudiantes.uv.mx>
А. Лора-Гонсалес, ORCID: 0000-0002-0154-7361 <zs15011639@estudiantes.uv.mx>
К. Лимон, ORCID: 0000-0003-4654-636X <hlimon@uv.mx>
К. Кортес-Вердин, ORCID: 0000-0002-6453-180X <kcortes@uv.mx>
Х.О. Очаран-Эрнандес, ORCID: 0000-0002-2598-1445 <jocharan@uv.mx>

Университет Веракрузана
Мексика, 91000, Веракрус, Халапа

Аннотация. Микросервисная архитектура позволяет разрабатывать распределенные системы с использованием набора независимых, легко связываемых, и пригодных к совместному использованию сервисов, удовлетворяющих насущным потребностям облачных вычислений. Каждый микросервис может быть реализован в различных технологиях с использованием общих каналов связи, что приводит к созданию гетерогенных распределенных систем, демонстрирующих высокую масштабируемость, эксплуатационную надежность, производительность и интероперабельность. В настоящее время существует множество вариантов построения микросервисов; некоторые из них основаны на паттернах, задающих общие структуры для решения повторяющихся задач. Тем не менее, поскольку микросервисы являются сравнительно новым трендом, взаимосвязи между атрибутами качества, метриками и паттернами четко не определены, что является проблемой с позиции программной инженерии, поскольку понимание этих взаимосвязей является фундаментом правильного проектирования систем с использованием этой архитектуры. Настоящая статья направлена на расширение знаний о проектировании микросервисных систем, представляя многопрофильный систематический обзор литературы о паттернах, касающихся микросервисов, и связывая их с атрибутами качества и метриками.

Ключевые слова: микросервисы; распределенные системы; архитектурные паттерны, паттерны проектирования; атрибуты качества.

Для цитирования: Вальдивия Х.А., Лора-Гонсалес А., Лимон К., Кортес-Вердин К., Очаран-Эрнандес Х.О. Паттерны микросервисной архитектуры: многопрофильный обзор литературы. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 81-96. DOI: 10.15514/ISPRAS–2021–33(1)–6

Patterns Related to Microservice Architecture: a Multivocal Literature Review

J.A. Valdivia, ORCID: 0000-0003-4270-0462 <zs15011636@estudiantes.uv.mx>
A. Lora-González, ORCID: 0000-0002-0154-7361 <zs15011639@estudiantes.uv.mx>
X. Limón, ORCID: 0000-0003-4654-636X <hlimon@uv.mx>
K. Cortes-Verdin, ORCID: 0000-0002-6453-180X <kcortes@uv.mx>
J.O. Ocharán-Hernández, ORCID: 0000-0002-2598-1445 <jocharan@uv.mx>
Universidad Veracruzana,
Xalapa, Veracruz, México, 91000

Abstract. A Microservice Architecture enables the development of distributed systems using a set of highly cohesive, independent, and collaborative services, ready for current cloud computing demands. Each microservice can be implemented in different technologies, sharing common communication channels, which results in heterogeneous distributed systems that exhibit high scalability, maintainability, performance, and interoperability. Currently, there are many options to build microservices; some of them led by patterns that establish common structures to solve recurrent problems. Nevertheless, as microservices are an emerging trend, the relationship between quality attributes, metrics, and patterns is not clearly defined, which is a concern from a software engineering point of view, since such understanding is fundamental to correctly design systems using this architecture. This paper aims to extend the knowledge on the design of microservices-based systems by presenting a multivocal systematic literature review for microservices related patterns, tying them together with quality attributes and metrics, as can be found in academic and industry research.

Keywords: Microservices; Distributed Systems; Architectural Patterns; Design Patterns; Quality Attributes

For citation: Valdivia J.A., Lora-González A., Limón X., Cortes-Verdin K., Ocharán-Hernández J.O. Patterns Related to Microservice Architecture: A Multivocal Literature Review. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 81-96 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–6.

1. Введение

Микросервисная архитектура (MicroService Architecture, MSA) возникла в ответ на быстрые технологические изменения, проблемы расширяемости и масштабируемости, а также на необходимость сокращения времени поставки программного обеспечения. Эти проблемы послужили мотивацией нескольких технологических тенденций, основанных на облачных вычислениях [1-4]. MSA можно понимать, как набор небольших, разнородных, легко связываемых и распределенных сервисов, которые используют общие каналы связи для взаимодействия друг с другом. Успешные MSA-системы Netflix, Amazon и SoundCloud [5-7] заложили основу растущего интереса к этому архитектурному стилю, позиционируя его как альтернативу сервисно-ориентированной архитектуре (Service Oriented Architecture, SOA), из которой и происходит MSA [8]. Кроме того, реализация микросервисов обеспечивает мелкоструктурные сервисы, поддерживает гетерогенные технологии и обладает высокой отказоустойчивостью [9]; подобные свойства можно перевести в атрибуты качества, такие как удобство эксплуатации, переносимость и надежность.

Атрибуты качества имеют первостепенное значение при разработке программного обеспечения, поскольку уровень их достижимости принципиально влияет на успех программного проекта. Выбор архитектурного стиля, наряду с архитектурными решениями, принятыми в процессе проектирования, обеспечивает достижение атрибутов качества [10]. Атрибут качества – это измеримое и поддающееся проверке свойство системы, которое помогает определить, насколько хорошо система отвечает потребностям заинтересованных лиц [11]. Решения архитектурных задач должны быть сбалансированными, следует соблюдать баланс между набором атрибутов качества, определенных для системы, и преимуществами предлагаемого решения [12].

С атрибутами качества связаны паттерны, в которых определяются наборы многократно используемых структур для решения схожих задач в разных контекстах и обеспечивается единый словарь для сообщества разработчиков программного обеспечения [13]. Структуры паттернов представляют собой расширение архитектурных элементов, используемых для построения системы; аналогично, комбинирование нескольких паттернов облегчают разработку более сложных систем [14].

Использование паттерна направлено на решение задачи или улучшение атрибута качества, связанного с требованиями [10]. Для достижения требуемого атрибута качества можно использовать один или несколько шаблонов; разные комбинации могут обеспечить разные уровни качества, но определение правильной комбинации может оказаться сложной задачей. Метрики уменьшают сложность количественной оценки свойств системы, предоставляя числовое значение как меру того, насколько близок атрибут к желаемому уровню [15]. Значение, полученное с применением метрики, позволяет сравнивать паттерны и выбирать правильный шаблон для достижения намеченных целей; также снижается уровень субъективности при оценке влияния выбранного решение на другие атрибуты качества.

Появление паттернов и их принятие для практического использования является результатом основанного на опыте высокого уровня зрелости в области разработкт программного обеспечения работы. Например, широко известная работа Бушманна (Frank Buschmann) и др. [13] явилась результатом опыта работы авторов в компании Siemens. Они создали каталог решений для совершенствования процесса архитектурного проектирования. Аналогично, публикация [12] подытоживает предыдущий опыт работы авторов, предоставляя набор объектно-ориентированных паттернов [16]. Что касается микросервисной архитектуры, то, насколько нам известно, коллекции паттернов ограничены, поскольку микросервисы являются новым трендом. Однако мы можем найти исследования (например, [17-19] с многочисленными архитектурными паттернами, предоставляющими некоторый диапазон альтернатив. Среди них только в последней работе обеспечивается соответствие между атрибутами качества и паттернами. Поэтому трудно идентифицировать подходящий шаблон, соответствующий заданному атрибуту качества. Кроме того, сложно выявить атрибуты качества, на которые влияет выбранный шаблон.

Отметим также, что нахождение доступных паттернов для микросервисов является сложной задачей, поскольку связанная с паттернами информации скудна и рассеяна по белой и серой литературе, то есть академическим и промышленным публикациям соответственно.

Цель нашей работы состоит в том, чтобы представить существующую в настоящее время коллекцию шаблонов MSA, соответствующих атрибутам качества с сопутствующими метриками, которые, в свою очередь, указывают на уровень достижения атрибута качества конкретным паттерном. Такие шаблоны были получены путем многоаспектного анализа как белой, так и серой литературы. Наша работа будет полезна специалистам-практикам, нуждающимся в понимании того, какое влияние на процесс разработки оказывает выбор того или иного шаблона MSA, а также исследователям для дальнейшей работы в этой области. В нашу задачу не входит классификация выявленных шаблонов, мы стремимся лишь связать шаблоны с соответствующими атрибутами качества и метриками.

Данная статья является развитием предыдущей работы [20], в которой представлена коллекция паттернов 2014-2018 гг., полученная на основе систематического анализа литературы, то есть рассматривалась только белая литература. Новый обзор охватывает и паттерны 2019 года, и в нем дополнительно рассматриваются публикации категории серой литературы с 2014 по 2019 год на основе использования должным образом скорректированной методологии многопрофильного обзора литературы (Multivocal Literature Review, MLR).

В нашем MLR объединяются академические и производственные знания, обеспечивая расширенное представление о паттернах от практиков в области MSA. Обсуждение

паттерном с точек зрения отрасли и академии обеспечит более четкое понимание того, для чего используются шаблоны, а также роли атрибутов качества и метрик, используемых вместе с такими шаблонами.

Статья организована следующим образом. В разд. 2 излагается методология, используемая для многопрофильного обзора литературы и включающая вопросы, которыми мы задаемся при анализе публикаций, ограничения и критерии выбора источников. В разд. 3 представлены результаты, полученные путем применения методологии и включающие ответы на наши вопросы, а также нашу интерпретацию этих данных. В разд. 4 приводится сводка наиболее важных результатов нашей работы и описываются направления будущих исследований.

2. *Регламент работы над обзором*

В нашей работе используется подход MLR для использования данных серой литературы с целью предоставления полезной информации исследователям из производственного сообщества. Мы использовали рекомендации [21] для использования серой литературы и создания многопрофильных обзоров литературы по программной инженерии. Предлагаемые ниже этапы основаны на работе [22], где содержатся основные принципы систематических обзоров литературы по программной инженерии. Первым этапом является планирование обзора с целью выявления потребности в обзоре и уточнения вопросов, на которые требуется получить ответы. Второй этап работы над обзором включает в себя выработку стратегии поиска и критериев выбора источников, выбор источника, оценку качества исследований, представленных серой литературой, извлечение данных и синтез данных. Наконец, на третьем этапе описываются результаты в виде ответов на основные вопросы обзора.

2.1 Планирование обзора

Основной мотивацией нашей работы является количественное и качественное уточнение информации о паттернах, представленной в нашем предыдущем исследовании, на основе отраслевых знаний, углубленное изучение взаимосвязей между паттернами и атрибутами качества и исследование метрик, связанных с паттернами и используемых в индустрии. Обзор основан на трех исследовательских вопросах (Research Question, RQ) о паттернах проектирования в микросервисной архитектуре:

RQ1. Какие паттерны выявляются в микросервисных системах?

RQ2. Какие атрибуты качества связаны с паттернами, используемыми при разработке микросервисных систем?

RQ3. Какие метрики используются для количественной оценки атрибутов качества, связанных с паттернами микросервисов?

Вопрос **RQ1** направлен на непосредственное выявление паттернов микросервисов. **RQ2** является в нашем исследовании основным вопросом, поскольку ответы на этот вопрос позволяют получить детальную информацию об атрибутах качества на основании уникальных характеристик каждого паттерна, что помогает объяснить включение шаблонов для конкретных задач. Вопрос **RQ3** дает возможность исследовать метрики для количественного измерения уровня достижимости атрибутов качества, помогая лучше понять связи между паттернами MSA и атрибутами качества.

3. *Ведение процесса*

Процесс подготовки обзора включает ручной и автоматический поиск, изучение оценок качества серой литературы, извлечение данных, а также синтез данных с использованием метаэтнографического метода взаимной транслокации [23], который улучшает интерпретацию качественной информации.

3.1 Стратегия поиска и критерии выбора публикаций

Для сбора соответствующей информации в белой литературе использовался автоматический поиск в следующих службах индексирования: Scopus, Science Direct, Springer Link, IEEE Xplore и ACM. В то же время, для поиска в серой литературе использовалась поисковая система Google.

Для ограничения результатов применялись следующие критерии:

- критерии включения:
  - публикации с 2014 по 2019 гг.;
  - язык написания английский;
  - тип публикации – в серую литературу включались блоги для повышения качества источников и упрощения оценки;
- критерий исключения:
  - книга или глава из книги.

В дополнение к автоматическому поиску в качестве последнего шага применялся ручной поиск с использованием обратного метода снежного кома [24].

3.2 Выбор из источников

Процесс отбора проходил в пять этапов:

1) исключение по названию;
2) включение по обратному методу снежного кома для белой литературы;
3) исключение по контрольному списку для белой литературы / исключение по оценке качества для серой литературы;
4) исключение по ответу на исследовательские вопросы;
5) включение по обратному методу снежного кома для серой литературы.

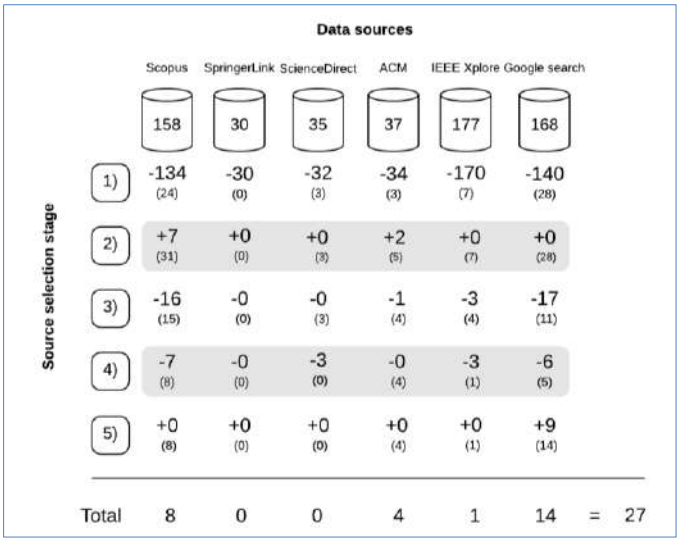


Рис. 1. Число работ для каждого источника на каждом этапе выбора источников. Каждое число показывает, насколько увеличилось или уменьшилось число статей, происходящих из данного источника представляет на данном этапе, а число в скобках показывает общее число статей из данного источника в конце данного этапа

Fig. 1. Data sources frequency against source selection stages. Each intersection represents the result between source selection stage and data source, and the numbers enclosed in parentheses highlight the remaining studies after each selection stage

Мы скорректировали процесс отбора, чтобы следовать рекомендациям по использованию серой литературы [21]. Оценка качества производилась для уменьшения предвзятости и валидации источника для серой литературы.

Был выполнен автоматический поиск по пяти источникам информации и в поисковой системе, в результате чего были получены 605 результатов, удовлетворяющих фильтрам стратегии поиска, как показано на рис. 1. После этого исключение на этапе 1 отбросило все документы с названиями, не связанными с темой исследования, что уменьшило количество публикаций до 65. Затем, на втором этапе в белой литературе мы нашли дополнительные девять публикаций. На этапе 3 после исключения по контрольному списку и оценки качества общее количество исследований было сокращено до 37. Наконец, на четвертом этапе было проведено полное чтение работ, и ответ хотя бы на один исследовательский вопрос был проверялся по другому контрольному списку, что сократило число работ до 18. С другой стороны, обратный метод снежного кома для серой литературы включил девять публикаций, что дало в общей сложности 27 источников для получения информации.

3.3 Оценка качества

В серой литературе отсутствует контроль качества загружаемых материалов и отсутствуют учреждения, проверяющие достоверность и объективность содержания. Следовательно, исследователи должны гарантировать качество включенных данных. Для этого мы применили контрольный список, предложенный в [21], к каждому из источников из серой литературы.

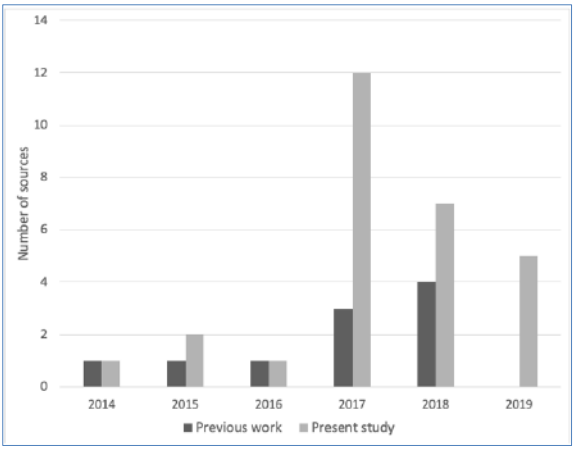


Рис. 2. Число работ, посвященных паттернам микросервисов; показаны также результататы нашего предыдущего исследования [20], которое не охватывает 2019 год и не использует серую литературу
Fig. 2. Microservice patterns trend, showing also results from our previous study [20], which does not cover 2019 and does not include grey literature.

4. Результаты и обсуждение

После завершения исследования мы можем составить представление об интересе к паттернам микросервисов в индустрии и академии. Число выбранных источников по годам приведено на рис. 2. Интересно, что в нашей предыдущей работе [20] с учетом только белой литературы больше всего работ было в 2018 году, но с учетом серой литературы пик интереса отмечается в 2017 году. Кроме того, как показано на рис. 3, включение серой литературы обеспечивает почти половину от общего числа публикаций, хотя учитывались только блоги. Между тем, в белой литературе наиболее частыми форматами были журнальные статьи и материалы конференций.

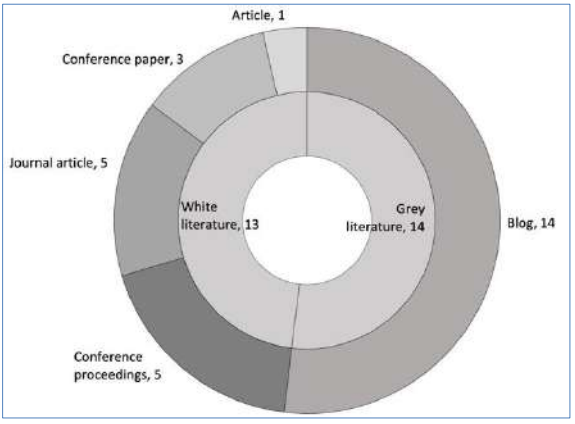


Fig. 3. Типы проанализированных публикаций
Fig. 3. Literature type identified in the sources analyzed..

В окончательно отобранных работах было выявлено 54 паттерна. Это довольно большое число, но следует отметить, что некоторые из паттернов представляют собой вариации или специализированные реализации других паттернов. Что касается атрибутов качества, мы использовали в качестве ориентира модель качества программного продукта, определенную в стандарте ISO/IEC 25010 [50]. Эта модель включает восемь характеристик качества. Однако мы обнаружили только шесть характеристик, имеющих отношение к паттернам микросервисов. Как видно на рис. 4, качественными характеристиками в порядке убывания числа упоминаний являлись удобство обслуживания, надежность, безопасность, эффективность, совместимость и переносимость. Никакой связи паттернов с атрибутами функциональной пригодности и удобства использования нами не обнаружено. Некоторые паттерны связаны с более чем одним атрибутом.

Рис. 4. Число упоминаний атрибутов качества, относящихся к выявленным паттернам
Fig. 4. Quality attributes frequency related to the identified patterns.

Табл. 1. Паттерны микросервисов, сгруппированные по основным преимуществам, и связанные с ними атрибуты качества
Table 1. Microservice patterns grouped by benefits and related quality attributes

Название группы	***Атрибут качества***	***Источник***
Постоянное хранение данных / Data Persistency		
Прокси локальной базы данных / Local Database Proxy	Удобство обслуживания и надежность	[25], [26]

Маршрутизатор на основе локального разделения данных / Local Sharding-based Router	Удобство обслуживания и эффективность	[25], [26]
Разделение ответственности за команды и запросы / Command and Query Responsibility Segregation	Удобство обслуживания, эффективность, надежность и безопасность	[27]–[29]
Порождение событий / Event Sourcing	Безопасность* и эффективность *	[27], [28], [30]
Масштабируемое хранилище / Scalable Store	Удобство обслуживания и надежность	[19]
Коммуникации		
Безопасный канал Secure Channel	Безопасность	[31]
Изменение зависимости уровня кода на зависимость уровня сервиса / Change Code Dependency to Service Call	Совместимость	[18], [32]
Реестр сервисов / Service Registry	Переносимость, удобство обслуживания и надежность	[18], [19], [30], [32]–[34]
Обнаружение сервисов / Service Discovery	Безопасность и надежность	[19], [35]
Локатор сервисов / Service Locator	Безопасность и удобство обслуживания *	[36]
Уведомление о событии / Event Notification	Надежность	[36]
Клиент реестра сервисов / Service Registry Client	Удобство обслуживания и надежность	[33]
Интеграция на основе REST / REST Integration	Удобство обслуживания и совместимость*	[18]
Конкурирующие потребители / Competing Customers	Удобство обслуживания	[25]
Каналы и фильтры / Pipes and Filters	Надежность и удобство обслуживания	[25]
Асинхронный обмен сообщениями / Asynchronous Messaging	Надежность, удобство обслуживания и эффективность	[19], [27], [28], [30], [34], [37]
Запрос-реакция / Request-Reaction	Удобство обслуживания*	[30]
Точка входа		
Шлюз API / API Gateway	Удобство обслуживания, надежность, эффективность и безопасность	[19], [27], [28], [30], [35], [38]–[41]
Отдельная серверная часть для каждого интерфейса / Backend for Frontend	Совместимость и удобство обслуживания	[18], [19], [30], [32], [35], [42]
Сервис аутентификации / Auth-Service	Надежность	[39]
Привратник / Gatekeeper	Безопасность	[25]
Распространение		
Душитель / Strangler	Совместимость	[28], [30], [43]
Уровень защиты от повреждений / Anti-Corruption Layer	Совместимость	[44]
Энтернализованная конфигурация /	Безопасность	[35]

Externalized Configuration		
Самодостаточность сервисов / Self-Containment of Services	Удобство обслуживания*	[18]
Контейнер / Container	Удобство обслуживания и надежность	[18], [19]
Развертывание кластера и оркестровка контейнеров / Deploy Cluster and Orchestrate Containers	Надежность и удобство обслуживания	[18]
DevOps микросервисов / Microservices DevOps	Удобство обслуживания *	[18], [19]
База данных является сервисом / Database is the Service	Удобство обслуживания и надежность	[19]
Разрешение непрерывной интеграции / Enable Continuous Integration	Удобство обслуживания	[19]
Самодостаточные системы / Self-Contained Systems	Удобство обслуживания и надежность	[30]
Отказоустойчивость		
Отсек / Bulkhead	Надежность	[45]
Автоматический выключатель / Circuit Breaker	Удобство обслуживания и надежность	[19], [28], [34]
Дополнения		
Очередь с приоритетами / Priority Queue	Удобство обслуживания	[25], [26]
Кеш результатов / Results Cache	Эффективность	[19], [32]
Кеш страниц / Page Cache	Эффективность	[19], [32]
Хранилище «ключ-значение» / Key Value Store	Безопасность и надежность	[19], [32]
Идентификатор корреляци / Correlation Id	Удобство обслуживания	[32]
Агрегатор журналов / Log Aggregator	Удобство обслуживания и эффективность работы	[19], [32]
Балансировщик нагрузки/Балансировка нагрузки / Load Balancer/Load-Balancing	Надежность и удобство обслуживания	[19], [33], [35], [41]
Посредник / Ambasador	Надежность и удобство обслуживания	[46]
Прицеп / Sidecar	Удобство обслуживания	[47]
Агрегатор шлюзов / Gateway Aggregator	Удобство обслуживания	[48]
Разгрузка шлюзов / Gateway Offloading	Удобство обслуживания и надежность	[49]
Асинхронный запрос / Asynchronous Query	Надежность	[36]
Маркер асинхронного завершения / Asynchronous Completion Token	Надежность	[36]
Пограничный сервер / Edge Server	Удобство обслуживания и надежность	[33]
Внутренний балансировщик нагрузки / Internal Load Balancer	Удобство обслуживания и надежность	[33]
Внешний балансировщик нагрузки / External Load Balancer	Удобство обслуживания и надежность	[33]

Проверка работоспособности / Health Check	Удобство обслуживания	[19]
Монитор / Monitor	Удобство обслуживания	[19]
Контракты, ориентированные на потребителя / Consumer-Driven Contracts	Удобство обслуживания *	[30]
Толерантный читатель / Tolerant reader	Удобство обслуживания *	[30]
Агрегатор / Aggregator	Удобство обслуживания	[28]
* В публикации явно не говорится, но следует из текста		

4.1 Анализ RQ1

Что касается паттернов, выявляемых в системах микросервисов, в табл. 1 представлены паттерны, сгруппированные по свойственным им преимуществам; паттерны указываются в хронологическом порядке по возрастанию времени. Большинство паттернов было выявлено в литературе, посвященной микросервисам, однако некоторые паттерны связаны также с SOA, которую можно считать более общим видом архитектуры распределенных служб [8]. К таким паттернам относятся, например, *проверка работоспособности*, *асинхронный обмен сообщениями*, *обнаружение сервисов*, автоматический выключатель и *монитор*. Это неудивительно, поскольку микросервисы можно понимать, как эволюцию SOA. SOA – это нецентрализованный архитектурный стиль, основанный на коммуникации сервисов для обеспечения требуемых функциональных возможностей и допускающий использование сервисов сторонних производителей [51]. Однако у SOA имеются проблемы с коммуникациями, промежуточным программным обеспечением и уровнем модульности; для смягчения этих проблем в качестве альтернативы были разработаны микросервисы [8].

Наиболее часто выявляемые паттерны связаны с решением коммуникационных задач и обеспечением координации между сервисами – например, *асинхронный обмен сообщениями*, *реестр сервисов* и *запрос-реакция*. В литературе чаще всего обнаруживаются паттерны *шлюз API*, *отдельная серверная часть для каждого интерфейса* и *реестр сервисов*. Поскольку термины «паттерн проектирования» и «архитектурный паттерн» у авторов четко не различаются, некоторые паттерны были у них отнесены к обеим категориям, а в некоторых случаях проходили под общим названием «паттерн». Например, в категории просто паттернов были обнаружены *разрешение непрерывной интеграции* [19] и *контейнер* [19], [25]. *Непрерывная интеграция* – это прием, связанный с гибкой (agile) разработкой программного обеспечения. Этот паттерн позволяет смягчить проблему так называемой «интеграции большого взрыва» (big-bang integration) с использованием инструментов для интеграции кода, запуска наборов тестов и создания артефактов развертывания [52]. Следование этому паттерну подразумевает деятельность по разработке, не связанную напрямую с системными структурами. Уровень абстракции *контейнера* может быть ограничен применяемой технологией, поскольку речь идет о специализированной реализации архитектурного стиля виртуализации [51].

В ходе обобщения данных нами были выявлены пять основных преимуществ использования паттернов. Эти преимущества позволили нам отнести каждый шаблон к одной из следующих групп: постоянное хранение данных, коммуникации, точка входа, распространение, отказоустойчивость и дополнительные компоненты. Каждая группа демонстрирует общие преимущества паттернов и облегчает интерпретацию результатов, но эти группы не предназначены для классификации паттернов.

4.2 Анализ RQ2

Как отмечалось выше, вопрос RQ2 касается атрибутов качества, связанных с паттернами, используемыми в микросервисах. Атрибуты качества, выявленные в литературе, заведомо связаны с MSA. В табл. 1 представлены атрибуты качества, свойственные каждому выявленному паттерну. Для сопоставления паттернов и атрибутов качества мы непосредственно в публикациях определяли атрибуты, связанные с паттернами. Однако не все найденные паттерны имели документированную связь с атрибутами качества; в этих случаях мы анализировали характеристики паттернов и консультировались с дополнительным исследователем, имеющим опыт работы с микросервисами.

На рис. 4 первые два наиболее часто встречающихся атрибута качества составляют почти 75% от общего количества, а остальные четыре атрибута – всего 25% совместно. Высокая частота упоминаний удобства обслуживания и надежности может рассматриваться как принципа быстрого и простого развертывания микросервисных систем, а также внимания к обеспечению функциональных возможностей, удовлетворяющих требованиям качества. Однако не все аспекты MSA оказались связанными с паттернами. Например, характеристика удобства обслуживания была наиболее часто встречающимся атрибутом качества, но ни в одном описании этого паттерна не упоминалась подхарактеристика тестируемости, несмотря на то, что тестирование является одним из важнейших аспектов MSA. Кроме того, некоторые паттерны связаны с тремя или четырьмя атрибутами качества; это может поставить под сомнение возможность полного удовлетворения каждого атрибута качества с учетом требуемых компромиссов.

4.3 Анализ RQ3

Наконец, вопрос RQ3 касается метрик, используемых для количественной оценки атрибутов качества, связанных с паттернами микросервисов. В литературе удалось выявить следующие метрики, связанные с оценкой атрибутов качества в паттернах:

- время (эффективность);
- процент принятых запросов (интероперабельность);
- число файлов, которые нужно изменить (удобство обслуживания);
- энергопотребление (эффективность).

Были выявлены три особых аспекта, касающихся времени: время ответа для выполнения определенного количества запросов в секунду, время взаимодействия в миллисекундах (время до того, как пользователь сможет взаимодействовать со своим графическим интерфейсом) и время ответа в миллисекундах. Что касается времени, в 2014 г. в [26] были проанализированы паттерны *прокси локальной базы данных*, *маршрутизатора на основе локального разделения данных* и *очереди с приоритетами*. Эти три шаблона положительно влияют на скорость выполнения запросов на чтение и запись данных. Первый шаблон больше подходит для запросов на чтение, второй – для записи, а последний оказывает умеренно положительное влияние на оба вида запросов. Разумно комбинировать последний паттерн с одним из первых двух. Кроме того, в этих трех паттернах выбор алгоритмов не дает большой разницы в скорости, поскольку ключевым моментом является принятие правильного решения в соответствии с паттерном.

В 2017 г. в [39] паттерны *самодостаточности* и *шлюза API* были оценены на предмет удобства обслуживания. Первый паттерн лучше второго с точки зрения числа файлов, которые нужно изменить, и процентного отношения приемлемых модификаций. У шлюза API имеются недостатки при наличии высоких нагрузок. Что касается числа миллисекунд, при небольшом количестве запросов оба шаблона показывали одно и то же время, но при большом числе запросов *шлюз API* требует немного больше времени.

В 2018 г. в [25] паттерны *очереди с приоритетами*, *прокси локальной базы данных*, *маршрутизатора на основе локального разделения данных*, *каналов и фильтров*, *конкурирующих потребителей* и *привратника* оценивались по потреблению энергии. Энергопотребление измерялось на уровне процессов с использованием Power-API, интерфейса прикладного программирования, написанного на Java API для отслеживания потребляемой энергии. В целом, все пять паттернов ведут к сокращению потребления энергии, измеряемой в килоджоулях, и времени, измеряемого в миллисекундах, повышая эффективность работы микросервисов по сравнению с монолитными системами. Паттерн *каналы и фильтры* сокращает время работы и снижают потребление энергии. К удивлению, этот паттерн не дает тех же преимуществ в монолитных системах. Точно так же эти преимущества могут быть получены с использованием паттернов *привратника* и *конкурирующих потребителей*, но отдельная реализация привратника может отрицательно повлиять на время отклика и потребление энергии. Комбинация *прокси локальной базы данных* и *конкурирующих потребителей* отрицательно влияет на время отклика, как и комбинация *конкурирующих потребителей*, *конкурирующих потребителей* и *каналов и фильтров*.

К сожалению, анализ публикаций 2019 года и учет отраслевой информации не позволили добавить какие-либо новые метрики к тому, что мы изложили в [20]. В соответствии с данными серой литературы, при выборе паттерна могут вообще не применяться количественные измерения, хотя можно усомниться в том, что в этом деле можно обойтись только опытом. Как следствие, можно сделать вывод о низком уровне зрелости метрик MSA в серой литературе.

5. Заключение

Проведение MLR помогло расширить список паттернов, представленных в нашей предыдущей работе [20]. При включении в обзор публикаций серой литературы было обнаружено 20 новых паттернов. Кроме того, в некоторых источниках серой литературы представлена дополнительная информация о паттернах, поскольку авторы не ограничены количеством страниц или слов. Например, Microsoft подробно описывает следующие аспекты каждого паттерна: контекст, проблему, решение, спорные моменты, соображения и, в некоторых случаях, примеры с кодом.

Изучение «серой» литературы требует дополнительной работы, например, привлечения контрольного списка, предложенного в [21] для оценки качества. Эту оценку необходимо производить для каждой выбранной публикации серой литературы. Некоторые вопросы в контрольном списке являются субъективными, а другие требуют дополнительного поиска. Еще один недостаток – огромное количество результатов поиска в серой литературе. На первом этапе поиска было найдено 2 640 000 результатов, хотя использовалась поисковая строка. Чтобы проанализировать наиболее релевантные для исследования результаты, мы использовали алгоритм ранжирования Google. Однако нерелевантный материал все же предоставлялся, например, ссылки на покупку книг, курсов или семинаров.

В белой литературе пояснения по поводу паттернов были краткими и не прямыми, авторы обычно не объясняют контекст и метод, использованный для получения информации. Напротив, в серой литературе авторы были более прямыми: если обсуждаемый вопрос требовал какого-либо пояснения контекста, авторы приводили ссылки на другие ресурсы, предоставляющие необходимую информацию. Другой момент заключается в том, что в анализируемой литературе уровень абстракции термина «паттерн» может быть неясен. Некоторые из описываемых паттернов не являются, строго говоря, паттернами, например, *разрешение непрерывной интеграции* [19] и *контейнер* [19], [25].

Таким образом, включение в обзор серой литературы принесло такие преимущества, как добавление в каталог новых паттернов, но также добавилась и дополнительная работа для

обеспечения качества. Несмотря на увеличение количества паттернов, представленный список не следует считать окончательным. В будущей работе ожидается включение большего числа типов публикаций, например, книг, материалов конференций и фирменных описаний. Кроме того, мы изучаем различные методы уменьшения количества нерелевантных публикаций при использовании серой литературы.

Список литературы / References

[1] Д.А. Грушин, Н.Н. Кузюрин. О задаче эффективного управления вычислительной инфраструктурой. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 123-142. DOI: 10.15514/ISPRAS-2018-30(6)-7 / D.A. Grushin and N.N. Kuzyurin. On Effective Scheduling in Computing Clusters. Programming and Computer Software, vol. 45, no. 7, 2019, pp. 398–404.

[2] Б.М. Шабанов, О.И. Самоваров. Принципы построения межведомственного центра коллективного пользования общего назначения в модели программно-определяемого ЦОД. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 7-24. DOI: 10.15514/ISPRAS-2018-30(6)-1 / B.M. Shabanov and O.I. Samovarov. Building the Software-Defined Data Center. Programming and Computer Software, vol. 45, no. 8, 2019, pp. 458–466.

[3] Вл.В. Воеводин, Н. Н. Попова. Инфраструктура суперкомпьютерных технологий. Программирование, том 45, no 3, 2019, стр. 6–13 / V.V Voevodin and N.N. Popova. Infrastructure of Supercomputing Technologies. Programming and Computer Software, vol. 45, no. 3, 2019, pp. 89–95.

[4] А.П. Крюков, А.П. Демичев. Децентрализованные хранилища данных: технологии построения. Программирование, том 44, no. 5, 2018, стр. 12–30. / A.P. Kryukov and A.P. Demichev. Decentralized Data Storages: Technologies of Construction. Programming and Computer Software, vol. 44, no. 5, 2018, pp. 303–315.

[5] T. Hoff. Amazon Architecture. High Scalability, 2007. [Online]. Available: http://highscalability.com/blog/2007/9/18/amazon-architecture.html. [Accessed: 07-Nov-2018].

[6] Netflix Technology Blog. Netflix Conductor: A microservices orchestrator. Netflix Techblog, 2016. [Online]. Available: https://medium.com/netflix-techblog/netflix-conductor-a-microservices-orchestrator-2e8d4771bf40. [Accessed: 07-Nov-2018].

[7] P. Calçado. Building Products at SoundCloud – Part I: Dealing with the Monolith. Developers Soundcloud, 2014. [Online]. Available: https://developers.soundcloud.com/blog/building-products-at-soundcloud-part-1-dealing-with-the-monolith. [Accessed: 03-Nov-2018].

[8] S. Newman, Building Microservices: Designing Fine-Grained Systems, 1st ed. O'Reilly Media, 2015, 280 p.

[9] J. Lewis and M. Fowler. Microservices – A definition of this new architectural term. Martinfowler.Com, 2014. [Online]. Available: http://martinfowler.com/articles/microservices.html.

[10] P. Clements, R. Kazman, and M. Klein. Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley, 2001, 368 p.

[11] L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice, 3rd ed. Addison-Wesley Professional, 2012, 624 p.

[12] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc., 1995, 416 p.

[13] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. Pattern-Oriented Software Architecture – Volume 1: A System of Patterns. Wiley Publishing, 1996, 476 p.

[14] D.C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. Pattern-Oriented Software Architecture, Volume 2: Patterns for Concurrent and Networked Objects. Wiley, 2000, 666 p.

[15] K.H. Möller and D.J. Paulish. Software metrics: a practitioner's guide to improved product development. IEEE Computer Society Press, 1993, 257 p.

[16] P. Clements, F. Bachmann et al. Documenting Software Architectures: Views and Beyond, 2nd ed. Addison-Wesley Professional, 2010, 592 p.

[17] D. Taibi, V. Lenarduzzi, and C. Pahl. Architectural Patterns for Microservices: A Systematic Mapping Study. In Proc. of the 8th International Conference on Cloud Computing and Services Science, vol. 1, 2018, pp. 221–232.

[18] F. Osses, G. Márquez, and H. Astudillo. Exploration of academic and industrial evidence about architectural tactics and patterns in microservices. In Proc. of the 40th International Conference on Software Engineering, 2018, pp. 256–257.

[19] G. Marquez and H. Astudillo. Actual Use of Architectural Patterns in Microservices-Based Open Source Projects. In Proc. of the 25th Asia-Pacific Software Engineering Conference (APSEC), 2018, pp. 31–40.
[20] J.A. Valdivia, X. Limón, and K. Cortes-Verdin. Quality attributes in patterns related to microservice architecture: A Systematic Literature Review. In Proc. of the 7th International Conference in Software Engineering Research and Innovation (CONISOFT), 2019, pp. 181–190.
[21] V. Garousi, M. Felderer, and M. V. Mäntylä. Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and Software Technology, vol. 106, 2019, pp. 101–121.
[22] B. Kitchenham and S. Charters. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE-2007-01, Keele University, University of Durham, 2007.
[23] G.W. Noblit and R.D. Hare. Meta-ethnography: Synthesizing qualitative studies. Counterpoints, vol. 44, 1988, pp. 93-123.
[24] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proc. of the 18th International Conference on Evaluation and Assessment in Software Engineering, 2014, article no. 38.
[25] F. Khomh and S.A. Abtahizadeh. Understanding the impact of cloud patterns on performance and energy consumption. Journal of Systems and Software, vol. 141, 2018, pp. 151–170.
[26] G. Hecht, B. Jose-Scheidt, C. De Figueiredo, N. Moha, and F. Khomh. An Empirical Study of the Impact of Cloud Patterns on Quality of Service (QoS). In Proc. of the IEEE 6th International Conference on Cloud Computing Technology and Science, 2014, pp. 278–283.
[27] K. Aram. A Microservices implementation journey – Part 1. Medium Koukia blog, 2018. [Online]. Available: https://koukia.ca/a-microservices-implementation-journey-part-1-9f6471fe917. [Accessed: 12-Jan-2020].
[28] K. Sahiti. Everything You Need to Know About Microservices Design Patterns. Edureka blog, 2019. [Online]. Available: https://www.edureka.co/blog/microservices-design-patterns. [Accessed: 12-Jan-2020].
[29] Microsoft. Responsibility Segregation (CQRS) pattern. Azure Architecture Center, 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs. [Accessed: 13-Jan-2020].
[30] J. Bogner, J. Fritzsch, S. Wagner, and A. Zimmermann. Assuring the Evolvability of Microservices: Insights into Industry Practices and Challenges. In Proc. of the IEEE International Conference on Software Maintenance and Evolution (ICSME), 2019, pp. 546-556.
[31] A.K. Dwivedi and S.K. Rath. Incorporating Security Features in Service-Oriented Architecture using Security Patterns. ACM SIGSOFT Software Engineering Notes, vol. 40, no. 1, 2015, pp. 1-6.
[32] K. Brown and B. Woolf. Implementation patterns for microservices architectures. In Proc. of the 23th International Conference on Pattern Languages of Programs, 2016, 35 p.
[33] A. Balalaie, A. Heydarnoori, P. Jamshidi, D. A. Tamburri, and T. Lynn. Microservices migration patterns. Software Practice and Experience, vol. 48, no. 11, 2018, pp. 2019-2042.
[34] G. Márquez and H. Astudillo. Identifying availability tactics to support security architectural design of microservice-based systems. In Proc. of the 13th European Conference on Software Architecture, vol. 2, 2019, pp. 123-129.
[35] K.A. Torkura, M.I.H. Sukmana, F. Cheng, and C. Meinel. Leveraging Cloud Native Design Patterns for Security-as-a-Service Applications. In Proc. of the 2nd IEEE International Conference on Smart Cloud, 2017, pp. 90–97.
[36] G. Rodríguez, J. A. Díaz-Pace, and Á. Soria. A case-based reasoning approach to reuse quality-driven designs in service-oriented architectures. Information Systems, vol. 77, 2018, pp. 167–189.
[37] Microsoft. Designing a microservice-oriented application. .NET Microservices: Architecture for Containerized .NET Applications, 2018. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/microservice-application-design. [Accessed: 12-Jan-2020].
[38] C. Richardson. Building Microservices: Using an API Gateway. Nginx blog, 2015. [Online]. Available: https://www.nginx.com/blog/building-microservices-using-an-api-gateway/. [Accessed: 12-Jan-2020].
[39] H. Harms, C. Rogowski, and L. Lo Iacono. Guidelines for adopting frontend architectures and patterns in microservices-based systems. In Proc. of the 11th Joint Meeting on Foundations of Software Engineering, 2017, pp. 902-907.
[40] Microsoft. Gateway Routing pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/gateway-routing. [Accessed: 13-Jan-2020].

[41] D. Müssig, R. Stricker, J. Lässig, and J. Heider. Highly Scalable Microservice-based Enterprise Architecture for Smart Ecosystems in Hybrid Cloud Environments. Proc. In Proc. of the International Conference on Enterprise Information Systems, vol. 1, 2017, pp. 454-459.
[42] Microsoft. Backends for Frontends pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/backends-for-frontends. [Accessed: 12-Jan-2020].
[43] Microsoft. Strangler pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/strangler. [Accessed: 13-Jan-2020].
[44] Microsoft. Anti-Corruption Layer pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/anti-corruption-layer. [Accessed: 12-Jan-2020].
[45] Microsoft. Bulkhead pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/bulkhead. [Accessed: 13-Jan-2020].
[46] Microsoft. Ambassador pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/ambassador. [Accessed: 12-Jan-2020].
[47] Microsoft. Sidecar pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar. [Accessed: 13-Jan-2020].
[48] Microsoft. Gateway Aggregation pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/gateway-aggregation. [Accessed: 13-Jan-2020].
[49] Microsoft. Gateway Offloading pattern. Azure Architecture Center, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/patterns/gateway-offloading. [Accessed: 13-Jan-2020].
[50] ISO/IEC 25010: 2011 Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models. ISO, 2011.
[51] R.N. Taylor, N. Medvidovic, and E. M. Dashofy. Software Architecture: Foundations, Theory, and Practice. Wiley Publishing, 2009, 750 p.
[52] C. Carneiro and T. Schmelmer. Microservices from Day One: Build Robust and Scalable Software from the Start. Apress, 2016, 268 p.

Информация об авторах / Information about authors

Хосе Али ВАЛЬДИВИЯ, бакалавр в области программной инженерии. Область научных интересов: архитектура программного обеспечения, распределенные системы, паттерны проектирования.

José Ali VALDIVIA, Bachelor of Software Engineering. Research interests: Software architecture, distributed systems, design patterns.

Алонсо ЛОРА-ГОНСАЛЕС, бакалавр в области программной инженерии, профессиональный разработчик программного обеспечения. Область научных интересов: программные архитектуры, паттерны проектирования и распределенные системы.

Alonso LORA-GONZALEZ, Bachelor degree in Software Engineering. Professional Software developer. Research interests: Software Architectures, Design patterns and Distributed Systems.

Ксавье ЛИМОН, кандидат наук в области искусственного интеллекта, доцент факультета статистики и информатики. Область научных интересов: распределенные системы, архитектуры программного обеспечения, многоагентные системы, машинное обучение.

Xavier LIMÓN, Doctor of Artificial Intelligence, Associate Professor of the Statistics and Informatics Faculty. Research interests: Distributed Systems, Software Architectures, Multi-agent systems, Machine Learning.

Карен КОРТЕС-ВЕРДИН, кандидат компьютерных наук, профессор факультета статистики и информатики. Область научных интересов: программная инженерия, качество программного обеспечения, семейства программных продуктов.

Karen CORTES-VERDIN, Doctor in Computing Sciences, Professor at the School of Statistics and Informatics. Research interests: software engineering, software quality, software product lines.

Хорхе Октавио ОЧАРАН-ЭРНАНДЕС, кандидат компьютерных наук, профессор факультета статистики и информатики. Область научных интересов: разработка программного обеспечения, архитектура программного обеспечения, разработка требований, разработка API.

Jorge Octavio OCHARÁN-HERNÁNDEZ, Doctor in Computing Sciences, Professor at the School of Statistics and Informatics. Research interests: software engineering, software architecture, requirements engineering, API design.

DOI: 10.15514/ISPRAS–2021–33(1)–7

Последние тенденции в развитии подводной беспроводной сенсорной сети: систематический обзор литературы

[1] *А. Тарик, ORCID: 0000-0002-9994-3900 <atariq17@ce.ceme.nust.edu.pk>*
[1] *Ф. Азам, ORCID: 0000-0002-7421-7400 <farooq@ceme.nust.edu.pk>*
[1] *М.В. Анвар, ORCID: 0000-0002-1193-5683 <waseemanwar@ceme.nust.edu.pk>*
[1] *Т. Захур, ORCID: 0000-0003-1923-0087 <tzahoor17@ ce.ceme.nust.edu.pk>*
[2] *А.В. Музаффар, ORCID: 0000-0001-7910-0378 <a.muzaffar@seu.edu.sa>*

[1] *Национальный научно-технологический университет (NUST), Пакистан, 44000, г. Исламабад, NUST HQ, H-12*
[2] *Саудовский Университет электроники, 13323, Саудовская Аравия, Эр-Рияд, Ан Нарджис*

Аннотация. Подводная беспроводная сенсорная сеть (Underwater Wireless Sensor Network, UWSN) – это новая технология мониторинга водных объектов, которая часто используется для сбора информации под водой, отбора проб в океане, выявления неопознанных движущихся объектов, предупреждения стихийных бедствий и обнаружения подводных лодок. В последнее время сети UWSN привлекают большое внимание исследователей как из академических, так и из промышленных кругов. В результате было проведено несколько исследований с целью усовершенствования методов, инструментов, протоколов и архитектуры UWSN. В связи с этим существует острая необходимость в изучении и обобщения современных тенденций развития UWSN в рамках одного исследования. Для достижения этой цели в данной статье проводится систематический обзор литературы, в котором комплексно анализируются последние разработки в области UWSN. В частности, было отобрано и проанализировано 34 научных исследования, опубликованных в период 2012-2020 гг. в области UWSN. В результате чего были определены 21 актуальный протокол маршрутизации и 11 инструментов. Кроме того, в контексте UWSN представлены пять различных типов архитектуры и три технологии средств коммуникации. Наконец, был проведен сравнительный анализ протоколов маршрутизации на основе значимых оценочных показателей. По итогам анализа был сделан вывод о том, что существуют отвечающие требованиям подходы, протоколы и инструменты для мониторинга посредством UWSN. Однако возможности верификации текущих подходов недостаточны для удовлетворения растущих требований к UWSN. Выводы, сделанные в этой статье, закладывают прочную основу для дальнейшего усовершенствования нынешних инструментов и методов, используемых в UWSN, для их реализации в крупных и сложных сетях.

Ключевые слова: подводные беспроводные сенсорные сети; систематический обзор литературы; протоколы; инструменты; сетевые симуляторы; архитектура

Для цитирования: Тарик А., Азам Ф., Анвар М.В., Захур Т., Музаффар А.В. Последние тенденции в развитии подводной беспроводной сенсорной сети: систематический обзор литературы. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 97-110. DOI: 10.15514/ISPRAS–2021–33(1)–7

Recent Trends in Underwater Wireless Sensor Networks (UWSNs) – A Systematic Literature Review

[1] *A. Tariq, ORCID: 0000-0002-9994-3900 <atariq17@ce.ceme.nust.edu.pk>*
[1] *F. Azam, ORCID: 0000-0002-1193-5683 <farooq@ceme.nust.edu.pk >*
[1] *M.W. Anwar, ORCID: 0000-0002-1193-5683 <waseemanwar@ceme.nust.edu.pk >*
[1] *T. Zahoor, ORCID: 0000-0003-1923-0087 <tzahoor17@ ce.ceme.nust.edu.pk>*
[2] *A.W. Muzaffar, ORCID: 0000-0001-7910-0378 <a.muzaffar@seu.edu.sa>*

[1] *National University of Sciences and Technology (NUST),*
44000 NUST HQ, H-12, Islamabad, Islamabad Capital Territory, Pakistan
[2] *Saudi Electronic University,*
An Narjis, Riyadh, 13323 Saudi Arabia

Abstract. Underwater Wireless Sensor Networks (UWSNs) is an emerging technology for the monitoring of aquatic assets and frequently applied in several domains like underwater information gathering, ocean sampling network, anonymous vehicles, disaster prevention and submarine detection. Recently, UWSNs have been getting significant attention of researchers from both academia and industry. As a result, several studies have been carried out to perform certain improvements in UWSNs techniques, tools, protocols and architectures. In this regard, there is a dire need to investigate and summarize the modern UWSNs trends altogether within a single study. To achieve this, a Systematic Literature Review (SLR) is performed in this article to comprehensively analyze the latest developments in UWSNs. Particularly, 34 research studies published during 2012-2020 have been selected and examined in the area of UWSNs. This leads to the identification of 21 modern routing protocols and 11 tools. Furthermore, 5 different architecture types and 3 communication media technologies are presented in the context of UWSNs. Finally, a comparative analysis of routing protocols is done on the basis of important evaluation metrics. It has been concluded that there exist adequate approaches, protocols and tools for the monitoring of UWSNs. However, the design verification capabilities of existing approaches are insufficient to meet the growing demands of UWSNs. The findings of this article provide solid platform to enhance the current UWSNs tools and techniques for large and complex networks.

Keywords: Underwater Wireless Sensor Networks; Systematic Literature Review; protocols; tools; network simulators; architectures; SLR; UWSNs

For citation: Tariq A., Azam F., Anwar M.W., Zahoor T., Muzaffar A.W. Recent Trends in Underwater Wireless Sensor Networks (UWSNs) – A Systematic Literature Review. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 97-110 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–7.

1. Введение

Подводная беспроводная сенсорная сеть (UWSN) — это сочетание беспроводной технологии и небольшого микромеханического сенсорного оборудования, обладающего функциями интеллектуального восприятия, вычисления и умных коммуникаций. Сети UWSN находят широкое применение при исследованиях под водой в научных, экологических и военных целях [1]. В последнее время было проведено несколько исследований с целью усовершенствования методов, инструментов, протоколов и архитектур UWSN. Имеется также несколько исследований [47], в которых рассматривается развитие беспроводных сенсорных сетей в целом, а не исключительно подводных. Кроме того, в последнее время проводятся исследования [35-38], в ходе которых изучаются различные аспекты UWSN, такие как проблемы маршрутизации, протокольные операции, технологии оценки и обеспечения. Однако, насколько нам известно, не существует исследования, в котором были бы проанализированы и изложены все последние достижения в области UWSN. Учитывая важность вопроса, мы провели систематический обзор литературы (Systematic Literature Review, SLR) за период с 2012-го по 2020-й гг., в рамках которого было рассмотрено 34 исследования, направленных на поиск решений следующих исследовательских вопросов (Research Question, RQ):

RQ1: Какие значительные исследования проводились в области UWSN в 2012-2020 гг.?

RQ2: Каковы основные протоколы маршрутизации, типы архитектуры и технологии связи, которые использовались для UWSN в ходе исследований 2012-2020 гг.?

RQ3: Какие инструменты/симуляторы регулярно использовались для UWSN в течение 2012-2020 гг.?

RQ4: Каковы перспективы развития в области UWSN?

В соответствии с критериями отбора материала (разд. 2) для выполнения данного систематического обзора литературы были использованы следующие четыре популярные научные базы данных: IEEE, Springer, ACM и Elsevier. Таким образом, было отобрано 34 исследования [1-34], связанные с UWSN. Результаты исследования приведены в разд. 3. И, наконец, в разд. 4 представлен анализ результатов исследования.

2. Методология исследования

Данное исследование проводится в соответствии с методическими указаниями по проведению систематического обзора литературы, сформулированными Барбарой Китченхэм (Barbara Kitchenham) [45], в которых неотъемлемым элементом является протокол обзора.

2.1 Критерии отбора материала

Для отбора исследований определены следующие четыре параметра, которые приводят к качественным результатам:

1) выбранные исследования должны быть связаны с UWSN;
2) выбранные исследования должны быть опубликованы в 2012-2020 гг.;
3) исследование выбирается только в том случае, если публикация проиндексирована в одной из следующих известных научных баз данных: IEEE, SPRINGER, ACM и ELSEVIER;
4) выбранные исследования должны быть ориентированы на результат и обеспечивать реальное решение ряда проблем UWSN.

В свою очередь, исследования, которые не соответствуют всем указанным выше параметрам отбора, не могут быть использованы.

2.2 Процесс поиска

Были выбраны четыре научные базы данных (т.е. IEEE, Springer, ACM и Elsevier) с целью найти достоверные и положительные результаты исследований из журналов с высоким импакт-фактором и материалов конференций. В ходе поиска материалов были использованы различные поисковые запросы, такие как «подводные беспроводные сенсорные сети», «протоколы UWSN», «подводные беспроводные акустические сети», «архитектура UWSN» и «разработка, управляемая моделью UWSN», приведенные в табл. 1.

Табл. 1. Поисковые запросы с результатами
Table 1. Search Terms with Results

№	Поисковый запрос	Результаты поиска (2012-2020)			
		IEEE	Springer	ACM	Elsevier
1	Подводные беспроводные сенсорные сети	1291	1202	955	1089
2	Протоколы UWSN	548	687	578	247
3	Подводные беспроводные акустические сети	1179	614	502	457

4	Архитектура UWSN	487	599	688	333
5	Разработка, управляемая моделью UWSN	246	362	187	638

Для получения релевантных результатов были применены различные фильтры, такие как логический оператор AND, фильтр по году (2012-2020) и др. В ходе анализа мы обнаружили, что лишь несколько исследований не соответствуют критериям отбора в полной мере. Например, результаты не были должным образом представлены в исследовании [48]. Кроме того, авторы исследования [49] не проводили анализ эффективности. Таким образом, по итогам детального анализа были исключены подобные исследования и, наконец, отобраны 34 исследования, которые полностью удовлетворяют критериям отбора (подраздел 2.1).

2.3 Оценка качества

Для обеспечения достоверности результатов данного систематического обзора литературы, мы старались максимально отбирать высококачественные исследования. Кроме того, было отобрано как можно больше новых исследований. На рис. 1 представлено распределение отобранных исследований по годам.

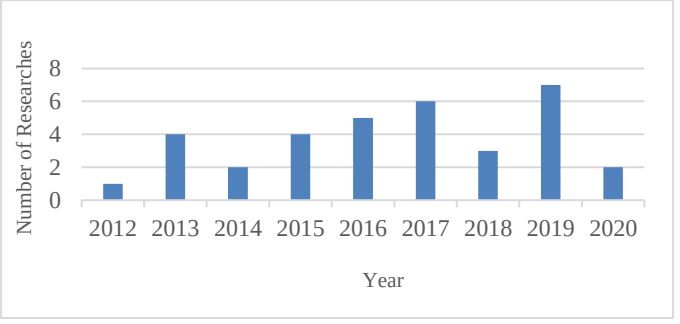


Рис. 1. Распределение отобранных исследований по годам
Fig 1. Yearly Distribution of Selected Researches

2.4 Извлечение данных

Для получения соответствующих данных были определены различные параметры их извлечения, как показано в табл. 2.

Табл. 2. Шаблон извлечения данных
Table.2. Template of Data Extraction

№	Описание	Подробная информация
1	Библиографическая информация	Авторы исследования, название, год издания и вид исследования (т.е. журнал/конференция).
Извлечение данных		
2	План	Основная цель выбранного исследования, например, назначение предлагаемого подхода и соответствующего инструмента/протокола в контексте подводных беспроводных сенсорных сетей.
3	Результаты	Результаты выбранных исследований
4	Предположения	Используется количественный или качественный метод
5	Проверка	Оценка результатов, например, тематическое исследование и т.д.

Синтез данных		
6	Категоризация	Соотнесение выбранного исследования с заранее определенной категоризацией (табл. 3)
7	Инструменты	Инструменты, используемые в подводных беспроводных сенсорных сетях (табл. 4)
8	Типы архитектуры	Типы архитектуры, используемые в подводных беспроводных сенсорных сетях (табл. 5)
9	Протоколы и методы оптимизации	Протоколы с соответствующей техникой оптимизации, используемые в каждом выбранном исследовании (табл. 6).
10	Коммуникационные технологии	Коммуникационные технологии, используемые в подводных беспроводных сенсорных сетях (табл. 7)
11	Оценка эффективности каждого из предложенных протоколов	Каждый протокол оценивается по матрице эффективности (табл. 8)

3. Результаты

В данном разделе изложены результаты, полученные по итогам проведения выбранных исследований. В частности, категоризация отдельных исследований приводится в подразделе 3.1. Кроме того, указанные инструменты, архитектуры, протоколы и технологии связи представлены в подразделах 3.2, 3.3, 3.4 и 3.5 соответственно.

3.1 Категоризация выбранных исследований

Мы разделили 34 отобранных исследования на пять групп по их соответствию целевым показаниям качества, как указано в табл. 3.

Табл. 3. Категории исследований в соответствии с их целями
Table.3. Categories of Researches on the Basis of Targeted Objective

№	**Категория**	**Источник**
1	Энергопотребление	[2][4][7][8][9][10][14][16][17][21][23] [25][28][33][34]
2	Локализация	[1][3][6][13][15][18][26][27][31]
3	Время выполнения	[11]
4	Использование системы	[19][20][30]
5	Пропускная способность	[5][12][22][24][29][32]

Описание этих категорий выглядит следующим образом.

I. Категория *Энергопотребление* включает в себя исследования, в которых предлагаются меры для снижения энергопотребления в соответствии со структурой UWSN. Энергопотребление является высоко значимой характеристикой [39] беспроводных сетей.

II. Категория *Размещение* включает в себя исследования, направленные на повышение точности размещения узлов в UWSN.

III. *Время выполнения* является важным показателем качества в беспроводных сетях [43] и других областях [42]. Таким образом, категория *Время выполнения* включает в себя те

отобранные исследования, в которых время передачи пакетов данных является целевым показателем, когда количество узлов превышено.

IV. Категория *Использование системы* включает в себя отдельные исследования, которые направлены на оптимизацию использования сетей путем применения различных стратегий размещения в UWSN.

V. В категории *Пропускная способность* рассматриваются исследования, целью которых является повышение коэффициента доставки пакетов в UWSN.

3.2 Инструменты/симуляторы

В случае с UWSN тестирование любой методики/алгоритма является дорогостоящим и трудным с точки зрения реализации в глубоководных частях океана. Таким образом, прежде чем перейти к фактическому развертыванию UWSN, важно провести тестирование. В 34 отобранных исследованиях мы выявили 11 инструментов/симуляторов, использованных для проведения анализа до фактического развертывания, как указано в табл. 4. Как видно из данных табл. 4, наиболее часто используемым инструментом в UWSN является сетевой симулятор (Network Simulator, NS). В частности, NS2 – это старая версия сетевого симулятора, которая теперь заменена на новую версию NS3. Лишь в немногих отобранных исследованиях сообщалось об использовании NS2. Более распространенным все же было использование NS3. Таким образом, в табл. 4 мы объединяем все исследования в рамках сетевого симулятора (как NS2, так и NS3 версии).

Симуляторы играют важную роль в сложной среде UWSN, так как они позволяют проверить проект сети перед ее фактическим развертыванием. К тому же, модельно-управляемый подход обеспечивает возможность повторного использования и облегчает проверку проекта сети [44]. Для достижения желаемой конструктивной простоты UWSN можно системно использовать концепции модельно-управляемого подхода. Например, NS3 – это симулятор, который зависит от платформы, в то время как при модельно-управляемом подходе используются модели, независящие от платформы. В связи с этим независимая от платформы модель может быть автоматически преобразована в модель, зависящую от платформы [46], как например, NS3 для моделирования среды UWSN. Комбинация модельно-управляемого подхода с использованием инструментов UWSN, безусловно, может упростить процесс моделирования и верификации сети. Однако подобная комбинация все еще нуждается в дальнейшем изучении.

Табл. 4. Инструменты/симуляторы UWSN
Table 4. Identification of UWSNs Tools / Simulators

№	Инструменты/симуляторы	Доступность	Источник
1	Сетевой симулятор (версии NS2 и NS3)	С открытым исходным кодом	[1] [2] [5][7] [9][11] [12][18][19][21][22][25][26] [27][28][29] [31][32][33][34]
2	AquaSim	С открытым исходным кодом	[3][5][7][18][22][25][26][34]
3	Matlab	Коммерческий	[3][10][17]
4	QualNet	Коммерческий	[6][8][11]
5	J-Sim	С открытым исходным кодом	[11]

6	SUNSET SDCS	С открытым исходным кодом	[23]
7	GME (Generic Modeling Environment)	С открытым исходным кодом	[30]
8	Eclipse Modeling Framework (EMF)	С открытым исходным кодом	[31]
9	DigitizeIt	С открытым исходным кодом	[11]
10	OPNET	Коммерческий	[16][11]
11	GloMoSim	Коммерческий	[11]

3.3 Архитектура UWSN

Из 34 отобранных исследований мы выделили четыре типа коммуникационной архитектуры UWSN: одномерную, двухмерную, трехмерную и четырехмерную. Классификация отобранных исследований по типу архитектуры приведена в табл. 5. Существуют исследования, которые вводят новые типы архитектуры, однако подробная информация по ним не приводится. Подобные типы архитектуры классифицированы как «Другие» (см. табл. 5).

Табл. 5. Коммуникационная архитектура UWSN
Table 5. UWSN Communication Architectures

№	Типы архитектуры	Источник
1	Одномерная (1D)	[3][8]
2	Двухмерная (2D)	[1][13] [25][28]
3	Трехмерная (3D)	[2][4][6][9][16][17][20][21][24][26][27][29][33]
4	Четырехмерная (4D)	[1][6][7]
5	Другие	[5] [8]

3.4 Протоколы маршрутизации

Мы выявили 21 важный протокол маршрутизации и соответствующую технику оптимизации, указанные в табл. 6.

Табл. 6. Протоколы маршрутизации и соответствующая оптимизация
Table 6. Routing protocols and corresponding optimization

Источник	Протокол	Технология оптимизации
[1]	Hop-by-Hop Dynamic Addressing Based (H2-DAB)	Алгоритм последовательной динамической адресации
[2]	Adaptive hop-by-hop Vector-Based Forwarding (AHH-VBF)	Технология адаптирующейся переадресации/передачи пакета
[5]	Geographic Routing Protocol	Жадная стратегия произвольной географической переадресации
[6]	Маршрутизация с учетом давления пустот (VAPR)	Жадный подход к гибкой переадресации

[7]	Geographic and Opportunistic Routing	Жадная гибкая стратегия переадресации с корректировкой глубины
[8]	HydroCast	Восстановление пустот с жадной гибкой переадресацией
[9]	Agent Based Approach (ABA)	Установление маршрута на основе энергии узла и количества переходов
[19]	Dynamic Reservation Access Protocols	Подход к динамическому резервированию множественного доступа с учетом состояния системы
[20]	Enhanced CARP (E-CARP)	Жадная последовательная стратегия маршрутизации, независящая от местоположения
[21]	Energy-Aware and Void-Avoidable Routing Protocol (EAVARP)	Стратегия гибкой переадресации, основанная на избежании пустотных участков
[22]	Deep Q-Network-Based Energy- and Latency-Aware (DQELR)	Алгоритм глубокой Q-сети с методами off-policy и on-policy
[23]	Channel-Aware Reinforcement Learning-Based Multi-Path (CARMA)	Фреймворк распределенного обучения с подкреплением
[24]	Proactive Routing Approach with Energy efficient Path	Алгоритм Дейкстры с вертикальным наслоением и формированием кластеров
[25]	Energy-Efficient Multipath Grid-based Geographic Routing (EMGGR)	Алгоритм выбора шлюза с механизмами переадресации пакетов
[26]	Energy-Efficient Localization-Based Geographic Routing (EEL)	Локализация NADV (нормализованное продвижение) и TOA (время прибытия)
[27]	Improved VBF (Vector Routing Forwarding)	Географическая стратегия маршрутизации с радиусом маршрутной трубки
[28]	Multi-Layered Routing Protocol (MRP)	Механизм маршрутизации, не зависящий от локализации
[29]	Grid Division Polar Tracing (GDPT)	Жадный алгоритм с использованием модели кубической сетки и полярным отслеживанием
[32]	On-Surface Wireless-Assisted Opportunistic (SurOpp)	Гибкая маршрутизация в буйковых узлах
[33]	Energy Efficient Data Gathering (EEDG)	Подход с использованием информации о глубине и переадресующих узлов
[34]	Dynamic Firefly Mating Optimization Inspired Routing Protocol (FFRP)	Самообучающийся динамический интеллектуальный алгоритм оптимизации Firefly Mating

В настоящее время существует тенденция к глубокого машинного обучения в области управления проблемами маршрутизации [40]. В связи с этим авторы [22] объединили нейронную сеть с Q-обучением для принятия глобально оптимального решения о

маршрутизации. В другом исследовании используется самообучающаяся динамическая интеллектуальная технология оптимизации *firefly mating* [34] для определения высокостабильных и безопасных путей маршрутизации пакетов в UWSN через коммуникационные пустоты и теневые зоны.

3.5 Коммуникационные технологии UWSN

В табл. 7 приведены краткие характеристики трех коммуникационных технологий (акустической, оптической электромагнитной), которые часто применяются в UWSN [41]. Каждая коммуникационная технология оценивается по различным атрибутам, в частности, по дальности связи и скорости передачи данных. Аналогично «задержка» означает задержку в передаче данных UWSN, а «мощность передачи» – потребление энергии во время передачи данных. «Затраты» подразумевает под собой затраты на развертывание и операционную деятельность каждой технологии. «Направленность» может означать всенаправленную приемопередачу, при которой сигнал передается и принимается с любого из возможных направлений, или однонаправленную приемопередачу, при которой сигнал передается или принимается только в одном направлении. Наконец, «потеря на пути распространения сигнала (затухание)» и «энергопотребление» [8] [12] [15] представляют собой соответствующие понятия каждой коммуникационной технологии.

Наш анализ показывает, что в настоящее время акустическая технология является основной технологией, используемой в UWSN. В частности, в девяти отобранных исследованиях ([5][6][8][12][13][14][15][17][18]) используется только акустическая технология связи. В одном исследовании [10] используется комбинация акустической и оптической технологий. В девяти исследованиях ([1][2][3][7][9][16][26][32][33]) используется комбинация акустических и радиочастотных технологий. Наконец, только в трех исследованиях ([4][11][34]) используется исключительно радиочастотная технология.

Табл.7. Сравнение различных технологий подводной связи
Table.7. Comparison of different Underwater Communication Technologies

Технологии	**Дальность связи**	**Скорость передачи данных**	**Задер-жка**	**Энергопотребление**	**Мощность передачи**	**Затраты**	**Затухание**	**Направление**
Акустическая	<20 км	<10 кбит/с	Высокая	100 бит/Дж	≈ 10 Вт	Высокие	Высокое	Все
Оптическая	100—200 м	<10 Гбит/с	Низкая	30 000 бит/Дж	≈ 1 Вт	Высокие	Зависит от прозрачности	Одно
Электромагнитная (радиочастотная)	< 100 м	<0,1 Гбит/с	Умеренная	Н/Д	≈ 100 Вт	Низкие	Умеренное	Все

4. Анализ

В этом разделе представлен анализ эффективности и оценка выявленных протоколов маршрутизации в UWSN. Рассматриваются шесть параметров оценки: энергоэффективность, коэффициент доставки, сквозная задержка, экономическая эффективность, мобильность узлов и интеграция с подводными протоколами MAC. Это позволяет провести полное сравнение протоколов маршрутизации, результаты которого приведены в табл. 8.

Табл. 8. Качественный анализ предлагаемых протоколов маршрутизации с помощью показателей эффективности
Table 8. Qualitative Analysis of Proposed Routing Protocols through Performance Metrics

Источник	**Протокол**	**Энергоэффективность**	**Коэффициент доставки**	**Сквоз-ная задер-жка**	**Экономичес-кая эффектив-ность**	**Мобильность узлов**	**Интеграция с протоколами MAC**
[1]	H2-DAB	Средняя	Средний	Низкая	Низкая	Высокая	IEEE 802.11
[2]	AHH-VBF	Высокая	Средний	Низкая	Низкая	Нет данных	Aloha
[5]	Geographic routing protocol	Средняя	Высокий	Средняя	Средняя	Нет данных	CSMA
[6]	VAPR	Нет данных	Высокий	Низкая	Нет данных	Высокая	CSMA
[7]	GEDAR	Высокая	Высокий	Средняя	Высокая	Средняя	CSMA
[8]	HydroCast	Высокая	Высокий	Низкая	Средняя	Средняя	CSMA
[9]	ABA	Высокая	Средний	Средняя	Высокая	Нет данных	
[19]	Dynamic reservation access protocols	Высокая	Нет данных	Низкая	Средняя	Средняя	Aloha
[20]	E-CARP	Высокая	Средний	Нет данных	Средняя	Высокая	Нет данных
[21]	EAVARP	Средняя	Средний	Низкая	Нет данных	Высокая	Нет данных
[22]	DQELR	Высокая	Средний	Средняя	Нет данных	Высокая	Нет данных
[23]	CARMA	Высокая	Высокий	Низкая	Нет данных	Нет данных	Нет данных
[24]	PA-EPS-Case I	Средняя	Высокий	Средняя	Низкая	Средняя	Нет данных
[25]	EMGGR	Высокая	Высокий	Низкая	Нет данных	Средняя	Нет данных
[26]	EEL	Высокая	Высокий	Средняя	Средняя	Низкая	Нет данных
[27]	Improved VBF Algorithm	Высокая	Высокий	Нет данных	Нет данных	Высокая	Нет данных
[28]	MRP	Высокая	Высокий	Средняя	Нет данных	Низкая	802.11-DYNAV
[29]	GDPT	Высокая	Высокий	Низкая	Средняя	Средняя	Нет данных
[32]	SurOpp	Высокая	Высокий	Низкая	Высокая	Средняя	CSMA, IEEE 802.11n
[33]	EEDG	Высокая	Средний	Низкая	Нет данных	Нет данных	Нет данных
[34]	FFRP	Высокая	Высокий	Низкая	Нет данных	Нет данных	CSMA

Теперь мы можем ответить на все RQ, указанные в разделе 1. В частности, ответ на RQ1 приведен в табл. 3. Ответ на RQ2 можно найти в табл. 5, 6 и 7. Ответ на RQ3 содержится в табл. 4. Наконец, отвечая на RQ4, можно сказать, что благодаря технологическому и экономическому прогрессу, UWSN вызывает особый интерес у исследователей и представителей промышленностей. За последние несколько десятилетий в этой области было разработано множество систем предупреждения, защиты и мониторинга в режиме реального

времени. Однако из-за неизбежных ограничений UWSN реализация полноценных решений и приложений в данной области является затруднительным. К подобным сдерживающим факторам относятся ограниченные аппаратные ресурсы (вычислительная мощность, хранилище), ненадежность коммуникационных технологий, длительное и нерегулярное затухание, ограниченный срок службы сети, фиксированная пропускная способность, шум, физическая восприимчивость, сетевые атаки и ошибки при передаче данных. Для устранения данных ограничений необходимо разработать более продвинутые инструменты и протоколы для управления будущими сложными UWSN.

Несмотря на то, что мы следовали стандартным рекомендациям систематического обзора литературы [45], существует небольшая вероятность того, что мы могли упустить некоторые исследования, опубликованные в других базах данных, таких как Taylor & Francis и др. Но были рассмотрены четыре авторитетные научные базы данных, такие как IEEE, Springer, ACM и Elsevier. Таким образом, результаты данного систематического обзора литературы надежные, и отсутствие нескольких исследований из других источников не оказывает существенного влияния на общие результаты.

5. Заключение

В данной статье представлен систематический обзор литературы (SLR) для изучения текущих приложений и разработок в области UWSN. Отобраны и тщательно проанализированы 34 исследования, опубликованные в течение 2012-2020 гг. В результате было выявлено 11 инструментов, 21 протокол маршрутизации с соответствующими методами оптимизации, пять типов внутренней архитектуры и три коммуникационные технологии.

Впоследствии был проведен сравнительный анализ протоколов маршрутизации. По его итогам можно сделать вывод, что NS2/NS3 с AquaSim являются самыми надежными симуляторами/инструментами для UWSN. Кроме того, в последнее десятилетие интенсивно используются трехмерные (3D) коммуникационные архитектуры. Акустическая технология является основной коммуникационной технологией, используемой в настоящее время в UWSN.

Наконец, в ходе исследования было обнаружено, что такие показатели производительности, как энергоэффективность, коэффициент доставки, сквозная задержка, экономическая эффективность, мобильность узлов и интеграция с подводным протоколом MAC, имеют достаточно важное значение при выборе правильных протоколов маршрутизации в соответствии с требованиями.

Детальная и надежная проверка UWSN перед фактическим ее развертыванием крайне необходима. В связи с этим требуется более детальный анализ инструментов, чтобы выявить конкретные преимущества и ограничения. Такой подробный анализ инструментов UWSN мы планируем провести в следующей статье.

Список литературы / References

[1]. Muhammad Ayaz, Azween Abdullah, Ibrahima Faye, Yasir Batira. An efficient Dynamic Addressing based routing protocol for Underwater Wireless Sensor Networks. Computer Communications, vol. 35, issues 4, 2012, pp. 475-486.

[2]. Haitao Yu, Nianmin Yao, Jun Liu. An adaptive routing protocol in underwater sparse acoustic sensor networks. Ad Hoc Networks, vol. 34, 2015, pp. 121-143.

[3]. Anjana P Das, Sabu M Thampi. Fault-resilient localization for underwater sensor networks. Ad Hoc Networks, vol. 55, 2017, pp. 132-142.

[4]. Jun Liu, Zhong Zhou, Zheng Peng et al. Mobi-Sync: Efficient Time Synchronization for Mobile Underwater Sensor Networks. IEEE Transactions on Parallel and Distributed Systems, vol. 24, issue 2, 2013, pp. 406-416.

[5]. Rodolfo W.L. Coutinho, Azzedine Boukerche, Luiz F.M. Vieira, Antonio A.F. Loureiro. A novel void node recovery paradigm for long-term underwater sensor networks. Ad Hoc Networks, vol. 34, 2015, pp. 144-156.

[6]. Youngtae Noh, Uichin Lee, Paul Wang et al. VAPR: Void-Aware Pressure Routing for Underwater Sensor Networks. IEEE Transactions on Mobile Computing, vol. 2, issue 5, 2013, pp. 895-908.

[7]. Rodolfo W. L. Coutinho, Azzedine Boukerche, Luiz F. M. Vieira, Antonio A. F. Loureiro. GEDAR: Geographic and Opportunistic Routing Protocol with Depth Adjustment for Mobile Underwater Sensor Networks. In Proc. of the IEEE International Conference on Communications (ICC), 2014, pp. 251-256,

[8]. Youngtae Noh, Uichin Lee, Saewoom Lee et al. HydroCast: Pressure Routing for Underwater Sensor Networks. IEEE Transactions on Vehicular Technology, vol. 65, issue 1, 2016, pp. 333-347,

[9]. Manjula R. Bharamagoudra, SunilKumar S. Manvi, Bilal Gonen. Event driven energy depth and channel aware routing for underwater acoustic sensor networks: Agent oriented clustering based approach. Computers & Electrical Engineering, vol. 58, 2017, pp. 1-19.

[10]. Hainan Chen, Xiaoling Wu, Guangcong Liu, Yanwen Wang. A Novel Multi-Module Separated Linear UWSNs Sensor Node. IEEE Sensors Journal, volume 16, issue 11, 2016, pp. 4119-4126.

[11]. Mukhtar Ghaleb, Emad Felemban, Shamala Subramaniam et al. A Performance Simulation Tool for the Analysis of Data Gathering in Both Terrestrial and Underwater Sensor Networks. IEEE Access, vol. 5, 2017, pp. 4190-4208.

[12]. Zhenghao Xi, Xiu Kan, Le Cao et al. Research on Underwater Wireless Sensor Network and MAC Protocol and Location Algorithm. IEEE Access, vol. 7, 2019, pp. 56606-56616.

[13]. Inam Ullah, Yiming Liu, Xin Su, Pankoo Kim. Efficient and Accurate Target Localization in Underwater Environment. IEEE Access, vol. 7, 2019, pp. 101415-101426.

[14]. Bingbing Zhang, Yiyin Wang, Hongyi Wang et al. Tracking a Duty-Cycled Autonomous Underwater Vehicle by Underwater Wireless Sensor Networks. IEEE Access, vol. 5, 2017, pp. 18016-18032.

[15]. Inam Ullah, Jingyi Chen, Xin Su et al. Localization and Detection of Targets in Underwater Wireless Sensor Using Distance and Angle Based Algorithms. IEEE Access, vol. 7, 2019, pp. 45693-45704.

[16]. Do Duy Tan, Tung Thanh Le, Dong-Seong Kim. Distributed Cooperative Transmission for Underwater Acoustic Sensor Networks. In Proc. of the IEEE Wireless Communications and Networking Conference Workshops (WCNCW), 2013, pp. 205-210.

[17]. S. Mohamad Dehnavi, Moosa Ayati, Mohammad Reza Zakerzadeh. Three Dimensional Target Tracking via Underwater Acoustic Wireless Sensor Network. In Proc. of the Conference on Artificial Intelligence and Robotics (IRANOPEN), 2017, pp. 153-157.

[18]. Jun Liu, Zhaohui Wang, Jun-Hong Cui et al. A Joint Time Synchronization and Localization Design for Mobile Underwater Sensor Networks. IEEE Transactions on Mobile Computing, vol. 15, issue 3, 2016, pp. 530-543.

[19]. Priyatosh Mandal, Swades De. New Reservation Multiaccess Protocols for Underwater Wireless Ad Hoc Sensor Networks. IEEE Journal of Oceanic Engineering, vol. 40, issue 2, 2015, pp. 277-291.

[20]. Zhangbing Zhou, Beibei Yao, Riliang Xing et al. E-CARP: An Energy Efficient Routing Protocol for UWSNs in the Internet of Underwater Things. IEEE Sensors Journal, vol. 16, issue 11, 2016, pp. 4072-4082.

[21]. Zhuo Wang, Guangjie Han, Hongde Qin et al. An Energy-Aware and Void-Avoidable Routing Protocol for Underwater Sensor Networks. IEEE Access, vol. 6, 2018, pp. 7792-7801.

[22]. Yishan Su, Rong Fan, Xiaomei Fu, Zhigang Jin. DQELR: An Adaptive Deep Q-Network-Based Energy- and Latency-Aware Routing Protocol Design for Underwater Acoustic Sensor Networks. IEEE Access, vol. 7, 2019, pp. 9091-9104.

[23]. Valerio Di Valerio, Francesco Lo Presti, Chiara Petrioli et al. CARMA: Channel-Aware Reinforcement Learning-Based Multi-Path Adaptive Routing for Underwater Wireless Sensor Networks. IEEE Journal on Selected Areas in Communications, vol. 37, issue 11, 2019, pp. 2634-2647.

[24]. Zahoor Ali Khan, Muhammad Awais, Turki Ali Alghamdi et al. Region Aware Proactive Routing Approaches Exploiting Energy Efficient Paths for Void Hole Avoidance in Underwater WSNs, IEEE Access, vol. 7, 2019, pp. 140703-140722.

[25]. Faiza Al Salti, N. Alzeidi, Bassel R. Arafeh. EMGGR: an energy-efficient multipath grid-based geographic routing protocol for underwater wireless sensor networks. Wireless Networks, vol. 23, issue 4, 2017, pp. 1301-1314.

[26]. Kun Hao, Haifeng Shen, Yonglei Liu et al. Integrating Localization and Energy-Awareness: A Novel Geographic Routing Protocol for Underwater Wireless Sensor Networks. Wireless Networks, vol. 23, issue 5, 2018, 1427-1435.

[27]. Sayyed Majid Mazinani, Hadi Yousefi, Mostafa Mirzaie. A Vector-Based Routing Protocol in Underwater Wireless Sensor Networks. Wireless Personal Communications, vol. 100, 2018, pp. 1569-1583.

[28]. Abdul Wahid, Sungwon Lee, Dongkyun Kim, Kyung-Shik Lim. MRP: A Localization-Free Multi-Layered Routing Protocol for Underwater Wireless Sensor Networks. Wireless Personal Communications, vol. 77, 2014, pp. 2997-3012.

[29]. BalajiVijayan Venkateswarulu, Neduncheliyan Subbu, Sivakumar Ramamurthy. An efficient routing protocol based on polar tracing function for underwater wireless sensor networks for mobility health monitoring system application. Journal of Medical Systems, vol. 43, 2019, article no. 18.

[30]. C Srimathi, Soo-Hyun Park, N Rajesh. Proposed framework for underwater sensor cloud for environmental monitoring. In Proc. of the Fifth Conference on Ubiquitous and Future Networks (ICUFN), 2013, pp. 104-109.

[31]. Charbel Geryes Aoun, Iyas Alloush, Yvon Kermarrec et al. A Mapping Approach for Marine Observatory Relying on Enterprise Architecture. In Proc. of the OCEANS - MTS/IEEE Washington, 2015, pp. 1-10.

[32]. Miaomiao Liu, Fei Ji, Quansheng Guan et al. On-Surface Wireless-Assisted Opportunistic Routing for Underwater Sensor Networks. In Proc. of the 11th ACM International Conference on Underwater Networks & Systems, 2016, pp. 1-5.

[33]. Fatemeh Banaeizadeh, Abolfazl Toroghi Haghighat. An energy-efficient data gathering scheme in underwater wireless sensor networks using a mobile sink. International Journal of Information Technology, vol. 12, 2020, pp. 513-522.

[34]. Muhammad Faheem, Rizwan Aslam Butt, Basit Raza et al. FFRP: Dynamic Firefly Mating Optimization Inspired Energy Efficient Routing Protocol for Internet of Underwater Wireless Sensor Networks. IEEE Access, vol. 8, 2020, pp. 39587-39604.

[35]. Mukhtiar Ahmed, Mazleena Salleh, M. Ibrahim Channa. Routing protocols based on node mobility for Underwater Wireless Sensor Network (UWSN): A survey. Journal of Network and Computer Applications, vol. 78, 2017, pp. 242-252.

[36]. Mukhtiar Ahmed, Mazleena Salleh, M.Ibrahim Channa. Routing protocols based on protocol operations for underwater wireless sensor network: A survey. Journal of Egyptian Informatics Journal, vol. 19, issue 1, 2018, pp. 57-62.

[37]. Mohammed Jouhari, Khalil Ibrahimi, Hamidou Tembine, Jalel Ben-Othman. Underwater Wireless Sensor Networks: A Survey on Enabling Technologies, Localization Protocols, and Internet of Underwater Things. IEEE Access, vol. 7, 2019, pp. 96879-96899.

[38]. Rodolfo W.L. Coutinho, Azzedine Boukerche. Data Collection in Underwater Wireless Sensor Networks: Research Challenges and Potential Approaches. In Proc. of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems, 2017, pp. 5-8.

[39]. Thinakaran Vasantha Chithra, Arulappan Milton. Energy Proficient Flooding Scheme Using Reduced Coverage Set Algorithm for Unreliable Links. Programming and Computer Software, vol. 44, issue 6, 2018, pp. 381–387.

[40]. D. Sreenivasulu, P.V. Krishna. Deep Learning Based Efficient Channel Allocation Algorithm for Next Generation Cellular Networks. Programming and Computer Software, vol. 44, issue 6, 2018, pp. 428–434.

[41]. Mohammad Furqan Ali, Dushantha Nalin K. Jayakody, Yury A. Chursi et al. Recent Advances and Future Directions on Underwater Wireless Communications. Archives of Computational Methods in Engineering, vol. 27, 2019, pp. 1379–1412.

[42]. Р. Массобрио, С. Несмачнов, А. Черных и др. Применение облачных вычислений для анализа данных большого объема в умных городах. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 121-140. DOI: 10.15514/ISPRAS-2016-28(6)-9 / R. Massobrio, S. Nesmachnow, A. Tchernykh et al. Towards a Cloud Computing Paradigm for Big Data Analysis in Smart Cities. Programming and Computer Software, vol. 44, issue 6, 2018, pp. 181–189.

[43]. В.П. Козырев. Методы оценки времени выполнения в системах реального времени. Программирование, том 42, no. 1, 2016 г., стр. 39-50 / V.P. Kozyrev. Estimation of the execution time in real-time systems. Programming and Computer Software, vol. 42, issue 1, 2016, pp. 41–48.

[44]. M.W. Anwar, M. Rashid, F Azam et al. A model-driven framework for design and verification of embedded systems through SystemVerilog. Design Automation for Embedded Systems, vol. 23, 2019, pp. 179–223.

[45]. B. Kitchenham. Procedures for Performing Systematic Reviews. Joint Technical Report. Keele University, Empirical Software Engineering National ICT Australia Ltd., 2004, 33 p.

[46]. М.Б. Кузнецов. Трансформация UML-моделей и ее использование в технологии MDA. Программирование, том 33, no. 1, 2097 г., стр. 65-79 / M.B. Kuznetsov. UML model transformation

and its application to MDA technology. Programming and Computer Software, vol. 33, issue 1, 2007, pp 44–53.

[47]. Muhammad Waseem Anwar, Farooque Azam, Muazzam A. Khan, and Wasi Haider Butt. The Applications of Model Driven Architecture (MDA) in Wireless Sensor Networks (WSN) – Techniques and Tools. Lecture Notes in Networks and Systems, vol. 69, 2019, pp. 14-27.

[48]. Konstantinos Skiadopoulos, Athanasios Tsipis, Konstantinos Giannakis et al. Synchronization of data measurements in wireless sensor networks for IoT application. Ad Hoc Networks, vol. 89, 2019, pp. 47-57.

[49]. Sudeep Varshneya, Chiranjeev Kumara, Abhishek Swaroop. Leach Based Hierarchical Routing Protocol for Monitoring of Overground Pipelines Using Linear Wireless Sensor Networks. In Proc. of the 6th International Conference on Smart Computing and Communications, 2017, pp. 208–214.

Информация об авторах / Information about authors

Аиша ТАРИК – студент магистратуры кафедры вычислительной техники и программной инженерии электротехнического и машиностроительного колледжа. Область научных интересов: проектирование на основе моделей и беспроводные сенсорные сети.

Ayesha TARIQ, Master's Student at the Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering. Research interests include Model Driven Engineering and Wireless Sensor Networks.

Фарук АЗАМ – кандидат наук, профессор кафедры вычислительной техники и программной инженерии электротехнического и машиностроительного колледжа. Область научных интересов: проектирование на основе моделей, Web-разработка и беспроводные сенсорные сети.

Farooque AZAM, PhD, Professor at the Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering. Research interests include Model Driven Engineering, Web Development and Wireless Sensor Network.

Мухаммад Васим АНВАР – кандидат наук, старший научный сотрудник кафедры вычислительной техники и программной инженерии электротехнического и машиностроительного колледжа. Область научных интересов: проектирование на основе моделей, встраиваемые системы, управляющие системы и беспроводные сенсорные сети.

Muhammad Waseem ANWAR, PhD, Senior Researcher at the Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering. Research interests include Model Driven Engineering, Embedded Systems, Control Systems and Wireless Sensor Networks.

Тайиба ЗАХУР – студент магистратуры кафедры вычислительной техники и программной инженерии электротехнического и машиностроительного колледжа. Область научных интересов: проектирование на основе моделей и облачные вычисления.

Tayyba ZAHOOR, Master's Student at the Department of Computer and Software Engineering, College of Electrical and Mechanical Engineering. Research interests include Model Driven Engineering and Cloud Computing.

Абдул Вахаб МУЗАФФАР – кандидат наук, доцент кафедры информационных технологий Колледжа вычислительной техники и информатики. Область научных интересов: программная инженерия, интеллектуальный анализ данных и текстов, машинное обучение, электронное обучение, биомедицинские и биоинформационные системы.

Abdul Wahab MUZAFFAR, PhD in Software Engineering, Assistant Professor, Departement of IT, College of Computing and Infomatics. Research interests include Software Engineering, Data and Text Mining, Machine Learning, eLearning, Biomedical and bio informatics systems.

DOI: 10.15514/ISPRAS–2021–33(1)–8

Качество обслуживания в программно-определяемых сетях для научных приложений: возможности и проблемы

[1,2] *Х.Э. Лосано-Риск, ORCID: 0000-0002-6154-5712 <jlozano@cicese.mx>*
[1] *Р. Ривера-Родригес, ORCID: 0000-0002-1968-8525 <rrivera@cicese.mx>*
[2] *Х.И. Ньето-Иполито, ORCID: 0000-0003-0105-6789 <jnieto@uabc.edu.mx>*
[1] *С. Вильярреаль-Рейес, ORCID: 0000-0002-7219-361X <svillar@cicese.mx>*
[1] *А. Галавис-Москеда, ORCID: 0000-0001-7304-1442 <agalaviz@cicese.mx>*
[2] *М. Васкес-Брисеньо, ORCID: 0000-0003-2545-2153 <mabel.vazquez@uabc.edu.mx>*

[1] *Центр научных исследований и высшего образования (CICESE), Мексика, 22860, Нижняя Калифорния, Энсенада, ш. Тихуана-Энсенада, 3918*

[2] *Автономный университет Нижней Калифорнии (UABC), Мексика, 21100, Нижняя Калифорния, Энсенада*

Аннотация. Для работы научных приложений требуется выполнять в кратчайшее время обработку, анализ и передачу больших объемов данных из распределенных источников данных. Чтобы улучшить производительность таких приложений, требуется наличие определенных параметров качества обслуживания (QoS). С другой стороны, программно-определяемая сеть (SDN) представляет собой новую парадигму коммуникационной связи, которая упрощает управление коммуникационной инфраструктурой и, следовательно, позволяет динамически внедрять параметры QoS в приложения, работающие в сети такого типа. Учитывая обе парадигмы, мы выполнили данное исследование с целью найти ответы на следующие вопросы. Повышается ли производительность научных приложений, работающих в распределенных центрах данных, которые построены на основе SDN? Принимаются ли во внимание при планировании заданий сетевые параметры QoS? Методология исследования заключалась в ознакомлении со статьями из специализированных баз данных, содержащими ключевые слова «SDN», а также «высокопроизводительных вычисления и обработка больших объемов данных в научных приложениях». Затем мы проанализировали статьи, где эти ключевые слова пересекаются с некоторыми параметрами, связанными с QoS в сетях передачи данных. Кроме того, мы рассмотрели предложения по обеспечению качества обслуживания в SDN, чтобы выявить достигнутый прогресс в этой области исследований. По итогам исследования были получены следующие результаты: (i) внедрение QoS в научные приложения, работающие в SDN, все еще является открытой проблемой; (ii) мы определили проблемы, связанные с соединением обеих парадигм; (iii) мы представили стратегию обеспечения QoS научным приложениям, которые выполняются в распределенных центрах обработки данных на основе SDN.

Ключевые слова: качество обслуживания; программно-определяемая сеть; большие данные; высокопроизводительные вычисления

Для цитирования: Лосано-Риск Х.Э., Ривера-Родригес Р., Ньето-Иполито Х.И., Вильярреаль-Рейес С., Галавис-Москеда А., Васкес-Брисеньо М. Качество обслуживания в программно-определяемых сетях для научных приложений: возможности и проблемы. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 111-122. DOI: 10.15514/ISPRAS–2021–33(1)–8

Quality of Service in Software Defined Networks for Scientific Applications: Opportunities and Challenges

[1,2] *J.E. Lozano-Rizk, ORCID: 0000-0002-6154-5712 <jlozano@cicese.mx>*
[1] *R. Rivera-Rodriguez, ORCID: 0000-0002-1968-8525 <rrivera@cicese.mx>*
[2] *J.I. Nieto-Hipolito, ORCID: 0000-0003-0105-6789 <jnieto@uabc.edu.mx>*
[1] *S. Villarreal-Reyes, ORCID: 0000-0002-7219-361X <svillar@cicese.mx>*
[1] *A. Galaviz-Mosqueda, ORCID: 0000-0001-7304-1442 <agalaviz@cicese.mx>*
[2] *M. Vazquez-Briseño, ORCID: 0000-0003-2545-2153 <mabel.vazquez@uabc.edu.mx>*

[1] *Center for Scientific Research and Higher Education in Ensenada (CICESE), Baja California, Ensenada, Mexico, 22860*
[2] *Autonomous University of Baja California (UABC), Ensenada, Mexico, 21100*

Abstract. Scientific applications require to process, analyze and transfer large volumes of data in the shortest possible time from distributed data sources. In order to improve their performance, it is necessary to provide them with specific QoS parameters. On the other hand, SDN is presented as a new paradigm of communications networks that facilitates the management of the communications infrastructure and consequently allows to dynamically incorporate QoS parameters to the applications running in this type of network. With both these paradigms in mind, we conducted this research to answer the following questions: Do scientific applications that are running in an SDN-Enabled distributed data centers improve their performance? Do they consider network QoS parameters for job scheduling? The methodology used was to consult articles in specialized databases containing the keywords SDN and for scientific applications: HPC and Big Data. Then, we analyzed the articles where these keywords intersect with some of the parameters related to QoS in communications networks. Also, we reviewed QoS proposals in SDN to identify the advances in this research area. The results of this paper are: i) QoS is an open issue to incorporate in scientific applications that are running in an SDN ii) we identified the challenges to join both these paradigms, and iii) we present a strategy to provide QoS to scientific applications that are being executed among SDN-Enabled distributed data centers.

Keywords: Quality of Service; Software Defined Networks; Big Data; High Performance Computing

For citation: Lozano-Rizk J.E., Rivera-Rodriguez R., Nieto-Hipolito J.I., Villarreal-Reyes S., Galaviz-Mosqueda A., Vazquez-Briseño M. Quality of Service in Software Defined Networks for Scientific Applications: Opportunities and Challenges. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 111-122 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–8.

1. Введение

В настоящее время генерация цифровой информации происходит из различных видов источников данных, которые собираются географически распределенным образом со всего мира. Стремительное развитие технологий Интернета, Интернета вещей и больших данных привело к взрывному росту объема данных практически во всех областях промышленности, бизнеса [1] и науки.

С появлением научных приложений, связанных с высокопроизводительными вычислениями (High Performance Computing, HPC) и обработкой больших объемов данных (Big Data), возникла потребность в обработке и анализе больших данных в кратчайшее время, что приводит к чрезмерному использованию вычислительных ресурсов и ресурсов хранения данных в традиционных центрах обработки данных. Эти приложения также способствовали росту числа облачных инфраструктур и вычислительных кластеров, разделяющих ресурсы между географически распределенными центрами обработки данных, в которых научные приложения обрабатывают и анализируют данные, хранящиеся в распределенных системах.

Для работы этих видов научных приложений необходимо обрабатывать и передавать данные большого объема (терабайты или даже петабайты) между географически распределенными

центрами обработки данных с помощью традиционных сетевых систем. Этот процесс предполагает наличие высокой пропускной способности (приблизительно несколько десятков Гбит/с). Для администрирования сети и решения возможных проблем необходимы человеческие ресурсы. Кроме того, требуется выполнять задачи, связанные со специальными настройками, которые затрагивают, среди прочего, качество обслуживания (Quality of Service, QoS), шифрование, трансляцию сетевых адресов (Network Address Translation, NAT), таблицы управления доступом (Access Control List, ACL) и другие. Эти настройки напрямую влияют на то, как пакеты данных пересылаются в сети, включая то, как и когда они отбраковываются. Некоторые задачи являются статическими и определяются фиксированной конфигурацией каждого сетевого устройства.

В традиционной сети нет централизованного механизма изменения этих настроек в каждом сетевом устройстве на основе динамических условий сети или требований приложения [2]. Настройка производится сетевыми администраторами вручную, и это существенно влияет на производительность приложения, в частности, на время, требующееся для прекращения или завершения его работы.

Программно-определяемые сети (Software Defined Network, SDN) обеспечивают механизм, позволяющий научным приложениям динамически взаимодействовать с сетевым контроллером, чтобы использовать функциональные возможности для пересылки потока данных внутри сети. В соответствии с этой схемой, эти приложения могут запрашивать сетевые параметры или услуги, такие как пропускная способность, политики QoS, безопасность и др., обеспечивающие их оптимальной производительности.

Остальная часть статьи организована следующим образом: в разд. 2 представлена общая архитектура программно-определяемой сети. В разд. 3 описываются виды научных приложений, являющихся предметом данного исследования. В разд. 4 представлен ряд исследовательских работ, преимущественно посвященных тому, как научные приложения используют SDN. В разд. 5 представлен обзор исследовательских работ по качеству обслуживания (QoS) в SDN. В разд. 6 представлены возможности и проблемы QoS в SDN, решение которых может способствовать повышению эффективности научных приложений. Также в этом разделе мы предлагаем стратегию внедрения QoS в научные приложения при использовании распределенных центров обработки данных на основе SDN. Наконец, в разд. 7 содержится заключение.

2. Программно-определяемая сеть

Парадигма обеспечения интерфейса прикладного программирования (API), который позволяет приложениям, управляющим сетью, контролировать правила пересылки посредством программирования, связана с понятием «программно-определяемой сети» [3].

Коротко говоря, SDN – это сетевая архитектура, в которой логика управления сетью (уровень управления) отделяется от маршрутизаторов и коммутаторов (уровень данных), которые перенаправляют трафик в сети. Благодаря разделению уровней управления и данных, коммутаторы становятся элементами переадресации, а логика управления реализуется в централизованном контроллере, что упрощает реконфигурацию сети. Контроллер может взаимодействовать с коммутаторами (на уровне данных) через API, являющийся частью южного интерфейса (Southbound Interface, SBI), и отличным примером тому служит OpenFlow [4] – стандарт, который был разработан для SDN и в настоящее время реализуется в самых разнообразных сетях и на различном коммуникационном оборудовании.

Разработчики приложения могут пользоваться API северного интерфейса (Northbound Interface, NBI) сетевого контроллера, что позволяет им посылать инструкции в элементы переадресации (коммутаторы/маршрутизаторы). На уровне приложений (на высшем уровне) находятся приложения или службы, которые общаются через интерфейс NBI для реализации

управления сетью и поддержки ее функционирования: маршрутизаторы, балансировщики нагрузки, брандмауэры и т.д., а также бизнес-приложения [5], рис. 1.

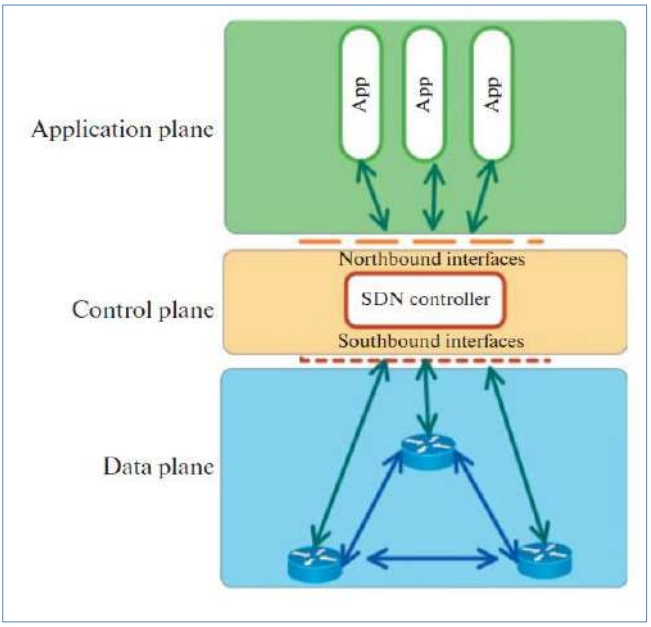


Рис. 1. Общая архитектура программно-определяемой сети
Fig 1. Software Defined Network General Architecture

Таким образом, SDN позволяет создавать научные приложения, которые взаимодействуют с сетевым контроллером с целью сокращения времени выполнения, запрашивая пересылку потока данных на сетевые узлы, которая удовлетворяет их требования пропускной способности, качества обслуживания и безопасности,

3. Научные приложения

В данном исследовании нас интересовали научные приложения, на производительность которых может влиять коммуникационная сеть. Принимая во внимание требования приложений к качеству обслуживания сети, их можно разделить на два класса:

- приложения, не требующие гарантий QoS, для выполнения которых достаточно отслеживать или регулировать пропускную способность сети; таким приложениям в основном требуется массивная передачи данных.
- приложения категории HPC, для которых необходимо гарантированно обеспечить QoS, учитывая такие параметры, как пропускная способность, задержка, колебания задержки, доступность службы, эффективная скорость передачи, скорость потери пакетов и размер круговой задержки (Round-Trip Delay Time, RTT).

В своем исследовании мы изучали те типы научных приложений, для улучшения производительности которых требуется гарантированно обеспечить некоторые сетевые сервисы. Кроме того, мы ограничились анализом параллельных приложений на основе MPI, обрабатывающих большие данные с использованием MapReduce.

3.1 Параллельные вычислительные приложения на основе MPI

MPI – это стандарт, широко используемый при разработке приложений с параллельными вычислениями для передачи сообщений между серверами или вычислительными узлами.

MPI позволяет одновременно выполнять несколько экземпляров программы в разных узлах многопроцессорной системы и обеспечивает связь между процессами [6].

Хотя традиционный MPI используется для разработки крупномасштабных научных приложений, одним из основных факторов повышения их эффективности является использование коллективной коммуникация [7], когда коммуникация возможна между двумя процессами MPI или большим их числом одновременно. Некоторые коллективные операции MPI предназначены для коллективных вычислений, например, такие MPI_Reduce и MPI_ALLreduce. На выполнение таких операций расходуется много времени, а их результаты требуется пересылать в другие процессы.

Эти операции могли бы повысить производительность крупномасштабных параллельных приложений, если бы приложения MPI могли управлять такими параметрами SDN, как пропускная способность и величина задержки, или применять политики QoS в соответствии со своими требованиями, то это сократило бы время отклика или полного завершения работы.

3.2 Приложения для высокопроизводительных вычислений и обработки больших объемов данных

В некоторых областях исследований анализ больших данных в настоящее время считается одной из основных задач. MapReduce – это модель программирования для обработки больших объемов данных с использованием параллельных/распределенных алгоритмов на кластерных, облачных вычислительных системах и грид-системах [8] для получения более быстрых результатов по сравнению с традиционной (последовательной) обработкой. Модель программирования MapReduce основана на двух основных функциях – map (распределение данных по узлам и их параллельная предварительная обработка) и reduce (параллельная свертка разделенных данных). Одной из наиболее используемых реализаций MapReduce является фреймворк Hadoop [9], разработанный Apache Foundation, основанный на Java и распределенной файловой системе HDFS (Hadoop Distributed File System) и оптимизированный для поддержки модели MapReduce [10].

Уже выполнялись исследовательские работы, в которых изучалось выполнение задач MapReduce с использованием центров обработки данных, подключенными программно-определяемой сетью, с учетом параметров сети, таких как пропускная способность, в процессе планирования [11], а в некоторых случаях также учитывалась задержка. В следующем разделе мы рассмотрим этот тип приложений и его взаимодействие с программно-определяемой сетью.

4. SDN и приложения HPC-Big Data

В этом разделе мы отвечаем на следующие вопросы:

- повышается ли производительность приложений высокопроизводительных вычислений и обработки больших объемов данных, если они работают в центрах обработки данных на основе SDN?
- учитываются ли параметры QoS сети в процессе планирования заданий приложения?

Ответы на эти вопросы помогут определить потенциал области исследований SDN, который может быть применен к приложениям HPC-Big Data для повышения их производительности.

Далее мы описываем серию исследовательских работ, направленных на сокращение времени выполнения приложений, основанных на MPI и MapReduce (Big Data) и работающих в вычислительных кластерах, соединенных программно-определяемыми сетями.

Алгоритм, используемый в GSBT [12], ориентирован только на получение кратчайших маршрутов в SDN для использования в дереве редукции без учета неблагоприятных сетевых условий, таких как перегрузка сети, задержка и потеря пакетов. В BLAR [13] оптимизированы

два наиболее важных параметра для высокопроизводительных вычислительных приложений – пропускная способность и задержка [14]. Другие работы в основном были направлены на оптимизацию пропускной способности (BASS [15], PYTHIA [16], CLS [17], ASETS [18]).

Несмотря на то, что даже простой учет этих сетевых параметров при планировании задач позволил повысить производительность приложений, некоторые результаты показывают, что в отдельных случаях производительность приложений в SDN ухудшалась. Это может быть связано с тем, что не учитывались такие состояния сети, как перегрузка канала, потери пакета, задержка, колебание задержки, доступность сервиса, эффективная скорость передачи и т.д. Таким образом, в перечисленных исследовательских работах следовало бы учесть указанные параметры сети, чтобы обеспечить должное качество обслуживания приложениям HPC-Big Data.

5. Качество обслуживания в SDN

QoS считается одним из существенных свойств, влияющих на производительность приложения, которое интенсивно использует коммуникационную сеть [17]. В контексте сети QoS рассматривается как способность обеспечения сервиса. Наиболее распространенные параметры для обеспечения QoS – это пропускная способность, задержка, коэффициент потери пакетов (Packet Loss Rate, PLR) и управление перегрузками [17] [18]. Выбор сетевых параметров QoS зависит от типа приложения.

В табл. 1 приведена сводка рассмотренных нами предложений по поводу QoS для тех случаев, в которых архитектура SDN формируется на основе подхода обеспечения качества обслуживания. Анализ этих предложений показывает потребность в дополнительном модуле, который взаимодействует с контроллером SDN для обеспечения качества обслуживания. В большинстве случаев в алгоритме маршрутизации QoS используется кратчайший маршрут (Shortest Part First, SPF), основанный на ограничениях, при этом, помимо учета расстояния между сетевыми узлами, в некоторых случаях учитывается и задержка между ними. Большинство предложений по QoS сосредоточены на управлении пропускной способности и перенаправлении потока данных на те каналы, где требования к пропускной способности выполнены. В основном данные предложения касаются мультимедийных приложений (потокового видео) и даже VoIP, поэтому неясно, в каких случаях можно за счет этого улучшить производительность приложений HPC-Big Data.

Табл. 1. Общие аспекты предложений по обеспечению качества обслуживания
Table.1. QoS Proposal General Aspects

Предложение	**Сетевые параметры**	**Алгоритм маршрутизации**	**Целевое приложение**
Q-Ctrl [19]	Пропускная способность	Управление пропускной способностью	Облачные мультимедийные службы
OpenQoS [20]	Пропускная способность, задержка, колебания задержки	SPF с ограничением задержки	Потоковое видео, мультимедиа
Flow QoS [21]	Пропускная способность	Задаваемый пользователем приоритет для каждого приложения	Мультимедийные сервисы, VoIP
QoS control [22]	Пропускная способность, задержка	Задаваемые пользователем политики приоритетов и управление пропускной способностью	Потоковое видео, мультимедиа
VSDN [24]	Пропускная способность,	SPF с ограниченной пропускной способностью и задержкой	Передача видео

	задержка, джиттер		
Q-Ctrl [26]	Пропускная способность	Задаваемая пользователем приоритетность и управление пропускной способностью	Мультимедийные сервисы, VoIP
CECT [27]	Пропускная способность	SPF на основе ограничений и задержка	Потоки передачи данных (не указано)
Amoeba Net [28]	Пропускная способность	SPF с ограничением пропускной способности	Потоки передачи данных (не указано)

Для эффективного использования приложений высокопроизводительных вычислений и обработки больших объемов данных требуются реализация многоцелевого алгоритма QoS, в котором должны учитываться такие параметры сети, как пропускная способность, двухточечная задержка, коэффициент потери пакетов и колебание задержки. Кроме того, требуется разработка стратегий, позволяющих пользоваться преимуществами SDN. В рассмотренных нами предложениях по поводу QoS эксперименты проводились только в пределах одного домена и с одним сетевым контроллером. Ни в одном из случаев не рассматривалось качество обслуживания при наличии более одного домена, иными словами, при использовании архитектуры, в которой приложения HPC и Big Data выполняются в более чем одном распределенном центре данных. При наличии такого сценария необходим модуль QoS для установления связи с контроллерами каждого задействованного домена, а также для взаимодействия с поставщиками услуг Интернет (Internet Service Provider, ISP). Следует заметить, что пропускная способность является параметром SDN, который используется в большинстве рассмотренных работ, но это не означает, что управление пропускной способностью равнозначно обеспечению QoS в SDN.

6. Возможности и проблемы

Рассмотренные нами работы показывают, что качество обслуживания сети является одним из главных факторов улучшения производительности научных приложений, работающих в программно-определяемой сети. При этом обеспечение качества обслуживания в SDN представляет собой одну из серьезнейших проблем, в частности, в процессе выбора сетевых путей, которые лучше соответствуют требованиям каждого приложения. Процесс выбора пути должен быть основан на политиках QoS, определяемых дополнительным модулем на уровне приложений или управления, взаимодействующим с контроллером SDN. Таким образом, процесс перенаправления потока данных будет основан на политиках QoS, а не только на использовании кратчайшего пути (что является традиционной схемой маршрутизации сетевого контроллера) или с учетом лишь одного сетевого параметра. В процессе планирования заданий приложения решение о распределении заданий по вычислительных узлах будет основывается на сетевых политиках QoS, четко определенных для каждого приложения. Ниже приведен перечень основных проблем обеспечения QoS в программно-определяемой сети для приложений HPC и Big Data:

1) Требуется разработать классификацию видов QoS в соответствии с типами приложений HPC и Big Data и уровнями приоритетов их потоков данных в SDN. В [29] автор классифицирует разновидности QoS в IP-сетях в соответствии с требованиями приложений к таким параметрам сети, как двухточечная задержка, колебания задержки и коэффициент потери пакетов (PLR). На основе максимальных значений сетевых параметров, а также контрольных значений задержки (см. табл. 2) предлагается классифицировать приложения HPC и Big Data по их функциональным характеристикам. С этой точки зрения для некоторых из этих приложений требуется массивная передача данных или интенсивная коммуникация с вычислительными узлами распределенных центров обработки данных. Эти приложения могут относиться к классам QoS уровней 0 и уровня 2. После классификации

приложениq контроллер SDN назначает более высокий приоритет правилам потоков данных приложениям из классов высоких уровней, чтобы было эти приложения могли выполнять передачу данных в кратчайшее время.

Табл. 2. Классы обеспечения качества обслуживания и максимальные значения показателей сети
Table 2. QoS Classes and Network Metric Upper Limits [29]

Класс QoS	Характеристики	Задержка (мс)	Колебания задержки (мс)	PLR
0	Режим реального времени, высокая чувствительность к колебаниям задержки, много сетевых коммуникаций	100	50	$1x10^{-3}$
1	Режим реального времени, чувствительность к колебаниям задержки, имеются сетевые коммуникации	400	50	$1x10^{-3}$
2	Транзакционные данные, много сетевых коммуникаций	100	не установлено	$1x10^{-3}$
3	Транзакционные данные, имеются сетевые коммуникации	400	не установлено	$1x10^{-3}$
4	Низкий уровень потерь (короткие транзакции, веб, потоковое видео)	1000	не установлено	$1x10^{-3}$
5	Традиционные приложения (с негарантированной скоростью доставки)	не установлено	не установлено	не установлено

2) Требуется выполнять процесс выбора сетевого пути и перенаправления потока данных с учетом политик QoS, основанных на приоритетности сетевых требований каждого приложения. Для выбора пути необходимо определить многоцелевой алгоритм QoS, чтобы учитывать четыре наиболее важных сетевых параметра, которые могут повлиять на производительность приложений HPC и Big Data: сквозная пропускная способность, задержка, колебания задержки и потеря пакетов. Также требуется учитывать приоритет, присвоенный каждому приложению. Для оптимизации QoS [30] в процессе переадресации потока могут быть применены эволюционные алгоритмы.

3) Требуется разработать интеллектуальный планировщик, динамически управляющий политиками QoS для приложений HPC и Big Data. Интеллектуальный планировщик, который удовлетворяет политикам QoS, требуемым для приложений, может быть разработан для распределения потоков данных приложения по узлам сети. Кроме того, он может динамически корректировать метод перенаправления потоков данных. У планировщика должна иметься информация о состоянии каждого выполняемого приложения, чтобы он мог динамически корректировать политики QoS для каждого приложения с целью поддержки требуемого качества обслуживания, способствующего повышению производительности. Кроме того, в политиках QoS может учитываться возможность энергосберегающего планирования при распределении заданий по узлам сети [31].

4) Требуется создать архитектуру QoS с учетом возможности выполнения приложений HPC и Big Data в узлах нескольких распределенных центров данных (междоменный подход). В рамках этой архитектуры мы предлагаем модель, в которой вычислительным узлам центра данных A требуется передавать потоки данных в вычислительные узлы центра

обработки данных B. Одним из наиболее важных аспектов, который необходимо учитывать, является потребность во взаимодействии контроллеров SDN всех центров данных для обеспечения возможности координировать политики QoS, определенные для междоменных приложений. Для создания подобной архитектуры может понадобиться разработка алгоритмов для работы с мультиконтроллерами или разработка стратегий, позволяющих координировать политики QoS при наличии нескольких контроллеров SDN. Такой подход мог бы повысить скорость анализа больших данных для Интернета вещей, когда большие объемы данных хранятся в географически распределенных облачных средах [32], и улучшить работу высокопроизводительных приложений, поддерживающих парадигмы MPI и OpenMPI и работающих в программно-определяемом центре данных [33].

Для решения каждой из этих проблем требуются анализ, исследования и даже разработка новых алгоритмов. Однако, рассматривая их совместно, мы тем самым предлагаем стратегию обеспечения качества обслуживания для повышения производительности приложений HPC и Big Data, работающих в распределенных центрах обработки данных на осноке SDN.

5. Заключение

В данном исследовании было проанализировано несколько работ, в которых приложения высокопроизводительных вычислений и обработки больших объемов данных используют программно-определяемую сеть для планирования своих задач и заданий. Несмотря на то, что в большинстве работ отмечается улучшение производительности приложений благодаря учетом некоторых сетевых параметров, наблюдались также случаи, когда они ухудшали операционные характеристики приложения. Кроме того, ни в одном из предложений (GSBT, BLAR, BASS, PYTHIA, CLS) не рассматривались политики QoS, которые требуются приложениям. В связи с этим мы выявили ряд возможностей и проблем, связанных с обеспечением качества обслуживания в SDN, а также то, как решение этих проблем может способствовать повышению производительности приложений.

Рассмотренные предложения по обеспечению качества обслуживания в SDN предоставляют ряд возможностей для повышения производительности приложений HPC и Big Data. Эти предложения частично решают некоторые проблемы, которые могут возникнуть у данного типа приложений. Однако по результатам нашего исследования можно сделать вывод о том, что имеется ряд проблем (см. пп. 1-4 в предыдущем разделе), решению которых следует посвятить новые исследования, связанные с динамическим и специализированным обеспечением качества обслуживания для приложений высокопроизводительных вычислений и обработки больших объемов данных и направленные на минимизацию времени выполнения приложения, а также оптимизацию использования вычислительных ресурсов центров данных.

Список литературы / References

[1]. A. Cravero. Big Data Architectures and the Internet of Things: A Systematic Mapping Study. IEEE Latin America Transactions, мол. 16, то. 4, 2018, pp. 1219-1226.

[2]. W. Stallings. Software-Defined Networks and OpenFlow. Internet Protocol Journal, vol. 16, no. 1, 2013, pp. 1-6.

[3]. I. Monga, E. Pouyoul and C. Gouk. Software-Defined Networking for Big-Data Science - Architectural Models from Campus to the WAN. In Proc. of the SC Companion: High Performance Computing, Networking, Storage and Analysis (SCC), 2012, pp. 1629-1635.

[4]. Openflow. Open Networking Foundation. Available at: https://www.opennetworking.org/, last accessed: December, 2019.

[5]. D. Kreutz, F. Ramos, P. Verissimo, E. Rothenberg, S. Azodolmolky and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. Proceedings of the IEEE, vol. 103, no.1, 2015, pp. 14-76.

[6]. С.П. Копысов, И.В. Красноперов, В.Н. Рычков. Совместное использование систем промежуточного программного обеспечения CORBA и MPI. Программирование, том 32, no. 3, 2006 г., стр. 51-61 /

S.P. Kopysov, I.V. Krasnopyorov, and V.N. Rychkov. CORBA and MPI code coupling. Programming and Computer Software, vol. 32, no. 3, 2006, pp. 276–283.

[7]. R. Thakur and W. Groop, Open Issues in MPI Implementation. In Proc. of the 12th Asia-Pacific Computer Systems Architecture Conference (ACSAC), 2007, pp. 327-338.

[8]. Р. Массобрио, С. Несмачнов, А. Черных, А. Аветисян, Г. Радченко. Применение облачных вычислений для анализа данных большого объема в умных городах. Труды ИСП РАН, том 28, вып. 6, 2016 г., стр. 121-140. DOI: 10.15514/ISPRAS-2016-28(6)-9 / R. Massobrio, S. Nesmachnow, A. Tchernykh, A. Avetisyan, and G. Radchenko. Towards a Cloud Computing Paradigm for Big Data Analysis in Smart Cities. Programming and Computer Software, vol. 44, no. 3, 2018, pp. 181-189.

[9]. C. Jayalath, J. Stephen, and P. Eugster. From the Cloud to the Atmosphere: Running MapReduce across Data Centers. IEEE Transactions on Computers, Vol. 63, No. 1, 2014, pp. 74-87.

[10]. S. Deshmukh, J. Aghav, and R. Chakravarthy. Job Classification for MapReduce Scheduler in Heterogeneous Environment. In Proc. of the IEEE International Conference on Cloud and Ubiquitous Computing and Emerging Technologies (CUBE), 2013, pp. 26-29.

[11]. Y. Watashiba, K. Kido, S. Date et al. Prototyping and evaluation of a network-aware Job Management System on a cluster system. In Proc. of the 19th IEEE International Conference on Networks (ICON), 2013, pp. 1-6.

[12]. P. Makpaisit, K. Ichikawa, and P. Uthayopas. MPI Reduce Algorithm for OpenFlow-Enabled Network. In Proc. of the 15th International Symposium on Communications and Information Technologies (ISCIT), 2015, pp. 261-264,

[13]. P. Uchupala, K. Ichikawa et al. Application-Oriented Bandwidth and Latency Aware Routing with OpenFlow Network. In Proc. of the IEEE 6th International Conference on Cloud Computing Technology and Science, 2014, pp. 775-780.

[14]. J. Huang, L. Xu, M. Zeng, C. Xing, Q. Duan, and Y. Yan, Hybrid Scheduling for Quality of Service Guarantee in Software Defined Networks to support Multimedia Cloud Services. In Proc. of the IEEE International Conference on Services Computing, 2015, pp. 788-792.

[15]. Q. Peng, B. Dai, B. Huang, and G. Xu. Bandwidth-Aware Scheduling with SDN in Hadoop: A New Trend for Big Data. IEEE Systems Journal, vol. 11, no. 4, 2017, pp. 2337-2344.

[16]. M. Veiga, C. Rose, K. Katrinis and H. Franke. Pythia: Faster Big Data in Motion through Predictive Software-Defined Network Optimization at Runtime. In Proc. of the IEEE 28th International Parallel & Distributed Processing Symposium, 2014, pp. 82-90.

[17]. H. Alkaff, I. Gupta and L. Leslie. Cross-Layer Scheduling in Cloud Systems. In Proc. of the IEEE International Conference on Cloud Engineering (IC2E), 2015, pp. 236-245.

[18]. S. Jamalian, H. Rajaei. ASETS: A SDN Empowered Task Scheduling System for HPCaaS on the Cloud. In Proc. of the IEEE International Conference on Cloud Engineering (IC2E), 2015, pp. 329-334.

[19]. K. Govindarajan, K. Meng, H. Ong, and W. Tat. Realizing the Quality of Service (QoS) in Software-Defined Networking (SDN) Based Cloud Infrastructure. In Proc. of the 2nd International Conference on Information and Communication Technology (ICoICT), pp. 505-510, 2014.

[20]. H. Egilmez, S. Dane, K. Bagci, and A. Tekalp. OpenQoS: An OpenFlow Controller Design for Multimedia Delivery with End-to-End Quality of Service over Software-Defined Networks. In Proc. of the Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012, pp. 1-8.

[21]. M. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y. Song. FlowQoS: Per-Flow Quality of Service for Broadband Access Networks. SCS Technical Report GT-CS-15-02, Georgia Institute of Technology, 2015.

[22]. M. Karaman, B. Gorkemli, S. Tatlicioglu, M. Komurcuoglu, and O. Karakaya. Quality of Service Control and Resource Priorization with Software Defined Networking. In Proc. of the 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1-6.

[23]. H. Owens and A. Durresi. Explicit Routing in Software-Defined Networking (ERSDN): Addressing Controller Scalability. In Proc. of the 17th International Conference on Network-Based Information Systems, 2014, pp. 128-134.

[24]. H. Owens and A. Durresi. Video over Software-Defined Networking (VSDN). In Proc. of the 16th International Conference on Network-Based Information Systems, 2013, pp. 44-51.

[25]. O. Younis and S. Fahmy. Constraint-based Routing in the Internet: Basic Principles and Recent Research. IEEE Communications Surveys, vol. 5, no. 1, 2003, pp. 2-13.

[26]. S. Tomovic, N. Prasad, and I. Radusinovic. SDN control framework for QoS provisioning. In Proc. of the 22nd Telecommunications Forum, 2014, pp. 111-114.

[27]. M. Tajiki, B. Akbari, M. Shojafar et al. CECT: computationally efficient congestion-avoidance and traffic engineering in software-defined cloud data centers. Cluster Computing, vol. 21, no. 4, 2018, pp. 1881-1897.

[28]. A. Shah, W. Wu, Q. Lu et al. AmoebaNet: An SDN-enabled network service for big data science. Journal of Network and Computer Applications, vol. 119, 2018, pp. 70-82.

[29]. M. Marchese. QoS over heterogeneous networks. John Wiley & Sons, 2007, 328 p.

[30]. С.Д. Итурриага Фабра, С.Е. Несмачнов Кановас, Н. Гони Бофриско, Б. Дорронзоро Диаз, А.Н. Черных. Конструирование и оптимизация сетей распространения контента. Труды ИСП РАН, том 31, вып. 2, 2019 г., стр. 15-20. DOI: 10.15514/ISPRAS-2019-31(2)-1 / S. Iturriaga, S. Nesmachnow, G. Goñi, B. Dorronsoro, and A. Tchernykh. Evolutionary Algorithms for Optimizing Cost and QoS on Cloud-based Content Distribution Networks. Programming and Computer Software, vol. 45, no. 8, 2019, pp. 544-556.

[31]. Ф. Армента-Кано, А. Черных, Х.М. Кортес-Мендоза и др. C.Min_c: стратегия неоднородной концентрации задач для энергосберегающих компьютерных расписаний. Труды ИСП РАН, том 27, вып. 6, 2015 г., стр. 355-380. DOI: 10.15514/ISPRAS-2015-27(6)-23 / F.A. Armenta-Cano, A. Tchernykh, J.M. Cortes-Mendoza et al. Min_c: Heterogeneous concentration policy for energy-aware scheduling of jobs with resource contention. Programming and Computer Software, vol. 43, no. 3, 2017, pp. 204-215.

[32]. M. Marjani, F. Nasaruddin, A. Gani, A. Karim et al. Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges. IEEE Access, vol. 5, 2017, pp. 5247-5261.

[33]. Б.М. Шабанов, О.И. Самоваров. Принципы построения межведомственного центра коллективного пользования общего назначения в модели программно-определяемого ЦОД. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 7-24. DOI: 10.15514/ISPRAS-2018-30(6)-1 / B.M. Shabanov and O.I. Samovarov. Building the Software-Defined Data Center. Programming and Computer Software, vol. 45, no. 8, 2019, pp. 458–466.

Информация об авторах / Information about authors

Хосе Элено ЛОСАНО-РИСК получил степень кандидата наук в 2019 году в UABC. В настоящее время он работает в компьютерном отделе CICESE и является преподавателем на факультете наук UABC. Его исследовательские интересы: высокопроизводительные вычисления, аналитика больших данных, распределенные системы, Интернет вещей (IoT) и программно-определяемые сети.

Jose Eleno LOZANO-RIZK received a Doctor of Science degree in 2019 from the UABC. He currently works at the Computer Department at CICESE Research Center in the Telematics Division and is a member of the Faculty of Sciences' academic staff at UABC Ensenada. His research interests are high-performance computing, big data analytics, distributed systems, internet of things (IoT), and software-defined networks.

Рауль РИВЕРА-РОДРИГЕС получил степень получил степень кандидата наук в UABC в 2010 году. В настоящее время он является директором отдела телематики CICESE. Его исследовательские интересы включают системы управления сетями, QoS в IP-сетях, обработку сигналов для беспроводной связи, кибербезопасность, облачные вычисления, межуровневое проектирование и теорию кодирования.

Raul RIVERA-RODRIGUEZ received his Ph.D. degree from the UABC in 2010. He is currently the Director of the Telematics Division, CICESE Research Center. His research interests include network management systems, QoS in IP networks, signal processing for wireless communications, cybersecurity, cloud computing, cross-layer design, and coding theory.

Хуан Иван НЬЕТО-ИПОЛИТО получил степень кандидата наук в Политехническом университете Каталонии, Испания в 2005 г. В настоящее время он является профессором UABC. Его научные интересы затрагивают компьютерную инженерию, компьютерные сети, мобильные вычисления.

Juan Iván NIETO-HIPÓLITO received the Ph.D. degree from Polytechnic University of Catalonia, Spain. He is currently a Full Professor of Baja California Autonomous University, Mexico. His research interests are in the areas computer engineering, computer networks, mobile computing.

Сальвадор ВИЛЛАРРЕАЛЬ-РЕЙС – доктор философии, исследователь. Научные интересы: Интернет вещей, системы электронного здравоохранения, радио по оптоволокну, беспроводная связь.

Salvador VILLARREAL-REYES – Ph.D., researcher. Research interests include Internet of Things, e-Health Systems, Radio over Fiber, Wireless Communications.

Алехандро ГАЛАВИС-МОСКЕДА получил степень кандидата наук в CICESE в 2013 году. В настоящее время он является исследователем. Его основные исследовательские интересы включают мобильные и беспроводные сети для интеллектуальных транспортных систем и мобильного здравоохранения.

Alejandro GALAVIZ-MOSQUEDA received Ph.D. degree from the CICESE in 2013. He is currently a researcher. His main research interests include mobile and wireless networks for intelligent transport systems and m-health.

Мабель ВАСКЕС-БРИСЕНО имеет степень кандидата наук. В настоящее время она работает на факультете архитектуры, дизайна и инженерии. Основной исследовательский интерес - мобильные вычисления.

Mabel VAZQUEZ-BRISENO has Ph.D. degree. Currently she works at the Faculty of Architecture, Design and Engineering, Main research interest is mobile computing.

DOI: 10.15514/ISPRAS–2021–33(1)–9

Безопасная реализация виртуальной сети на плоскости данных SDN

[1] *И.Б. Бурдонов, ORCID: 0000-0001-9539-7853 <igor@ispras.ru>*
[1,2] *Н.В. Евтушенко, ORCID: 0000-0002-4006-1161 <evtushenko@ispras.ru>*
[1] *А.С. Косачев, ORCID: 0000-0001-5316-3813 <kos@ispras.ru>*

[1] *Институт системного программирования РАН им. В.П. Иванникова, 109004, Россия, г. Москва, ул. А. Солженицына, д. 25*
[2] *Национальный исследовательский университет «Высшая школа экономики», 101000, Россия, г. Москва, ул. Мясницкая, д. 20*

Аннотация. В статье исследуется задача виртуализации сети на плоскости данных программно-конфигурируемой сети, моделируемой графом физических связей между узлами сети. Виртуальная сеть задается как множество упорядоченных пар хостов (отправитель, получатель), а реализуется множеством путей хост-хост, однозначно определяющим настройки коммутаторов. Возможности передачи пакетов ограничиваются весами (приоритетами) хостов: пакет может быть передан только от хоста к хосту с не меньшим приоритетом. Соответственно множество путей допустимое, если любое подмножество связываемых им пар хостов является допустимым. В работе показывается, что в отличие от случая, когда любая пара различных хостов является допустимой, в графе с приоритетами не для любого множества пар допустимых хостов существует допустимая реализация, т.е. реализация в виде допустимого множества путей. Кроме того, показывается, что в ряде случаев, когда такая реализация существует, она не всегда возможна без путей с циклами, т.е. путей, допускающих бесконечное движение пакетов по циклу, и без дублирующих путей, когда хост получает один и тот же пакет несколько раз. С использованием понятия совершенного множества путей сформулировано и доказано требование к графу с приоритетами, которое достаточно для допустимой реализации любого допустимого множества пар хостов без циклов с возможным дублированием.

Ключевые слова: программно-конфигурируемые сети; виртуализация сети; безопасность; приоритеты хостов; (приоритетно-)допустимая реализация множества пар хостов

Для цитирования: Бурдонов И.Б., Евтушенко Н.В., Косачев А.С. Безопасная реализация виртуальной сети на плоскости данных SDN. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 123-136. DOI: 10.15514/ISPRAS–2021–33(1)–9.

Secure Implementing a Virtual Network on the SDN Data Plane

[1] *I.B. Burdonov, ORCID: 0000-0001-9539-7853 <igor@ispras.ru>*
[1,2] *N.V. Yevtushenko, ORCID: 0000-0002-4006-1161 <evtushenko@ispras.ru>*
[1] *A.S. Kossatchev, ORCID: 0000-0001-5316-3813 <kos@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, 20, Myasnitskaya st., Moscow, 101000, Russia*

Abstract. The paper continues the investigations on the implementation of virtual networks on the SDN data plane which is modeled by a graph of physical connections between network nodes. A virtual network is defined as a set of ordered host pairs (sender, receiver), and it is implemented by a set of host-host paths that uniquely determine the switch settings. The opportunities to transmit a packet are limited by the host weights (priorities): a packet can be only transmitted from a host to a host if the sender has at most the same priority as the recipient, and thus, a set of paths is permissible if its every subset connects permissible host pairs. In the paper, it is proven that differently from the case when every host pair is permissible, in the graph with priorities a permissible path implementation does not exist for every set of permissible hosts. Moreover, it is shown that is some cases when such an implementation exists, the implementation is not possible without paths with cycles where packets can move infinite and without duplicate paths when a host can get the same packet several times. Using the notion of a perfect set of paths a criterion is established when every permissible set of hosts can be safely implemented by a set of paths without cycles but possibly with duplicate paths.

Ключевые слова: software defined networks (SDN); network virtualization; security; host priority; permissible implementation of the host set

For citation: Burdonov I.B., Yevtushenko N.V., Kossatchev A.S. Secure Implementing a Virtual Network on the SDN Data Plane. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 123-136 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–9.

1. Введение

Одной из основных технологий виртуализации [1-5] в настоящее время являются программно-конфигурируемые сети (SDN) с разделенными плоскостями данных и управления. Пакеты между хостами пересылаются на плоскости данных через промежуточные коммутаторы, система правил которых (настройка) осуществляется специальными SDN-контроллерами. Правило определяет, каким соседним узлам пересылается принятый коммутатором пакет в зависимости от того, откуда пришел пакет, и от вектора параметров в заголовке пакета [6]. Иными словами, настройка коммутаторов определяет множество путей от хоста к хосту, по которым и будут пересылаться пакеты. Ситуация может быть промоделирована с использованием графа физических связей, вершинами которого являются хосты и коммутаторы, а ребра соответствуют физическим связям между ними, при этом каждый хост может быть соединен только с одним коммутатором.

В работе [7] обсуждаются вопросы возможности реализации заданного множества пар (хост, хост) через подходящие пути хост-хост в графе физических связей, которые, в свою очередь, определяют те или иные настройки коммутаторов. Как известно, при решении этой задачи возникают три эффекта. 1) Как показано в [4-5] при реализации (через подходящие настройки коммутаторов) некоторого множества реберно-простых путей, связывающих заданные пары (хост, хост), на панели данных могут появляться непредусмотренные пути хост-хост, т.е. пути, которых нет в этом множестве путей и которые могут связывать пары хостов, отсутствующие в заданном множестве пар хостов. Для учёта этого эффекта при реализации нужно рассматривать только замкнутые по дугам множества реберно-простых путей, которые не меняются при их реализации через настройки коммутаторов. 2) При реализации

пар хостов могут появиться циклы, по которым пакеты будут передаваться бесконечно и, соответственно, бесконечно размножаться. 3) Кроме того, могут появиться дублирующие пути, из-за чего хост-адресат получает один и тот же пакет не один, а несколько раз. В [7] показано, что строгая реализация заданного множества пар хостов, при которой пути связывают все заданные пары хостов и только такие пары хостов, не всегда возможна без появления циклов и/или без дублирования. В то же время нестрогая реализация, когда требуется связать все заданные пары хостов, но разрешено связывать и непредусмотренные пары хостов, всегда возможна без появления циклов и без дублирования.

В данной статье рассматривается задача реализации множества пар хостов с учётом политики безопасности (security), согласно которой некоторые хосты не могут передавать пакеты определенным хостам. Для описания множества хостов, от которых заданный хост, может принимать пакеты, каждому хосту приписано натуральное число (*приоритет*), и хост может передавать пакеты только хостам с не меньшими приоритетами. Соответственно, упорядоченные пары хостов разделяются на (приоритетно-) *допустимые* и *недопустимые*. Все пары из заданного множества пар хостов должны быть допустимыми, но при реализации этого множества разрешается связывать путями и другие допустимые пары хостов. В то же время недопустимые пары хостов ни при каких условиях не должны связываться путями. Такое ослабление требования о непредусмотренных парах хостов даёт больше свободы по сравнению со строгой реализацией, но ограничивает свободу нестрогой реализации, а именно для заданных пар хостов допускается нестрогая реализация, но не любая, а только та, которая удовлетворяет требованиям безопасности.

Таким образом, реализующее множество путей тоже должно быть (приоритетно-) допустимым, т.е. не должно нарушать правило приоритетов, и не иметь циклов, по которым пакеты будут циркулировать бесконечно и бесконечно размножаться. Кроме того, в ряде случаев для сокращения нагрузки на сеть желательно, чтобы множество не содержало дублирующих путей, т.е. разных путей, соединяющих одну и ту же пару хостов. В работе показано, что в отличие от подобной задачи (произвольной нестрогой реализации) для графа без приоритетов, допустимая реализация не всегда возможна, и аналогично строгой реализации не всегда возможна без дублирования и не всегда возможна без циклов.

Сформулировано и доказано требование к графу, которое достаточно для допустимой реализации любого допустимого множества пар хостов без циклов с возможным дублированием. С этой целью понятие почти совершенного множества путей из [7] расширяется на граф с приоритетами, и требование формулируется как наличие в графе множества путей, которое соединяет все допустимые пары хостов и в котором пути после слияния (с проходом по одной или нескольким общим дугам) не разделяются. Если требуется отсутствие дублирования, то в достаточном условии после разделения путей запрещается их слияние (с проходом по общей дуге).

2. Основные понятия

Графом физических связей (далее просто графом) будем называть связный неориентированный граф $G = (V, E)$ без кратных ребер и петель, где V – множество коммутаторов и хостов, $E \subseteq V \times V$ – множество ребер, моделирующих физические связи между коммутаторами и между коммутаторами и хостами. Поскольку ребро, соединяющее вершины a и b, неориентированное и нет кратных ребер, его можно обозначать как ab, так и ba. Поскольку нет петель, ребер вида aa в E нет. Поскольку нет кратных ребер, путь как последовательность смежных ребер однозначно задается последовательностью вершин $a_1 \ldots a_n$, через которые он проходит. Если путь проходит по ребру ab из a в b, то будем говорить, что он проходит дугу ab. Если x и y хосты, то путь из x в y, в котором все остальные вершины коммутаторы, будем называть полным и обозначать как xy-путь. Путь, в котором вершины (дуги) не повторяются, называется вершинно-простым (реберно-простым).

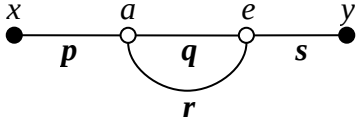
Вершины графа будем обозначать строчными буквами *a, b, c, ... x, y, z*, пути – жирными строчными буквами ***p, q, r,***…, а множества путей – прописными буквами – *P, Q, R, ...* На рисунках коммутаторам соответствуют белые кружки, а хостам — чёрные кружки.

Мы будем предполагать, что каждый хост x подсоединен ровно к одному коммутатору [3]. Поэтому хост – это терминальная вершина графа, т.е. вершина степени 1. Если коммутатор a имеет степень 1 и соединен с вершиной *b*, то любой полный путь, проходящий через *a*, имеет вид *...bab...*; удаляя из него все циклы *bab*, получаем путь, не проходящий через *a*. Это значит, что такой коммутатор «лишний», и достаточно рассматривать графы, в которых терминальные вершины – это только хосты. Множества хостов и коммутаторов обозначим через *H* и *S*, соответственно; $H \cup S = V$, $H \cap S = \varnothing$.

В общем случае правило коммутатора *b* имеет вид σ*abc*, где *a* и *c* соседи *b*, а σ вектор значений параметров заголовка пакета, которые используются в правилах. Такое правило означает, что коммутатор *b*, получив пакет с вектором σ от соседа *a*, пересылает его соседу *c*. В настоящей работе предполагается, что коммутатор не меняет σ. Тем самым, для вектора σ порождаются полные пути вида $a_1 \ldots a_n$, где для $i = 2 \,..\, n - 1$ в коммутаторе a_i есть правило $\sigma a_{i-1} a_i a_{i+1}$, хост a_1 соединен с коммутатором a_2 и хост a_n соединен с коммутатором a_{n-1}. Если в коммутаторе есть два правила σ*abc* и σ*abc*`, где $c \neq c$`, говорят, что пакет клонируется, т.е. пересылается обоим соседям *c* и *c*`.

Заданное множество *P* полных путей однозначно определяет минимальный набор правил коммутаторов, порождающий все пути из *P* [5]. Однако это не значит, что порождаются только пути из *P*. Будем говорить, что два пути *сливаются* на дуге *ab* в вершине *a*, если у них дуга *ab* общая и не первая, а непосредственно предшествующие ей дуги *ca* и *c*`*a* разные (т.е. $c \neq c$`), и *разделяются* после дуги *de* в вершине *e*, если у них дуга *de* общая и не последняя, а непосредственно следующие дуги *ef* и *ef*` разные (т.е. $f \neq f$`).

Цикл порождается, если полный *xy*-путь проходит через некоторую дугу дважды, т.е. путь имеет вид ***p****ab****q****ab****s***, где *a* и *b* коммутаторы. Чтобы понять, каким образом путь с циклами связан с понятиями слияния и разделения по дуге обозначим путь с циклами как ***p****a****q****e****r***(*a****q****e****r***)**a****q****e****s***, где отрезок ***p*** начинается в хосте *x*, отрезки ***p*** и ***r*** не заканчиваются в одной вершине, после этих отрезков в пути следует коммутатор *a*, отрезок *a****qer*** проходится несколько раз, после коммутатора *e* отрезки ***r*** и ***s*** не начинаются в одной вершине, и отрезок ***s*** заканчивается в хосте *y* (см. рис. 1). Двигаясь вдоль пути из хоста *x* в хост *y*, мы видим, что в вершине *a* путь сливается сам с собой, потом в вершине *e* разделяется сам с собой, а затем это слияние и разделение происходит ещё раз (если путь проходит цикл *aqer* *k* раз, то $(k + 1)$ раз встречается как разделение после слияния, так и слияние после разделения). Пакеты будут не только бесконечно ходить по циклу *aqer*, но и бесконечно клонироваться в вершине *e*, так что хост *y* будет получать бесконечное число клонов пакета.



коммутатор *b* находится на отрезке (*a, e*],
т.е. на $b \in Im(\boldsymbol{q})$ или $b = e$

Рис. 1. Порождение цикла
Fig. 1. Cycle Generation

Путь, который не сливается сам с собой, это реберно-простой путь. Для отсутствия циклов необходимо, чтобы все пути множества *P* были реберно-простыми. Но этого недостаточно. Если два реберно-простых полных пути из *P* после слияния на дуге *ab* разделяются (после этой же или другой дуги), т.е. имеют вид *x****p****ab****q****y* и *x*`*p*`*ab****q***`*y*` с разными начальными и конечными хостами $x \neq x$` и $y \neq y$`, то порождаются и новые пути *x****p****ab****q***`*y*` и *x*`***p***`*ab****q****y*. Эта операция порождения новых путей называется замыканием по дугам, а результат замыкания по дугам всех пар путей из *P* обозначается $P{\downarrow}{\uparrow}$ [4-5]. Очевидно, $P \subseteq P{\downarrow}{\uparrow}$. Если $P \neq P{\downarrow}{\uparrow}$, т.е.

P не замкнуто по дугам, то возникают непредусмотренные пути. В частности, могут возникнуть не реберно-простые пути и, следовательно, циклы. Появление циклов в замыкании по дугам множества полных путей всегда свидетельствует о бесконечности этого замыкания и, тем самым, наличии дублирования. Если множество P полных реберно-простых путей конечно и замкнуто по дугам, то в нем циклов нет.

Для множества полных путей P через $H(P) \subseteq H \times H$ обозначим множество пар xy, для которых в P есть xy-путь. Множество пар хостов $D \subseteq H \times H$, не содержащее пар вида xx, будем называть *нормальным*. Будем говорить, что нормальное множество D *(нестрого) реализуется* замкнутым по дугам множеством полных путей P, если $D \subseteq H(P)$. Множество D *реализуется без циклов*, если P конечно, и *строго реализуется*, если $D = H(P)$. Множество D *реализуется без дублирования*, если P содержит ровно один xy-путь для каждой пары $xy \in D$.

Как отмечается в работе [7], если адрес хоста-отправителя (хоста-получателя) входит в вектор параметров σ, то проблемы со строгой реализацией без циклов и дублирования любого множества пар разных хостов не существует. Действительно, если адрес хоста-отправителя входит в вектор параметров σ, то правила коммутатора для векторов параметров с разными адресами хоста-отправителя работают независимо друг от друга. Для каждого хоста-отправителя x в графе можно выбрать исходящее из x дерево I_x кратчайших путей, ведущих во все остальные хосты. Для любого множества D пар разных хостов и любого хоста x выбирается подмножество D_x пар, где первый элемент пары – это хост x, а в дереве I_x – поддерево $I_x(D)$, в котором листовые вершины – это вершины y такие, что $xy \in D_x$. В исходящем дереве все пути реберно-простые (даже вершинно-простые), и нет слияния, тем самым, нет и разделения после слияния. Поэтому $I_x(D)$ замкнуто по дугам и, очевидно, строго реализует D_x без циклов и дублирования, причём используются кратчайшие полные пути. Реализация множества пар разных хостов оказывается подмножеством одного и того же множества путей – объединения деревьев I_x по всем хостам x. Аналогичная процедура с аналогичным результатом применима тогда, когда в вектор параметров σ входит адрес хоста-получателя. Только здесь строится входящее дерево O_x для каждого хоста x.

Одна из причин, по которой адрес хоста-отправителя и адрес хоста-получателя не включают в вектор параметров σ и не используют в правилах коммутаторов, заключается в том, что при таком использовании число правил коммутатора зависит от числа хостов в сети (растёт вместе с ним). Поэтому в данной работе мы рассматриваем реализацию множества разных пар хостов для случая, когда адрес хоста-отправителя и адрес хоста-получателя не входят в вектор параметров σ. Остальные параметры никак не влияют на перемещение пакетов с данным вектором σ, поэтому мы будем опускать σ в обозначении правила и писать вместо σ*abc* просто *abc*.[1] Вектор σ определяет идентификатор потока, в котором передаются векторы с такими параметрами. Иными словами, правила коммутатора (для данного вектора σ) определяют, кому должен быть послан пакет, только в зависимости от соседа, от которого пакет принят. В этом случае число правил, по которым работает коммутатор, зависит только от числа его соседей и не зависит от числа хостов в сети.

Политика безопасности (security) может запрещать передавать пакеты от хоста к некоторым другим хостам, а такая передача может случиться, если появляются «непредусмотренные» пути, когда заданное множество пар хостов реализуется не замкнутым по дугам множеством путей. Для описания множества хостов, от которых данный хост может принимать пакеты, каждому хосту h приписан приоритет — натуральное число, обозначаемое $A(h)$. Имеется в виду, что чем больше приоритет, тем больше прав хоста по доступу, т.е. к получению пакетов. Будем считать, что пакет m имеет приоритет $A(m)$, равный приоритету хоста-отправителя h: $A(m) = A(h)$, и хост $h`$ может принимать пакет $m$, если $A(m) \le A(h`)$. Граф G, хостам которого

[1] В настоящей работе рассматриваются пакеты потока с одним идентификатором, и соответственно, при описании правил и путей опускается вектор параметров σ.

приписаны приоритеты, будем называть графом с приоритетами и обозначать G^*. На рисунках приоритет хоста обозначается числом через запятую после идентификатора хоста.

Полный путь p от хоста h к хосту $h`$ назовём *допустимым*, если $A(h) \le A(h`)$, и *недопустимым* в противном случае. Множество P полных путей *допустимое*, если каждый путь из его замыкания по дугам $P{\downarrow}{\uparrow}$ является допустимым. Замкнутое по дугам множество полных путей допустимое тогда и только тогда, когда оно состоит из допустимых путей. Нормальное множество пар хостов, не нарушающее правило приоритетов, т.е. $\forall\ (x, y) \in D\ \ A(x) \le A(y)$, будем называть *допустимым*. Будем говорить, что допустимое множество D *допустимо реализуется*, если оно может быть реализовано замкнутым по дугам множеством допустимых полных путей P.

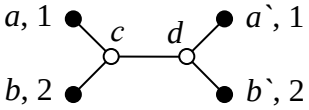
3. Допустимая реализация versus циклы и дублирование

В этом разделе мы исследуем связь допустимой реализации множества пар хостов с наличием или отсутствием циклов и дублирования. В статье [7] нами показано, что для любого нормального множества пар хостов существует реализация без циклов и дублирования (утверждение 1), которая возможно не является строгой, однако, как показывает следующее утверждение, это не всегда выполняется, если требовать, чтобы реализация была допустимой.

Утверждение 1. Допустимая реализация не всегда возможна: существует такой граф с приоритетами G^*, на котором некоторое допустимое множество пар хостов D может быть реализовано, но не может быть допустимо реализовано.

Доказательство. На рис. 2 показан пример такого графа G^* и множества D. Легко убедиться, что D допустимое множество пар хостов. Рассмотрим множество $P = \{\ acda`, bcdb`\ \}$. Легко убедиться, что это множество полных путей и $D \subseteq H(P)$. Следовательно, замыкание по дугам $P{\downarrow}{\uparrow}$ является реализацией D. Однако это замыкание содержит путь $bcda`$, который недопустимый, поскольку $A(b) = 2 > 1 = A(a`)$.

Пусть теперь P произвольное замкнутое по дугам множество допустимых полных путей, реализующее множество D. Любые два $aa`$-путь и $bb`$-путь имеют общую дугу cd, поэтому в множестве P должны оказаться $ba`$-путь, однако этот путь недопустимый. Следовательно, реализация множества P не является допустимой.



$D = \{\ (a, a`), (b, b`)\ \}$

Рис. 2. Допустимое множество D не имеет допустимой реализации
Fig. 2. A permissible set D has no permissible implementation

Пример на рис. 2 показывает также, что строгая реализация (не обязательно допустимая) не всегда возможна: наличие $aa`$-пути и $bb`$-пути в замыкании по дугам порождает $ab`$-путь и $ba`$-путь. Как отмечено выше, если не требовать допустимости (и строгости) реализации, то она всегда возможна.

В работе [7] исследуются также возможности строгой реализации нормального множества пар хостов без циклов и дублирования, и показано, что строгая реализация не всегда возможна без дублирования (утверждение 3) и не всегда возможна без циклов (утверждение 4). В данной работе мы показываем, что при реализации допустимого множества пар хостов свойства из утверждений 3 и 4 [7] сохраняются и для нестрогой, но допустимой, реализации этого множества.

Для полного пути $\boldsymbol{p}$ через $\boldsymbol{p}^\circ$ [7] обозначим путь, который получается из $\boldsymbol{p}$ применением, пока возможно, следующей операции удаления циклов: путь $\boldsymbol{p} = \boldsymbol{qaras}$ превращается в путь $\boldsymbol{qas}$. Заметим, что результат операции "°", вообще говоря, неоднозначный. Для однозначности

будем считать, что удаление цикла применяется только тогда, когда каждая вершина префикса ***q*** имеет только одно вхождение в ***p***, а ***r*** не содержит вхождений вершины *a*. Иными словами, среди всех вершин, имеющих несколько вхождений в ***p***, выбирается та вершина *a*, вхождение которой встретилось первым, и удаляется цикл от этого первого вхождения *a* до ее второго вхождения. Например, для ***p*** = *xacbcaby*, ***p***° = *xaby* (не *xacby*). Для множества полных путей *P* обозначим через *P*° множество путей, получаемых удалением циклов из всех путей *P*, т.е. *P*° = { ***p***° | ***p*** ∈ *P* }.

Утверждение 2. Пусть *P* множество полных путей. Тогда множество *P*° состоит из вершинно-простых полных путей и связывает те же самые пары хостов, что множество *P*: *H*(*P*°) = *H*(*P*). Если *P* замкнуто по дугам, то *P*° тоже замкнуто по дугам. Если *P* не содержит дублирующих путей, то *P*° тоже не содержит дублирующих путей. Если *P* конечно, то *P*° тоже конечно. Если *P* допустимое, то *P*° тоже допустимое.

Доказательство. Очевидно, что удаление всех циклов из полного пути делает путь вершинно-простым и оставляет его полным. Поэтому *P*° состоит из вершинно-простых полных путей. Очевидно, что операция удаления одного цикла из одного пути не меняет *H*(*P*). Следовательно, цепочка таких операций также не меняет *H*(*P*). Следовательно, *P*° связывает те же самые пары хостов, что множество *P*: *H*(*P*°) = *H*(*P*).

Докажем, что если *P* замкнуто по дугам, то *P*° тоже замкнуто по дугам. Пусть *P*° содержит пути ***p****ab****q*** и ***p***$_1$*ab****q***$_1$ с общей дугой *ab*. Эти пути получены удалением циклов из путей ***r*** и ***r***$_1$, соответственно, имеющихся в *P*. При удалении циклов непустая последовательность вершин между вхождением *a* и вхождением *b* может быть полностью удалена только вместе с удалением вхождения *a* и/или *b*. Поэтому дуга *ab* может остаться в результате удаления циклов только в том случае, если она была в исходном пути и некоторое ее вхождение не лежало на удаляемых циклах. Тогда это вхождение дуги *ab* разбивает путь на три отрезка: отрезок до *a*, данное вхождение дуги *ab* и отрезок после *b*, причем циклы не содержат данное вхождение дуги *ab*, т.е. любой цикл целиком лежит либо на отрезке до *a*, либо на отрезке после *b*. Следовательно, пути ***r*** и ***r***$_1$ можно представить в виде ***p***`*ab****q***` и ***p***`$_1$*ab****q***`$_1$, соответственно, где ***p*** = ***p***`°, ***p***$_1$ = ***p***`$_1$°, ***q*** = ***q***`°, ***q***$_1$ = ***q***`$_1$°. Поскольку *P* замкнуто по дугам, в нем имеются пути ***p***`*ab****q***`$_1$ и ***p***`$_1$*ab****q***`. А тогда в *P*° имеются пути ***p****ab****q***$_1$ = ***p***`°*ab****q***$_1$`° = (***p***`*ab****q***$_1$`)° и ***p***$_1$*abq* = ***p***`$^1$°*ab****q***`° = (***p***`$_1$*ab****q***`)°. Следовательно, множество *P*° замкнуто по дугам.

Результатом операции “°”, примененной к одному пути, является один путь с теми же начальным и конечным хостом. Отсюда непосредственно следует: если *P* не содержит дублирующих путей, то *P*° тоже не содержит дублирующих путей; если *P* конечно, то *P*° тоже конечно; если *P* допустимое, то *P*° тоже допустимое.

□

Данное выше определение множества *P*° то же самое, что в [7], а доказанное выше утверждение 2 дополняет утверждение 2 из [7].

В следующих утверждениях мы показываем, что в отличие от произвольной нестрогой реализации нормального множества пар хостов, допустимая реализация допустимого множества пар в некоторых случаях невозможна как без циклов, так и без дублирования.

Утверждение 3. Допустимая реализация не всегда возможна без дублирования: существует такой граф с приоритетами *G**, на котором некоторое допустимое множество *D* пар хостов может быть допустимо реализовано, но только с дублированием.

Доказательство. На рис. 3 показан пример графа *G** и множества *D*. Легко убедиться, что *D* – допустимое множество пар хостов. Рассмотрим множество *P* = { $x_0a_0a_1b_1b_4a_4b_0y_0$, $x_0a_0b_3a_3a_2b_2b_0y_0$, $x_0a_0a_1b_1y_1$, $x_0a_0b_3a_3a_2b_2y_2$, $x_1a_1b_1b_4a_4b_0y_0$, $x_1a_1b_1y_1$, $x_2a_2b_2b_0y_0$, $x_2a_2b_2y_2$, $x_3a_2b_3y_3$, $x_4a_4b_4y_4$ }. Легко убедиться, что множество *P* состоит из допустимых полных путей, замкнуто по дугам и *D* ⊆ *H*(*P*) (и *D* = *H*(*P*)). Следовательно, множество *P* допустимо (и строго)

реализует множество D. Однако множество P содержит дублирующие пути $x_0a_0a_1b_1b_4a_4b_0y_0$ и $x_0a_0b_3a_3a_2b_2b_0y_0$.

Пусть теперь $P`$ некоторая допустимая реализация множества $D$, т.е. $P`$ есть замкнутое по дугам множество, такое, что все пути в $P`$ допустимые, и $H(P`) \supseteq D$. Согласно утверждению 2 мы можем считать, что $P`$ состоит из вершинно-простых путей. Допустим, что $P`$ не содержит дублирующих путей. В графе G коммутаторы образуют цикл $a_0b_3a_3a_2b_2b_0a_4b_4b_1a_1a_0$, и любой вершинно-простой путь может проходить только часть этого цикла по часовой или против часовой стрелки.

Покажем, что x_4y_4-путь из $P`$ должен проходить цикл коммутаторов по часовой стрелке, т.е. это должен быть путь $x_4a_4b_4y_4$. Допустим обратное: $x_4y_4$-путь из $P`$ проходит цикл коммутаторов против часовой стрелки, т.е. это путь $x_4a_4b_0b_2a_2a_3b_3a_0a_1b_1b_4y_4$. Если хотя бы один x_iy_j-путь из $P`$, где $(x_i, y_j) \in D \setminus \{(x_4, y_4)\}$, проходит цикл коммутаторов тоже против часовой стрелки, то в замыкании по дугам появляется $x_4y_j$-путь, где $j \neq 4$. А поскольку $P`$ замкнуто по дугам, этот путь есть и в $P`$, но такой путь недопустимый, поскольку хост $x_4$ имеет приоритет 4, который больше чем приоритет любого хоста $y_j$, где $j \neq 4$. Следовательно, все $x_iy_j$-пути из $P`$, где $(x_i, y_j) \in D \setminus \{(x_4, y_4)\}$, проходят цикл коммутаторов по часовой стрелке. Среди этих путей есть пути $x_1a_1a_0b_3a_3a_2b_2b_0y_0$ и $x_2a_2b_2y_2$, которые в замыкании по дугам порождают путь $x_1a_1a_0b_3a_3a_2b_2y_2$, который в силу замкнутости по дугам множества $P`$, должен быть в $P`$, но этот путь недопустимый, так как $A(x_1) = 2 > 1 = A(y_2)$. Мы пришли к противоречию, следовательно, x_4y_4-путь из $P`$ проходит цикл коммутаторов по часовой стрелке, т.е. это путь $x_4a_4b_4y_4$.

Покажем, что x_3y_3-путь из $P`$ должен проходить цикл коммутаторов против часовой стрелки, т.е. это должен быть путь $x_3a_3b_3y_3$. Действительно, в противном случае в $P`$ есть пути $x_3a_3a_2b_2b_0a_4b_4b_1a_1a_0b_3y_3$ и $x_4a_4b_4y_4$, которые в замыкании по дугам порождают путь $x_4a_4b_4b_1a_1a_0b_3y_3$, который в силу замкнутости по дугам множества $P`$, должен быть в $P`$, но этот путь недопустимый, так как $A(x_4) = 4 > 3 = A(y_3)$. Мы пришли к противоречию, следовательно, x_3y_3-путь из $P`$ проходит цикл коммутаторов против часовой стрелки, т.е. это путь $x_3a_3b_3y_3$.

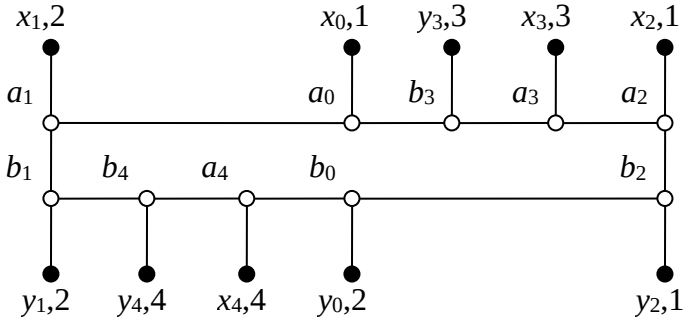
Покажем, что в $P`$ нет x_2y_1-пути. Допустим противное, и тогда этот путь проходит цикл коммутаторов либо по часовой стрелке, либо против часовой стрелки. В первом случае это путь $x_2a_2b_2b_0a_4b_4b_1y_1$, который вместе с путем $x_4a_4b_4y_4$ порождает в замыкании по дугам x_4y_1-путь $x_4a_4b_4b_1y_1$, который в силу замкнутости по дугам множества $P`$ должен быть в $P`$, но этот путь недопустимый, так как $A(x_4) = 4 > 2 = A(y_1)$. Во втором случае это путь $x_2a_2a_3b_3a_0a_1b_1y_1$, который вместе с путем $x_3a_3b_3y_3$ порождает в замыкании по дугам x_3y_1-путь $x_3a_3b_3a_0a_1b_1y_1$, который в силу замкнутости по дугам множества $P`$ должен быть в $P`$, но этот путь недопустимый, так как $A(x_3) = 3 > 2 = A(y_1)$. Мы пришли к противоречию, следовательно, наше допущение не верно, и в $P`$ нет x_2y_1-пути.

Также в $P`$ нет x_1y_2-пути, поскольку этот путь недопустимый: $A(x_1) = 2 > 1 = A(y_2)$.

Рассмотрим пути из $P`$, которые начинаются в хостах $x_0$, $x_1$, $x_2$ и заканчиваются в хостах $y_0$, $y_1$, $y_2$. Поскольку в $P`$ нет дублирующих путей, и мы показали, что в $P`$ нет $x_2y_1$-пути и $x_1y_2$-пути, то из 9 возможных путей в $P`$ может быть только 7 путей. И эти 7 путей должны быть в $P`$, поскольку в множестве D есть соответствующие пары хостов.

Для того чтобы попасть из хоста x_i в хост y_j, где $i = 0, 1, 2$ и $j = 0, 1, 2$, путь должен пройти либо по дуге a_1b_1, либо по дуге a_2b_2. Пусть для $t = 1, 2$ имеется m_t путей, которые проходят дугу a_tb_t, и эти пути начинаются в n_t хостах и заканчиваются в k_t хостах. Тогда $n_1 + n_2 = 3$, $k_1 + k_2 = 3$, $m_1 + m_2 = 7$. Из замкнутости по дугам множества $P`$ следует $n_1k_1 = m_1$, $n_2k_2 = m_2$. Отсюда $n_1k_1 + (3 - n_1)(3 - k_1) = 7$, что влечет $2n_1k_1 = 3(n_1 + k_1) - 2$. Поскольку $k_1 \leq 3$ и $n_1 \leq 3$, имеем: $3k_1 = 2$ для $n_1 = 0$, $k_1 = -1$ для $n_1 = 1$, $k_1 = 4$ для $n_1 = 2$, $3k_1 = 7$ для $n_1 = 3$. Каждое из этих уравнений не имеет решения в целых неотрицательных числах или противоречит условию $k_1 \leq 3$.

Мы пришли к противоречию, следовательно, наше допущение не верно, и в $P`$ есть дублирующие пути.



$$D = \{ (x_0, y_0), (x_0, y_1), (x_0, y_2), (x_1, y_0), (x_1, y_1), (x_2, y_0), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$$

Рис. 3. Множество D допустимо реализуется только с дублированием
Fig. 3. The set D is permissible only with duplication

Отметим, что в соответствующем утверждении 3 из [7] доказано аналогичное утверждение для строгой реализации, при условии, что передача пакетов возможна для любой пары различных хостов: строгая реализация не всегда возможна без дублирования. Пример на рис. 3 является усложнённым вариантом примера на рис. 1 из [7]. В обоих случаях во множестве D содержится 7 из 9 пар хостов вида (x_i, y_j), где $i = 0, 1, 2$ и $j = 0, 1, 2$, и рассматриваются только вершинно-простые пути. Но в [7] число 7 путей, связывающих эти хосты, непосредственно следует из строгости реализации, а в этой статье для нестрогой, но допустимой, реализации нам потребовалось запретить 2 из 9 возможных путей. Один x_1y_2-путь мы запретили соответствующей расстановкой приоритетов, а другой x_2y_1-путь мы запретили с помощью введения четырех дополнительных хостов x_3, y_3, x_4, y_4 с подходящими приоритетами, четырех коммутаторов a_3, b_3, a_4, b_4 и двух пар добавленных хостов (x_3, y_3) и (x_4, y_4).

Утверждение 4. Допустимая реализация не всегда возможна без циклов: существует такой граф с приоритетами G^*, на котором некоторое допустимое множество D пар хостов допустимо реализуемо, но только бесконечными замкнутыми множествами путей.

Доказательство. Рассмотрим пример на рис. 4. Множество D допустимое и допустимо (и строго) реализуется замыканием по дугам $P{\downarrow}{\uparrow}$ множества полных путей P. Это замыкание допустимое, поскольку его пути соединяют только хосты с равными приоритетами. Но в P есть пути $x_1a_1b_1c_2c_1a_2b_2y_2$ и $x_2a_2b_2d_2d_1a_1b_1y_1$, которые в $P{\downarrow}{\uparrow}$ порождают не реберно-простой путь $x_1a_1b_1c_2c_1a_2b_2d_2d_1a_1b_1y_1$ (проходит дважды по дуге a_1b_1), т.е. $P{\downarrow}{\uparrow}$ бесконечно. Пусть замкнутое по дугам множество допустимых путей $P`$ реализует множество $D$. Допустим, оно конечно. Тогда по утверждению 2 мы можем выбрать множество $P`$, состоящее из вершинно-простых путей. Коммутаторы образуют цикл $a_1b_1c_2c_1a_2b_2d_2d_1a_1$, вершинно-простой путь может проходить только часть этого цикла по часовой или против часовой стрелки.

1. Пусть u_1u_2-путь $\boldsymbol{p}_1$ идет по часовой стрелке, тогда $\boldsymbol{p}_1 = u_1d_1a_1b_1c_2c_1a_2b_2d_2u_2$.

1.1. Если x_2y_2-путь $\boldsymbol{p}_2$ идет по часовой стрелке, тогда $\boldsymbol{p}_2 = x_2a_2b_2y_2$; u_1u_2-путь $\boldsymbol{p}_1$ и x_2y_2-путь $\boldsymbol{p}_2$ имеют общую дугу a_2b_2, что в замыкании по дугам порождает u_1y_2-путь, но этот путь недопустимый: $A(u_1) = 2 > 1 = A(y_2)$. Значит $\boldsymbol{p}_2$ идет против часовой стрелки, тогда $\boldsymbol{p}_2 = x_2a_2c_1c_2b_1a_1d_1d_2b_2y_2$.

1.2. Если x_1y_1-путь $\boldsymbol{p}_3$ идет по часовой стрелке, тогда $\boldsymbol{p}_3 = x_1a_1b_1y_1$; u_1u_2-путь $\boldsymbol{p}_1$ и x_1y_1-путь $\boldsymbol{p}_3$ имеют общую дугу a_1b_1, что в замыкании по дугам порождает u_1y_1-путь, но этот путь

недопустимый: $A(u_1) = 2 > 1 = A(y_1)$. Значит $\boldsymbol{p}_3$ идет против часовой стрелки, тогда $\boldsymbol{p}_3 = x_1a_1d_1d_2b_2a_2c_1c_2b_1y_1$.

1.3. Из 1.1 и 1.2 следует, что x_2y_2-путь $\boldsymbol{p}_2$ и x_1y_1-путь $\boldsymbol{p}_3$ имеют общую дугу d_2b_2, что в замыкании по дугам порождает путь $x_2a_2c_1c_2b_1a_1d_1d_2b_2a_2c_1c_2b_1y_1$, который не является реберно-простым, так как дважды проходит по дуге a_2c_1, т.е. порождается цикл и, следовательно, бесконечное множество путей, что противоречит конечности множества P`.

Следовательно, u_1u_2-путь $\boldsymbol{p}_1$ идет против часовой стрелки, т.е. $\boldsymbol{p}_1 = u_1d_1d_2u_2$.

2. В силу симметрии аналогично доказывается (рассматривая v_1v_2-путь, x_1y_1-путь и x_2y_2-путь), что v_1v_2-путь $\boldsymbol{p}_4$ идет против часовой стрелки, т.е. $\boldsymbol{p}_4 = v_1c_1c_2v_2$.

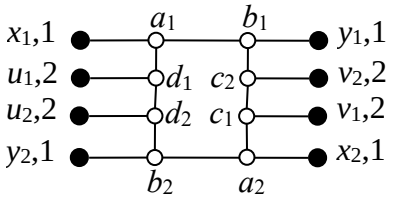
3. Если x_1y_2-путь $\boldsymbol{p}_5$ идет против часовой стрелки, тогда $\boldsymbol{p}_5 = x_1a_1d_1d_2b_2y_2$; u_1u_2-путь $\boldsymbol{p}_1$ и x_1y_2-путь $\boldsymbol{p}_5$ имеют общую дугу d_1d_2, что в замыкании по дугам порождает u_1y_2-путь, но этот путь недопустимый: $A(u_1) = 2 > 1 = A(y_2)$.

Следовательно, $\boldsymbol{p}_5$ идет по часовой стрелке, т.е. $\boldsymbol{p}_5 = x_1a_1b_1c_2c_1a_2b_2y_2$.

4. В силу симметрии аналогично доказывается (рассматривая x_2y_1-путь и v_1v_2-путь), что x_2y_1-путь $\boldsymbol{p}_6$ идет по часовой стрелке, тогда $\boldsymbol{p}_6 = x_2a_2b_2d_2d_1a_1b_1y_1$.

Но тогда пути $\boldsymbol{p}_5$ и $\boldsymbol{p}_6$ имеют общую дугу a_2b_2 и в замыкании по дугам порождают путь $x_1a_1b_1c_2c_1a_2b_2d_2d_1a_1b_1y_1$, который не является реберно-простым, так как дважды проходит по дуге a_1b_1, т.е. порождается цикл и, следовательно, бесконечное множество путей, что противоречит конечности множества P`.

Мы пришли к противоречию и, следовательно, наше допущение не верно, и P` бесконечно.



$D = \{ x_1y_1, x_1y_2, x_2y_2, x_2y_1, u_1u_2, v_1v_2 \}$
$P = \{ x_1a_1b_1y_1, x_1a_1b_1c_2c_1a_2b_2y_2,$
$x_2a_2b_2y_2, x_2a_2b_2d_2d_1a_1b_1y_1,$
$u_1d_1d_2u_2, v_1c_1c_2v_2 \}$

Рис. 4. Множество D допустимо реализуется только бесконечным множеством путей
Fig. 4. The set D may be permissible implemented only by an infinite set of paths

Отметим, что в соответствующем утверждении 4 из [7] доказано аналогичное утверждение для строгой реализации, при условии, что все хосты имеют одинаковый приоритет: строгая реализация не всегда возможна без циклов. Пример на рис. 4 аналогичен примеру на рис. 2 из [7]. Заметим, что в публикации [7] допущены ошибки в рис. 4 и доказательстве утверждения 4. Правильный рисунок – это рис. 4 в данной статье. Доказательство аналогично доказательству утверждения 4 в данной статье, но только игнорируются приоритеты, и вместо фразы «этот путь недопустимый» следует читать «этот путь соединяет пары хостов, отсутствующие в заданном множестве D».

В обоих случаях (в данной статье и в исправленной статье [7]) пути из u_1 и v_1 не должны заканчиваться в y_1 и y_2. Но в [7] это следует из строгости реализации, а в данной статье для нестрогой, но допустимой, реализации это следует из правила приоритетов: пакет не должен пересылаться из хоста с большим приоритетом в хост с меньшим приоритетом.

Аналогичный пример демонстрирует «бóльшую свободу» допустимой нестрогой реализации. Для этого достаточно назначить хостам y_1 и y_2 приоритет 3. Тогда становится возможной допустимая реализация без циклов и дублирования, например, на основе множества кратчайших путей, соединяющих пары хостов из множества D: $\{ x_1a_1b_1y_1, x_1a_1d_1d_2b_2y_2, x_2a_2c_1c_2b_1y_1, x_2a_2b_2y_2, u_1d_1d_2u_2, v_1c_1c_2v_2 \}$. Замыкание по дугам этого множества конечно (не порождает циклов), не содержит дублирующих путей, но содержит два дополнительных пути $u_1d_1d_2b_2y_2$ и $v_1c_1c_2b_1y_1$, связывающих пары хостов (u_1, y_2) и (v_1, y_1),

которые нельзя связывать при строгой реализации, но можно связывать при допустимой реализации.

4. Достаточное условие допустимой реализации любого множества пар хостов без циклов и дублирования

В этом разделе мы исследуем достаточные условия на граф с приоритетами, позволяющие допустимо реализовывать без циклов и дублирования любые допустимые множества пар хостов.

В [7] введены понятия *разделения после слияния* и *слияния после разделения*. Если для двух путей есть слияние на дуге ab и есть разделение после дуги cd, то будем говорить, что разделение происходит после слияния, если хотя бы в одном из этих путей сначала проходится дуга ab, а потом дуга cd. Соответственно, слияние происходит после разделения, если хотя бы в одном из этих путей сначала проходится дуга cd, а потом дуга ab.

Также в [7] доказаны следующие утверждения:

Утверждение 5. Если множество полных путей конечно, то отсутствие разделения после слияния достаточно, но не необходимо для замкнутости по дугам и отсутствия циклов.

Утверждение 6. Для замкнутого множества полных путей условие отсутствия слияния после разделения необходимо и достаточно для отсутствия дублирования.

Также в [7] даны определения (почти) хорошего графа, (почти) совершенного множества путей и (почти) совершенного графа. Для допустимой, но возможно нестрогой, реализации мы предлагаем следующие аналогичные определения.

Граф, в котором любое допустимое множество пар хостов можно допустимо реализовать без циклов, назовём *почти допустимо-хорошим*. Граф, в котором любое допустимое множество пар хостов можно допустимо реализовать без циклов и без дублирования, назовём *допустимо-хорошим*.

Конечное замкнутое множество P допустимых путей, связывающее все пары хостов, не нарушающие правило приоритетов, т.е. все пары хостов (x, y), где $x \neq y$ и $A(x) \leq A(y)$, назовем *почти допустимо-совершенным*, если в нем нет разделения после слияния путей, и *допустимо-совершенным*, если в нем, кроме того, нет слияния после разделения путей. Граф будем называть *почти допустимо-совершенным* или *допустимо-совершенным*, если в нем есть, соответственно, почти допустимо-совершенное или допустимо-совершенное множество путей.

Достаточное условие допустимой реализации любого допустимого множества пар хостов без циклов и дублирования теперь можно сформулировать в виде следующего утверждения. Оно является аналогом утверждения 7 из [7] для строгой реализации.

Утверждение 7. Почти допустимо-совершенный граф является почти допустимо-хорошим, а допустимо-совершенный граф является допустимо-хорошим. При этом почти допустимо-совершенное множество путей для каждого допустимого множества пар хостов содержит его допустимую реализацию без циклов как подмножество, а допустимо-совершенное множество путей для каждого допустимого множества пар хостов содержит его допустимую реализацию без циклов и без дублирования как подмножество.

Доказательство. Поскольку почти допустимо-совершенное множество путей конечно и в нем нет разделения после слияния, любое его подмножество также конечно и в нем нет разделения после слияния, поэтому, по утверждению 5, это подмножество замкнуто по дугам и не порождает циклов. По определению допустимо-совершенное множество является почти допустимо-совершенным, поэтому любое его подмножество также конечно, замкнуто по дугам и не порождает циклов. Поскольку в допустимо-совершенном множестве путей нет слияния после разделения, в любом его подмножестве также нет слияния после разделения, то, по утверждению 6, это подмножество не порождает дублирование. Для любого

допустимого множества D пар хостов в (почти) допустимо-совершенном множестве P путей можно выбрать подмножество $P(D)$ такое, что $H(P(D)) = D$. Множество $P(D)$ путей допустимо реализует множество D пар хостов без циклов и, если множество P допустимо-совершенное, без дублирования.

□

В [7] была высказана гипотеза о том, что существование наибольшего совершенного множества путей является не только достаточным, но и необходимым условием строгой реализации любого нормального множества пар хостов без циклов и дублирования. Однако в [8] компьютерные эксперименты показали, что это не так. Поэтому существование наибольшего допустимо-совершенного множества путей не необходимо для допустимой реализации любого допустимого множества пар хостов без циклов и дублирования, по крайней мере, для этого случая равноприоритетных хостов.

5. Заключение

В статье рассматривается реализация множества пар разных хостов с помощью путей в графе физических связей и соответствующей настройки коммутаторов, при условии, что определены пары хостов, между которыми можно передавать пакеты с заданным приоритетом. Соответственно рассматриваются допустимые множества пар разных хостов, т.е. удовлетворяющих правилу приоритетов: пакеты, посланные из хоста с данным приоритетом, могут получать только хосты с тем же или бо̀льшим приоритетом. Реализующее множество путей тоже должно быть допустимым, т.е. не должно нарушать правило приоритетов, и не иметь циклов, по которым пакеты будут циркулировать бесконечно и бесконечно размножаться. Кроме того, в ряде случаев для сокращения нагрузки на сеть желательно, чтобы множество не содержало дублирующих путей, т.е. нескольких путей, соединяющих одну и ту же пару хостов. В работе показано, что в отличие от подобной задачи для нестрогой реализации на графе без приоритетов, допустимая реализация не всегда возможна, и тем более, не всегда возможна без дублирования и не всегда возможна без циклов.

Сформулировано и доказано требование к графу, которое достаточно для допустимой реализации любого допустимого множества пар хостов без циклов с возможным дублированием. Это требование формулируется как наличие в графе наибольшего почти допустимо-совершенного множества путей, которое соединяет все допустимые пары хостов и в котором пути не разделяются после слияния. Если требуется отсутствие дублирования, то в достаточном условии запрещается слияние путей после их разделения.

Дальнейшие исследования могут развиваться в двух направлениях. Одно направление – это исследование свойств графа, которые позволяют или не позволяют иметь в нём наибольшее (почти) допустимо-совершенное множество путей. Другое направление – это обобщение идеи приоритетов, когда некоторые веса приписывается не только хостам (в данной статье приоритеты), но и другим коммуникационным элементам: коммутаторам, дугам графа и парам смежных дуг, т.е. правилам коммутатора, с различными целями и по различным правилам.

Список литературы / References

[1]. S. Sezer, S. Scott-Hayward, P.K. Chouhan et al. Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Communications Magazine, vol. 51, no. 7, 2013, pp. 36-43.

[2]. J. López, N. Kushik, N. Yevtushenko, and D. Zeghlache. Analyzing and Validating Virtual Network Requests. In Proc, of the 12th International Conference on Software Technologies (ICSOFT 2017), 2017, pp. 441-446.

[3]. N. Yevtushenko, A. Kossatchev, J. Lopez et al. Test Derivation for the Software Defined Networking Platforms: Novel Fault Models and Test Completeness. In Proc. of the IEEE East-West Design and Test Symposium, 2018, pp 1-5.

[4]. И.Б. Бурдонов, Н.В. Евтушенко, А.С. Косачев. Тестирование правил настройки сетевого коммутатора программно конфигурируемой сети. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 69-88 / I.B. Burdonov, N.V. Yevtushenko, A.S. Kossatchev. Testing switch rules in software defined networks. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 6, 2018, pp. 69-88 (in Russian). DOI: 10.15514/ISPRAS-2018-30(6)-4.

[5]. I. Burdonov, A. Kossatchev, N. Yevtushenko et al. Verifying SDN Data Path Requests. arXiv:1906.03101, 2019.

[6]. Y. Boufkhad, R. De La Paz, L. Linguaglossa et al. Forwarding tables verification through representative header sets. arXiv:1601.07002, 2016.

[7]. I. Burdonov, N. Yevtushenko, and A. Kossatchev. Implementing a virtual network on the SDN data plane. In Proc. of the IEEE East-West Design and Test Symposium, 2020, pp 1-5.

[8]. И.Б. Бурдонов, Е.М. Винарский, Н.В. Евтушенко, А.С. Косачев. Совершенные множества путей в полном графе коммутаторов SDN-сети. Труды ИСП РАН, том 32, вып. 4, 2020 г., стр. 243–258 / I.B. Burdonov, E.M. Vinarskii, N.V. Yevtushenko, A.S. Kossatchev. Perfect sets of paths in the full graph of SDN network switches. Trudy ISP RAN/Proc. ISP RAS, vol. 32, issue 4, 2020. pp. 243–258 (in Russian). DOI: 10.15514/ISPRAS–2020–32(4)–18.

Информация об авторах / Information about authors

Игорь Борисович БУРДОНОВ – доктор физико-математических наук, ведущий научный сотрудник. Научные интересы: формальные спецификации, генерация тестов, технология компиляции, системы реального времени, операционные системы, объектно-ориентированное программирование, сетевые протоколы, процессы разработки программного обеспечения.

Igor Borisovich BURDONOV – Doctor of Physical and Mathematical Sciences, Leading Researcher. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, software development processes.

Нина Владимировна ЕВТУШЕНКО, доктор технических наук, профессор, г.н.с. ИСП РАН, до 1991 года работала научным сотрудником в Сибирском физико-техническом институте. С 1991 г. работала в ТГУ профессором, зав. кафедрой, зав. лабораторией по компьютерным наукам. Её исследовательские интересы включают формальные методы, теорию автоматов, распределенные системы, протоколы и тестирование программного обеспечения.

Nina Vladimirovna YEVTUSHENKO, Doctor of Technical Sciences, Professor, a Leading Researcher of ISP RAS, worked at the Siberian Scientific Institute of Physics and Technology as a researcher up to 1991. In 1991, she joined Tomsk State University as a professor and then worked as the chair head and the head of Computer Science laboratory. Her research interests include formal methods, automata theory, distributed systems, protocol and software testing.

Александр Сергеевич КОСАЧЕВ – кандидат физико-математических наук, ведущий научный сотрудник. Научные интересы: формальные спецификации, генерация тестов, технология компиляции, системы реального времени, операционные системы, объектно-ориентированное программирование, сетевые протоколы, процессы разработки программного обеспечения.

Alexander Sergeevitch KOSSATCHEV – Candidate of Physical and Mathematical Sciences, Leading Researcher. Research interests: formal specifications, test generation, compilation technology, real-time systems, operating systems, object-oriented programming, network protocols, software development processes.

DOI: 10.15514/ISPRAS–2021–33(1)–10

Выявление неисправностей в группах мобильных роботов с использованием скользящих наблюдателей

[1] *О.Ю. Сергиенко, ORCID: 0000-0003-4270-6872 <srgnk@uabc.edu.mx>*
[2] *А.Н. Жирабок, ORCID: 0000-0001-5927-7117 <zhirabok@mail.ru>*

[1] *Автономный университет Нижней Калифорнии (UABC),*
Мексика, 21100, Нижняя Калифорния, Энсенада
[2] *Дальневосточный федеральный университет,*
Россия, 690090, Владивосток, ул. Суханова, 8

Аннотация. Работа посвящена изучению главных направлений в усовершенствованных вычислительных и информационных технологиях для эффективного решения задачи идентификации дефектов в группе мобильных роботов в присутствии возмущений. Для решения этой задачи привлекаются скользящие наблюдатели. Это облегчает реализацию концепции «умных» элементов в рассматриваемой группе роботов в процессе обобщенного управления ими: «умных» датчиков окружающей среды, связь, обработку информации и хранение данных сканирования. Предложенный новый подход к построению скользящих наблюдателей базируется на построении модели пониженного порядка исходной системы. Это позволяет уменьшить сложность реализации скользящих наблюдателей и ослабить ограничения, накалываемые на систему.

Ключевые слова: группа мобильных роботов; диагностирование; идентификация дефектов; скользящие наблюдатели.

Для цитирования: Сергиенко О.Ю., Жирабок А.Н. Выявление неисправностей в группах мобильных роботов с использованием скользящих наблюдателей. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 137-150. DOI: 10.15514/ISPRAS–2021–33(1)–10

Благодарности. Работа частично поддержана Российским научным фондом 16-19-00046-Р и Автономным университетом Нижней Калифорнии, Мексика (проект № 111/2071, 2018-2020).

Fault Identification in Mobile Robot groups using Sliding Mode Observers

[1] *O. Sergiyenko, ORCID: 0000-0003-4270-6872 <srgnk@uabc.edu.mx>*
[2] *A. Zhirabok, ORCID: 0000-0001-5927-7117 <zhirabok@mail.ru>*
[1] *Autonomous University of Baja California (UABC),*
Ensenada, Mexico, 21100
[2] *Far Eastern Federal University,*
8, Sukhanova, Vladivostok, 690091 Russia

Abstract. The paper studies the emerging trends in advanced computing and information technology for efficient solutions of fault identification problem for mobile robot groups under the unmatched disturbances. The sliding mode observers are considered for mentioned problem solution. It facilitates the concept of Smart everything inside the considered robotic group during its generalized control: smart surrounding sensing, communication, processing, and scanned data storing. The suggested novel approach to sliding mode observer

design is based on obtaining the reduced order model of the initial system. This allows reduce the complexity of sliding mode observer and relax restrictions imposed on the initial system.

Keywords: group of mobile robots; diagnosis; identification of faults; sliding mode observers

For citation: Sergiyenko O., Zhirabok A. Fault Identification in Mobile Robot groups using Sliding Mode Observers. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 137-150 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–10.

Acknowledgments. The work is partially supported by Russian Scientific Foundation 16-19-00046-P and by the Autonomous University of Baja California, Mexico (project No. 111/2071, in 2018-2020).

1. Введение

В настоящее время мобильные роботы (МР) являются одной из притягательных сфер применения современной теории управления [1-3]. Важнейший вызов в этой области – зависимость управляемого объекта (как отдельного робота, так и их групп) от надежности их компонент. Очевидно, что при функционировании отдельных МР и их групп актуальными становятся эффективные методы обнаружения и идентификации дефектов. Важность этой задачи подтверждается многочисленными статьями [4-8], опубликованными в этой области последние годы, где рассматриваются различные подходы для достижения этой цели.

Очень показательный пример такой группы роботов – несколько автоматических устройств, объединенных в группировку для решения глобальной задачи автоматизации сельскохозяйственных процессов. Такая группировка состоит из подгруппы летающих роботов [9] и подгруппы колесных наземных мобильных роботов [10], а также подсистем для информационного обмена [11] и подсистемы исполнительных приводов для процесса управления. Такая комплексная система представляет классический объект для многоцелевой оптимизации в нестационарной окружающей обстановке.

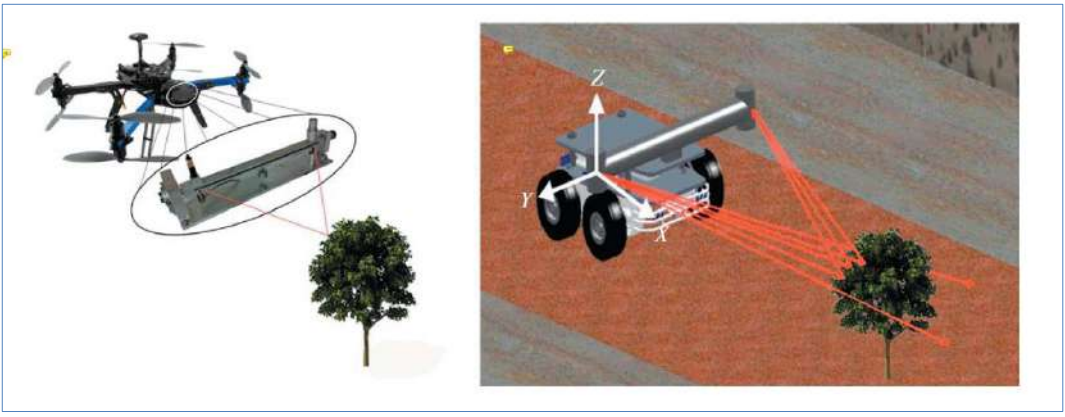


Рис.1. Использование лазерной сканирующей системы для измерения жизнеспособности насаждений: слева летающий мобильный робот; справа колесный мобильный робот

Fig. 1. Use of laser scanning TVS to measure the vegetation vitality by normalized differenced vegetation index: using flying mobile robot (left); b) using wheeled mobile robot (right)

Такая сложная группа состоит из летающих роботов (дронов) [9] и группы колесных МР [10, 11]. Подробности независимого использования летающих и наземных роботов приведены в наших предыдущих публикациях. Оба случая представлены на рис. 1, каждый из них имеет определенные преимущества и недостатки в смысле эффективности получения информации, ее обработки и хранения.

Работа [9] рассматривает использование дронов с лазерным сканером [10] для пролета над посадками, например, оливковыми садами, виноградниками (или другими видами зеленых насаждений). В экспериментах мы установили, что форма сигналов, полученных от сканера,

сильно зависит от интенсивности зеленого цвета (см. рис. 2). То есть имеется возможность оценить степень влажности почвы и использовать это для управления ирригацией.

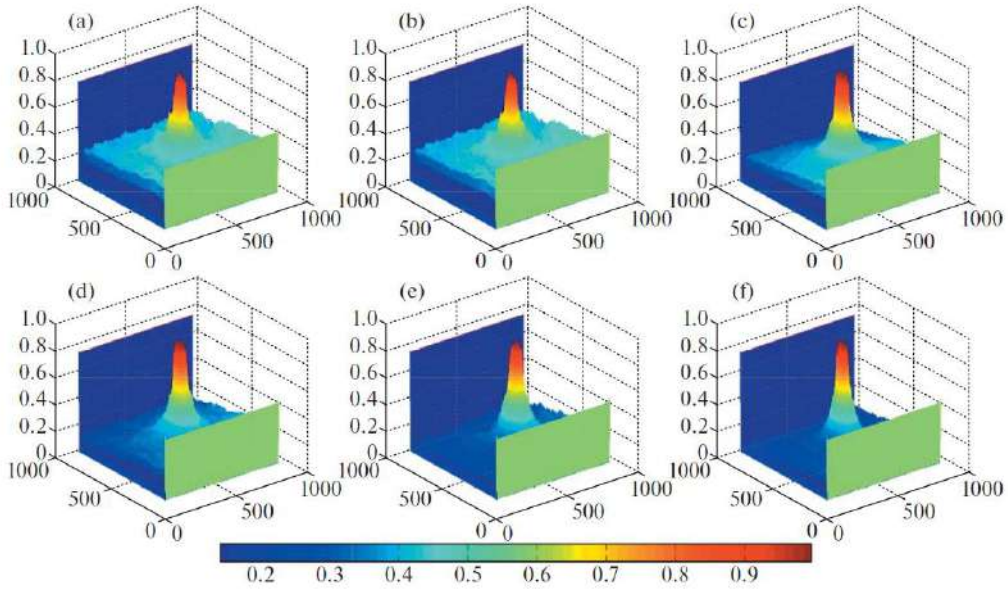


Fig. 2. Относительная шкала интенсивности лазерного отражения от подсвеченной поверхности с различными уровнями растительной жизнеспособности (зелёная пигментация листа)
Fig. 2. Relative scale of Reflectance of laser intensity on highlighted surface with different vegetation vitality (leaf green pigmentation)

Однако существенная нестабильность расположения летающих роботов делает затруднительным направление лазерного луча на каждый отдельный листок. Существенно проще собрать эти данные от наземных и медленно перемещающихся МР (рис. 1b). Такие данные будут более точными и лучше защищенными от искажений. Очевидно, однако, что наземным роботам потребуется больше времени на выполнение задания, и в некоторых случаях они не смогут выйти за определенные границы. Следовательно, логично дополнить их летающей группой, которая может обозреть большое пространство за то же самое время, и теоретически не будет иметь препятствий для получения необходимой информации.

Проблемы ирригации будут решены лучше, если во внимание принимаются объединенные данные, полученные от обеих групп. Ясно, что в этом случае очень желательно иметь инструмент, выявляющий, какая часть полученной информации имеет меньше ошибок. Также важно отметить, что значительные ошибки в расположении дрона из-за больших вибраций и ветровых нагрузок будут на несколько порядков больше, чем шумы наземной части группы.

В упомянутых выше МР с системой технического зрения на основе лазерного сканирования интересным объектом для обнаружения и коррекции ошибок является электрическая система, предназначенная для коррекции пространственного положения лазерного луча. Ключевым проблемой в этой задаче [12, 13] является надлежащая компенсация эффектов нелинейного трения на конечной стадии остановки луча в желаемой точке, когда модель сухого трения не может учесть квантовый скачок между динамическим и статическим трением, и, следовательно, система управления не может компенсировать этот физический эффект для верного позиционирования сканирующего инструмента.

В [12, 13] приведен подход к компенсации этих эффектов, проверенный статистически. Очевидно, однако, что подходы к управлению, использующие очень примитивные модели

трения [14], никогда не обеспечат эффективную компенсацию трения, применимую к широкому классу приборов без использования очень строгих ограничений на условия среды вокруг трибологического узла. Конечно, такие модели будут перспективными при использовании более адекватного подхода к составляющим силы трения [15]. Однако улучшение таких моделей вызовет нежелательное увеличение числа датчиков и сложность системы, что приведет к понижению надежности и росту обрабатываемого объема данных.

Таким образом, в этом случае более продуктивным решением проблемы видится проектирование устройств, способных обнаруживать дефекты во всех элементах, входящих в систему, и затем искусственно уменьшать объем получаемой информации путем преднамеренного исключения данных, полученных от отказавших элементов. Разумеется, в этом случае исходное число данных должно быть избыточным относительно экспериментально (или модельно) установленного процента отказов.

Работа посвящена проблеме диагностирования дефектов в МР системах. Процесс диагностирования включает в себя генерацию невязки как результата рассогласования между поведением системы и ее эталонной модели, сопровождаемую принятием решения за счет анализа невязки. Проблема диагностирования интенсивно исследуется последние 30 лет, см., например, [16-18]. Было предложено несколько методов диагностирования: диагностические наблюдатели, соотношения паритета, идентификация. Существует много методов идентификации, один из них базируется на так называемых скользящих наблюдателя и использует особенности скользящего режима [19].

2. Методы идентификации дефектов

Скользящие наблюдатели используются для идентификации дефектов в линейных [20-24], нелинейных [25-27] и сингулярных системах [28], для обеспечения отказоустойчивого управления [29-31], в ряде практических приложений [32-34].

Для обеспечения отказоустойчивого управления работа [35] использует стратегию управления на основе скользящих наблюдателей, которые обеспечивают идентификацию дефектных элементов. Как только эти элементы идентифицированы, закон управления изменяется определенным образом, и система остается работоспособной. Подход, предложенный в [31, 36], предполагает каскадную схему реконфигурации и технику распределения управлений, что позволяет избежать перестройки контроллера.

Отметим, что скользящие наблюдатели в [37] и аналогичных работах строятся на основе исходной системы. Как результат, скользящие наблюдатели имеют полный порядок, в [26] строятся два скользящих наблюдателя. Кроме того, при построении таких наблюдателей на исходную систему накладывается ряд ограничений, в частности, в [37] и аналогичных работах требуется, чтобы система была минимально фазовой.

Новизна настоящей работы состоит в том, что скользящий наблюдатель строится не по исходной системе, а по ее редуцированной модели. В результате размерность наблюдателя получается меньшей, чем у исходной системы. Отметим, что редуцированная модель может не иметь некоторых особенностей исходной системы, которые препятствуют построению скользящего наблюдателя. Таким образом, ограничения, накладываемые на исходную систему, могут быть ослаблены. Это позволяет расширить класс систем, для которых скользящие наблюдатели могут быть построены.

В настоящей работе метод на основе скользящих наблюдателей используется для решения проблемы идентификации дефектов в МР при наличии возмущений. Для построения скользящего наблюдателя мы использует редуцированную модель исходной системы, не чувствительную к возмущениям. Это позволяет уменьшить сложность скользящего наблюдателя по сравнению с работами [21, 37, 26], где строятся наблюдатели полного порядка. Кроме того, ослабляются ограничения, накладываемые в [21, 37, 26] на исходную систему.

3. Формулирование проблемы

Рассмотрим МР, описываемый нелинейной моделью (детализированная версия системы (2.6) работы [1], p. 42):

$$\dot{x}(t) = Fx(t) + Gu(t) + Dd(t) + L\rho(t) + C\Psi\big(x(t), u(t)\big),$$
$$y(t) = Hx(T), \tag{1}$$

где $x(t) \in R^n, u(t) \in R^m, y(t) \in R^l$ – векторы состояния, управления и выхода; F, G, H, C, D и L – известные постоянные матрицы, $d(t) \in R$ – функция, описывающая дефекты: при их отсутствии $d(t) = 0$ при появлении дефекта $d(t)$ становится неизвестной функцией времени, $\rho(t) \in R^P$ – неизвестная функция времени, описывающая действующие на систему возмущения; $\Psi(x, u)$ – нелинейная составляющая:

$$\Psi(x, u) = \begin{pmatrix} \varphi_1(A_1 x, u) \\ \dots \\ \varphi_s(A_s x, u) \end{pmatrix},$$

$A_1, \dots, A_s$ – постоянные матрицы, $\varphi_1, \dots, \varphi_s$ – нелинейные функции. Предполагается, что функция $\Psi(x, u)$ удовлетворяет условию Липшица по аргументу x:

$$\big\|C\Psi(x, u) - \Psi(x', u)\big\| \le N\big\|x - x'\big\|,$$

Где $N \ge 0$ – некоторая константа.

Для ряда практических значимых функций это условие не выполняется, однако выполняется обобщенное условию Липшица:

$$\big\|C\Psi(x, u) - \Psi\big(x^{'}, u\big)\big\| \le N\big\|x - x^{'}\big\| + M, \tag{2}$$

$N, M \ge 0$ – некоторые константы.

Напомним, что в [27] предполагается, что система (1) удовлетворяет следующим условиям:

1) $rank(H[LD]) = rank([LD]),$,
2) все инвариантные нули тройки $(F, [LD], H)$ лежат в левой полуплоскости.

Известно, что при этих предположениях система (1) может быть преобразована в композицию двух подсистем, имеющих следующие особенности: функции $d(t)$ и $\rho(t)$ входят только в первую подсистему, вектор выхода $y(t)$ зависит только от вектора состояния этой подсистемы и вторая подсистема устойчива. В настоящей работе задача идентификации решается без этих условий.

4. Построение редуцированной модели

Решение проблемы базируется на редуцированной модели системы (1), в общем случае описываемой следующими уравнениями:

$$\dot{x}_*(t) = F_* x_*(t) + G_* u(t) + J_* y(t) + D_* d(t) + L_* \rho(t) + C_* \Psi\big(x_*(t), y(t), u(t)\big),$$
$$y_*(t) = H_* x_*(T), \tag{3}$$

где $x_* \in R^k$ – вектор состояния, F_*, G_*, J_*, D_*, L_* и H_* – постоянные матрицы,

$$C_* \Psi(x_*, y, u) = \begin{pmatrix} \varphi_{i_1}\big(A_{*1i_1} x_* + A_{*2i_1} y, u\big) \\ \dots \\ \varphi_{i_k}\big(A_{*1i_k} x_* + A_{*2i_k} y, u\big) \end{pmatrix}, \tag{4}$$

$A_{*1i_1}, A_{*2i_1}, \dots, A_{*1i_k}$, A_{*2i_k}– матрицы, подлежащие определению.

Предполагается, что $x_*(t) = \Phi x(t)$ и $y_*(t) = R_* y(t)$ для некоторых матриц Φ и R_* при $d(t) = 0$ и $\rho(t) = 0$. Из [38, 39] известно, что эти матрицы удовлетворяют условиям

$$\Phi F = F_* \Phi + J_* H,\ R_* H_* = H\Phi,\ G_* = \Phi G,\ D_* = \Phi D,\ L_* = \Phi L,$$
$$C_* = \Phi C, \tag{5}$$
$$A_{*i} = (A_{*1i}\ A_{*2i})\binom{\Phi}{H}, i = i_1, \dots, i_k.$$

Рассмотрим метод построения системы (3), не чувствительной к возмущениям, которая будет использоваться при построении скользящего наблюдателя. Матрицы F_* и H_* ищутся в канонической форме следующего вида:

$$F_* = \begin{pmatrix} 0 & 1 & 0 & & 0 \\ 0 & \alpha_1 & 1 & \cdots & 0 \\ 0 & 0 & 0 & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & \alpha_{k-1} & 0 & \cdots & 0 \end{pmatrix}, \quad (6)$$

$$H_* = (1 \;\; 0 \;\; 0 \;\; \ldots \;\; 0),$$

где $\alpha_1, \ldots, \alpha_{k-1}$ – коэффициенты обратной связи, обеспечивающие устойчивость части модели, описываемой вектором $x_{**} = (x_{*2}, \ldots, x_{*k})^T$.

Отметим, что устойчивость этой подсистемы гарантируется в [27] для произвольного k требованием "все инвариантные нули тройки $(F, [LD], H)$ лежат в левой полуплоскости". Наша цель – ослабить это ограничение для практически важного случая $k \le 3$.

Используя F_* и H_* в (6), получим из (5) уравнения для строк матрицы Φ и J_*:

$$\Phi_1 = R_* H, \Phi_1 F = \Phi_2 + J_{*1} H,$$
$$\Phi_I F = \Phi_{I-1} + \alpha_{I-1} \Phi_2 + J_{*i} H, \qquad i = 2, \ldots, k-1, \quad (7)$$
$$\Phi_k F = \alpha_{k-1} \Phi_2 + J_{*k} H,$$

где Φ_I и J_{*i} – i-е строки матриц Φ и J_*, $i = 1, \ldots, k$, k – размерность модели (3).

Рассмотрим случай $k = 3$.. Уравнения (7) можно привести к единственному уравнению путем ряда преобразований:

$$R_* HF = \Phi_2 + J_{*1} H,$$
$$R_* HF^2 = \Phi_2 F + J_{*1} HF = J_{*1} HF + J_{*2} H + \alpha_1 (R_* HF + J_{*1} H) =$$
$$= J_{*1} HF + J_{*2} H + \alpha_1 \Phi_2 = (J_{*1} + \alpha_1 R_*)\, HF + (J_{*2} - \alpha_1 J_{*1}) H.$$

Продолжая по аналогии, получим

$$R_* HF^3 = (J_{*1} + \alpha_1 R_*) HF^2 + (J_{*2} - \alpha_1 J_{*1} + \alpha_2 R_*) HF + (J_{*3} + \alpha_2 J_{*1}) H =$$
$$= J'_{*1} HF^2 + J'_{*2} HF + J'_{*3} H, \quad (8)$$

где

$$J'_{*1} = J_{*1} + \alpha_1 R_*,$$
$$J'_{*2} = J_{*2} - \alpha_1 J_{*1} + \alpha_2 R_*,$$
$$J'_{*3} = J_{*3} - \alpha_2 J_{*1}. \quad (9)$$

Известно [38, 39], что условие нечувствительности модели (3) к возмущениям имеет вид $\Phi L = 0$. Можно показать, что

$$\Phi_1 = R_* H, \Phi_2 = \Phi_1 F - J_{*1} H = R_* HF - J_{*1} H.$$
$$\Phi_3 = \Phi_2 F - J_{*2} H - \alpha_1 \Phi_2 = R_* HF^2 - J_{*1} HF - \alpha_1 \Phi_2.$$

Отсюда получаем, что условие $\Phi L = 0$ может быть записано в виде [39, 39]

$$(R_* - J_{*1} - J_{*2} - J_{*3}) L^{(3)} = 0, \quad (10)$$

где

$$L^{(3)} = \begin{pmatrix} HL & HFL & HF^2 L \\ 0 & HL & HFL \\ 0 & 0 & HL \\ 0 & 0 & 0 \end{pmatrix}.$$

Как и в [27], функция $d(t)$ должна содержаться в первой компоненте вектора x_*. Это эквивалентно условию $\Phi_I D = 0, i = 2,3$, которое можно представить в виде 39, 39]

$$(R_* - J_{*1} - J_{*2} - J_{*3}) D^{(3)} = 0, \quad (11)$$

где

$$D^{(3)} = \begin{pmatrix} 0 & HFD & HF^2D \\ 0 & HD & HFD \\ 0 & 0 & HD \\ 0 & 0 & 0 \end{pmatrix}.$$

Как правило, на практике $k \leq 3$. Отметим, что при $k > 3$ матрицы $L^{(k)}$ и $D^{(k)}$ становятся более сложными, в частности

$$L^{(4)} = \begin{pmatrix} HL & HFL & HF^2L & HF^3L \\ 0 & HL & HFL & HF^2L \\ 0 & 0 & HL & HFL + \alpha_1 HL \\ 0 & 0 & 0 & HL \\ 0 & 0 & 0 & 0 \end{pmatrix} \text{ и}$$

$$D^{(4)} = \begin{pmatrix} 0 & HFD & HF^2D & HF^3D \\ 0 & HD & HFD & HF^2D \\ 0 & 0 & HD & HFD + \alpha_1 HD \\ 0 & 0 & 0 & HD \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Из (10) и (11) следует

$$(R_* - J_{*1} - \cdots - J_{*k})\left(L^{(k)} - B^{(k)}\right) = 0, \tag{12}$$

$k = 1,2,3..$ Для построения модели (3) минимальной размерности необходимо найти минимальное k, для которого (12) имеет решение; для конкретности, рассмотрим $k = 3$. Затем находится решение уравнения (12) в виде $(R_* - J_{*1} - J_{*2} - J_{*3})$ и решение уравнения (8) в виде $(J'_{*1}\, J'_{*2}\, J'_{*3})$ для матрицы R_*, найденной из (12). Для выяснения совместимости этих решений необходимо проверить разрешимость уравнения (9) для некоторых α_1 и α_2. Если (9) разрешимо для α_1 и α_2, удовлетворяющих условию устойчивости, то линейная часть модели построена. Отметим, что если $k = 1$, то $J'_{*1} = J_{*1}$ и решения совместимы.

Для построения нелинейной части найдем из (7) строки матрицы Ф, примем $C_* = \Phi C$ и проверим условие

$$rank \begin{pmatrix} \Phi \\ H \end{pmatrix} = rank \begin{pmatrix} \Phi \\ H \\ A_i \end{pmatrix}, i = i_1, \dots, i_k. \tag{13}$$

Если оно выполняется, принять $G_* = \Phi G$ и $D_* = \Phi D$; матрицы A_{*1i_1} и A_{*2i_1}, $i = i_1, \dots, i_k$ из нелинейной функции (4) определяются из (5). Если (13) не выполняется или условия (9) не разрешимы, находится другое решения уравнения (12) с прежней или большей величиной k.

Как итог, модель (3) принимает вид

$$\begin{gathered} \dot{x}_{*1}(t) = x_{*2}(t) + G_{*1}u(t) + J_{*1}y(t) + C_{*1}\Psi\big(x_*(t), y(t), u(t)\big) + ad(t), \\ \dot{x}_{**}(t) = F_{**}x_{**}(t) + G_{**}u(t) + J_{**}y(t) + C_{**}\Psi\big(x_*(t), y(t), u(t)\big), \\ y_*(t) = x_{*1}(T), \end{gathered} \tag{14}$$

где $a = \Phi_1 D \neq 0, x_{**} = (x_{*2}, \dots, x_{*K})^T$, F_{**}, G_{**}, J_{**} и C_{**} обозначают подматрицы матриц F_*, G_*, J_* и C_*, соответствующих вектору x_{**}, матрица F_{**} устойчива.

5. Построение скользящего наблюдателя

Скользящий наблюдатель ищется в виде

$$\begin{gathered} \hat{\dot{x}}_{*1}(t) = \hat{x}_{*2}(t) + G_{*1}u(t) + J_{*1}y(t) + C_{*1}\Psi\big(\hat{x}_*(t), y(t), u(t)\big) - be_y(t) + v(t), \\ \hat{\dot{x}}_{**}(t) = F_{**}\hat{x}_{**}(t) + G_{**}u(t) + J_{**}y(t) + C_{**}\Psi\big(\hat{x}_*(t), y(t), u(t)\big), \\ \hat{y}_*(t) = \hat{x}_{*1}(t), \end{gathered} \tag{15}$$

где $b > 0$,

$$v(t) = \begin{cases} -g|a|\frac{e_y(t)}{\|e_y\|}, \text{если } e_y(t) \neq 0, \\ 0 \text{ в противном случае,} \end{cases} \quad (16)$$

$$e_y(t) = \hat{y}_*(t) - y_*(t) = \hat{y}_*(t) - R_* y(t).$$

Используя (14) и (15), запишем уравнения для ошибок $e_1(t)$ и $e_2(t) = \hat{x}_{**}(t) - x_{**}(t)$:

$$\dot{e}_1(t) = -be_1 + Qe_2(t) - ad(t) + v(t) + \triangle \Psi_1(t),$$
$$\dot{e}_2(t) = F_{**}e_2(t) + \triangle \Psi_2(t), \quad (17)$$

где $Q = (1\ 0\ \ldots\ 0)$,

$\triangle \Psi_1(t) = C_{*1}(\Psi(\hat{x}_*(t), y(t), u(t)) - \Psi(x_*(t), y(t), u(t)))$,

$\triangle \Psi_2(t) = C_{**}(\Psi(\hat{x}_*(t), y(t), u(t)) - \Psi(x_*(t), y(t), u(t)))$.

Поскольку функции $C_{*1}\Psi(x_*, y, u)$ и $C_{**}\Psi(x_*, y, u)$ удовлетворяют условию Липшица (2) относительно x, то функции $C_{*1}\Psi(\hat{x}_*, y, u)$ и $C_{**}\Psi(\hat{x}_*, y, u)$ удовлетворяют этому условию относительно $\hat{x}_*$ также, и

$$\|\triangle \Psi_1(t)\| \leq N_{*1}\|e(t)\| + M_{*1} \leq N_{*1}\|e_1(t)\| + N_{*1}\|e_2(t)\| + M_{*1},$$
$$\|\triangle \Psi_1(t)\| \leq N_{*2}\|e_1(t)\| + N_{*2}\|e_2(t)\| \quad (18)$$

для некоторых чисел N_{*1}, M_{*1} и N_{*2}. Отметим, что ненулевая величина M_{*1} используется для обеспечения существования скользящего режима в наблюдателе (15) (смотрите условие (19)). С другой стороны, известно [37], что скользящее движение обеспечивает равенство $e(t) = 0$, т.е. $\hat{x}_*(t) = x_*(t)$, и здесь можно принять $M_{*1} = 0$.

Поскольку матрица F_{**} устойчива, то для произвольной симметрической положительно определенной матрицы W существует симметрическая положительно определенная матрица P_2, такая, что

$$F_{**}^{\mathrm{T}}P_2 + P_2F_{**} = -W.$$

Теорема. Если скаляр g удовлетворяет условию

$$g > \|d(t)\| + \frac{M_{*1}}{|a|}, \quad (19)$$

то существует скаляр b, такой, что скользящее движение системы (17) асимптотически устойчиво.

Доказательство теоремы основано на анализе функции Ляпунова

$$V(t) = e_1^{\mathrm{T}}(t)e_1(t) + e_2^{\mathrm{T}}(t)Pe_2(t)$$

и доказательстве того, что ее производная $\dot{V}(t) < 0$. Из этого следует, что скользящее движение системы (17) асимптотически устойчиво, при этом $e_1(t) \to 0$ and $e_2(t) \to 0$.

Известно [37], что скользящее движение обеспечивает равенства $\dot{e}(t) = 0$ и $e(t) = 0$, а поскольку матрица F_{**} устойчива, равенство (17) влечет $0 = v(t) - ad(t) + \triangle \Psi_1(t)$. Так как согласно замечанию после (18) справедливо неравенство $\|\triangle \Psi_1(t)\| \leq N_{*1}\|e(t)\| = 0$, то функция $d(t)$ может быть оценена в виде

$$d(t) = -g|a|\frac{e_y(t)}{a(\|e_y(t)\|)+\delta},$$

где δ – малое положительное число. Отметим, что правая часть этого уравнения зависит только от ошибки $e_y(t) = y_*(t) - R_*y(t)$.

6. Шумы измерений

В случае, когда присутствуют шумы измерений, основной результат работы остается прежним, но требования к величине g становятся более жесткими для обеспечения скользящего режима. Действительно, пусть $y(t) = Hx(t) + \rho_s(t)$, где $\rho_s(t)$ – ограниченная функция времени, описывающая шумы измерений и $\|\rho_s(t)\| \leq S_*$.

Чтобы принять эти шумы во внимание, модель (14) корректируется следующим образом:

$$\dot{x}_{*1}(t) = x_{*2}(t) + G_{*1}u(t) + J_{*1}Hx(t) + C_{*1}\Psi\big(x_*(t), Hx(t), u(t)\big) + ad(t),$$
$$\dot{x}_{**}(t) = F_{**}x_{**}(t) + G_{**}u(t) + J_{**}Hx(t) + C_{**}\Psi\big(x_*(t), Hx(t), u(t)\big),$$
$$y_*(t) = x_{*1}(t).$$

Тогда уравнение (17) для переменной $e_1(t)$ дополняется слагаемым $J_{*1}\rho_s(t)$. Предполагается для простоты, что шумами измерений во второй подсистеме можно пренебречь; такое предположение справедливо, например, когда шумы во второй подсистеме существенно (на несколько порядков величины) меньше:

$$\dot{e}_1(t) = -be_1 + Qe_2(t) - ad(t) + J_{*1}\big(y(t) - Hx(t)\big) + \triangle\,\Psi_1(t) + v(t) =$$
$$-be_1 + Qe_2(t) - ad(t) + J_{*1}\rho_s(t) + \triangle\,\Psi_1(t) + v(t) \qquad (20)$$

при этом

$$\|\triangle\,\Psi_1(t)\| \le N_{*1}\|e(t)\| + M_{*1} \le N_{*1}\|e_1(t)\| + N_{*1}\|e_2(t)\| + M_{*1} + S_{**},$$

где

$$S_{**} = S_* \left\|\frac{\partial C_*\Psi(x_*, y, u)}{\partial y}\right\|.$$

Тогда можно показать, что $V(t) < 0$, если

$$g > \|d(t)\| + \frac{M_{*1} + \|J_{*1}\|S_* + S_{**}}{|a|}.$$

что гарантирует существование скользящего режима. Поскольку шумы измерений входят в уравнение (20), оценка функции $d(t)$ производится с погрешность, не превышающей $\|J_{*1}\|S_* + S_{**}$.

7. Практический пример

Этот пример привлекает разработанные выше теоретические результаты для решения рассмотренной в разделе 1 проблемы неточного позиционирования лазерного луча исполнительным приводом, детально описанной в [9, 12, 13]. Рассмотрим модель электропривода согласно [31]:

$$\dot{x}_1(t) = \frac{1}{i_p}x_2(t),$$
$$\dot{x}_2(t) = \frac{K_M}{J_H}x_3(t) - \frac{K_d}{J_H}sign\big(x_2(t)\big) + \rho(t), \qquad (21)$$
$$\dot{x}_3(t) = -\frac{K_\omega}{L_m}x_2(t) - \frac{R_m}{L_m}x_3(t) + \frac{K_u}{L_m}u(t) + d(t),$$

где $x_1(t)$ – угол поворота выходного вала редуктора; $x_2(t)$ – скорость вращения ротора электродвигателя; $x_3(t)$ – ток якоря; i_p – передаточное отношение редуктора; J_H – номинальный момент инерции вала электродвигателя и вращающихся частей редуктора; K_ω и K_M – коэффициенты противо-э.д.с. (электродвижущая сила) и крутящего момента; K_d – момент сухого трения на валу электродвигателя; R_m и L_m – активное сопротивление и индуктивность цепи якоря; K_u – коэффициент усиления.

Предполагается, что дефект $d(t) = -\big(\tilde{R}_m/L_m\big)x_3(t)$ соответствует отклонению $\tilde{R}$ активного ≠внешним нагрузочным моментом $\tilde{M}(t)$, приложенным к валу электродвигателя.

Обозначим

$$k_1 = \frac{1}{i_p}, k_2 = \frac{K_M}{J_H}, k_3 = -\frac{K_\omega}{L_m}, k_4 = -\frac{R_m}{L_m}, k_5 = \frac{K_u}{L_m}.$$

Предполагая, что измеряются переменные $x_1(t)$ и $x_1(t)$, получаем матрицы, описывающие электропривод:

$$F = \begin{pmatrix} 0 & k_1 & 0 \\ 0 & 0 & k_2 \\ 0 & k_3 & k_4 \end{pmatrix}, G = \begin{pmatrix} 0 \\ 0 \\ k_5 \end{pmatrix}, H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, D = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, L = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix},$$
$$\Psi(x, u) = sign(Ax), A = (0\ \ 1\ \ 0).$$

Отметим, что $rank(H[LD) = 1] \neq rank([LD]) = 2$, следовательно, подход, предложенный в [21], здесь не может быть использован.

Построим редуцированную модель, не чувствительную к возмущениям. Примем $k = 1$ и найдем матрицы $V^{(1)}, L^{(1)}$ и $D^{(1)}$:

$$V^{(1)} = \begin{pmatrix} 0 & k_1 & 0 \\ 0 & k_3 & k_4 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad L^{(1)} = D^{(1)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Ясно, что произвольная строка $(R_* - J_*)$ удовлетворяет уравнению (12). Поскольку $rank\left(V^{(1)}\right) = 3 < 2(1+1) = 4$, уравнение (8) имеет решение с матрицами $R_* = (k_3 - k_1)$ и $J'_* = J_* = (0 - k_1 k_4)$, что в результате дает $\Phi = (k_3 \quad 0 \quad -k_1)$, $C_* = 0$, и $G_* = -k_1 k_5$. В результате модель (14) становится линейной и имеет вид

$$\dot{x}_* = k_1 k_4 y_2(t) - k_1 k_5 u(t) - k_1 d(t),$$
$$y_{*2}(t) = x_*(t), \tag{22}$$

где $x_*(t) = k_3 x_1(t) - k_1 x_3(t), a = \Phi D = -k_1$.

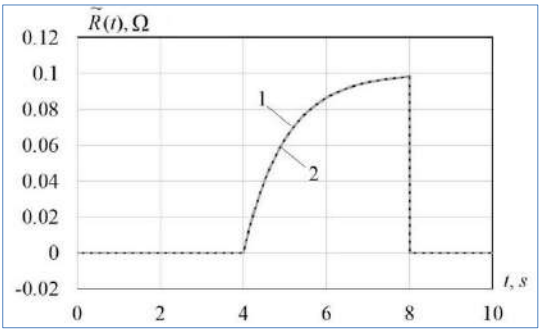


Рис. 3. Поведение функции $\tilde{R}$ и ее оценка
Fig. 3. Behavior of the function $\tilde{R}$ and its estimation

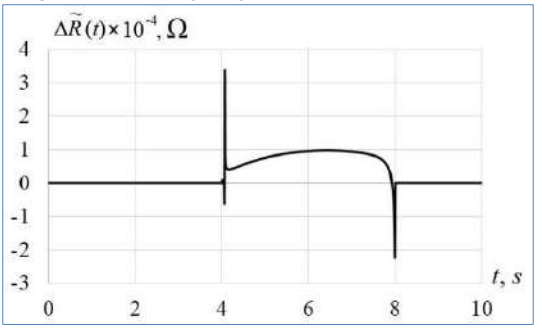


Рис. 4. Поведение ошибки оценивания
Fig. 4. Behavior of the fault estimation error $\hat{\tilde{R}} - \tilde{R}$

Скользящий наблюдатель описывается уравнениями

$$\hat{\dot{x}}_*(t) = -k_1 k_4 y_2(t) - k_1 k_5 u(t) - b e_y(t) + v(t),$$
$$\hat{y}_{*2}(t) = \hat{x}_*(t), \tag{23}$$

где $b > 0$,

$$e_{y2}(t) = \hat{y}_*(t) - \left(k_3 y_1(t) - k_1 y_2(t)\right),$$

разрывная переменная $v(t)$ дается выражением (16). Так как $a = -k_1 < 0$, функция $d(t)$ оценивается в виде

$$\hat{d}(t) = g\frac{e_y(t)}{|e_y(t)|+\delta}. \quad (24)$$

Для моделирования рассмотрим систему (21) и наблюдатель (23) и примем $J_H = 0.0001Kgm^2$, $K_\omega = 0.02Vs$, $K_U = 100$, $R_m = 0.4\Omega$, $L_m = 0.004H$, $K_M = 0.02\,Nm/A$, $i_p = 100$. Для обеспечения качественного переходного процесса используется PID регулятор. Управление $u(t)$ выбирается таким, чтобы заданное величина угла поворота выходного вала редуктора подчинялась закону $x_1^{ref}(t) = \sin(t)$.

Возмущение $\rho(t)$ интерпретируется как внешний нагрузочный момент и задается функцией $\widetilde{M}(t) = 0.10\sin(0.8t)\,Nm$ на интервале от 1 до 10 ms. Дефект $d(t)$ вызван отклонением $\tilde{R}(t) = 0.2\sin(0.2t - 0.8)$ активного сопротивления R_m на интервале от 4 до 8 ms.

Результаты моделирования с $g = 100$ и $\delta = 10^{-5}$ в (24) приведены на рис. 3 и 4, символом 1 обозначена функция $\tilde{R}(t)$, символом 2 – ее оценка согласно (24). Рис. 4 показывает поведение ошибки оценивания. Поскольку на интервале от 0 до 4 ms и от 8 до 10 ms оценка равна нулю, наблюдатель (23) нечувствителен к возмущению $\rho(t)$.

8. Заключение

В статье решалась задача идентификации дефектов в системах, описываемых нелинейными моделями в присутствии возмущений, на основе скользящих наблюдателей с применением к группе различных типов мобильных роботов. Предложенная модификация этого метода состоит в использовании редуцированной (имеющей меньшую размерность) модели исходной системы, представленной уравнениями (3), (14) и (22), которые получены на основе канонической формы (6) матриц F_* и H_*. Это, как следствие, дает такие преимущества для рассматриваемой технической системы группы МР: уменьшение сложности скользящих наблюдателей и ослабление условий, налагаемых на исходную систему, по сравнению с известными методами. На практике это означает, что управленческие решения в этой группе роботов становятся более робастными и инвариантными к целому классу возмущающих факторов. Это позволяет на физическом уровне фильтровать информационные выбросы и существенно улучшить показатели группы в реальной обстановке.

Список литературы / References

[1]. L. Jaulin. Mobile Robotics, 1st Edition. ISTE Press-Elsevies, 2015, 314 p.

[2]. G. Klancar, A. Adezar, S. Blazic, and I. Skrjanc. Wheeled Mobile Robots: From Fundamentals Towards Autonomous Systems. Elsevier/Butterwoth-Heinemann, 2017, 502 p.

[3]. G. Klancar, D. Matko, and S. Blazic. Wheeled mobile robots control in a linear platoon. Journal of Intelligent and Robotic Systems, vol. 54, 2017, pp. 709-731.

[4]. М.В. Иванов, О.Ю. Сергиенко, В.В. Тырса и др. Интеграция беспроводной связи для оптимизации распознавания окружения и расчёта траектории движения группы роботов. Труды ИСПРАН, том 31, вып. 2, 2019 г., стр. 67-82. DOI: 10.15514/ISPRAS-2019-31(2)-6 M. Ivanov, O. Sergiyenko, V. Tyrsa et al. Software advances using n-agents wireless communication integration for optimization of surrounding recognition and robotic group dead reckoning. Programming and Computer Software, vol.45, issue 8, 2019, pp. 557-569.

[5]. В.Ю. Осипов. Автоматический синтез программ действий интеллектуальных роботов. Программирование, том 42, no. 3, 2016 г., стр. 47-54 / V. Osipov. Automatic synthesis of action programs for intelligent robots. Programming and Computer Software, Springer, vol. 42, no. 3, 2016, pp. 155–160.

[6]. А.Д. Жданов, Д.Д. Жданов, Н.Н. Богданов и др. Проблемы дискомфорта зрительного восприятия в системах виртуальной и смешанной реальностей. Программирование, том 45, no. 4, 2019 г., стр. 9-18 / A.D. Zhdanov, D.D. Zhdanov, N.N. Bogdanov et al. Discomfort of Visual Perception in Virtual and Mixed Reality Systems. Programming and Computer Software, vol. 45, no. 4, 2019, pp. 147-155.

[7]. А.С. Ярыгина. Методы выполнения и оптимизации приближенных запросов в неоднородных системах. Программирование, том 39, no. 6, 2013 г., стр. 33-44 / A. Yarygina. Execution and

optimization techniques for approximate queries in heterogeneous systems. Programming and Computer Software, vol.39, 2013, pp. 309–317.

[8]. Г.В. Безмен, Н.В. Колесов. Использование программных моделей для диагностирования информационно-управляющих систем. Программирование, том 40, no. 5, 2014, стр. 56-67 / G. Bezmen and N. Kolesov. Program models for diagnosis of information control systems. Programming and Computer Software, vol. 40, no. 2014, pp. 250–258.

[9]. L. Lindner, O. Sergiyenko, M. Rivas-Lopez et al. Exact laser beam positioning for measurement of vegetation vitality. Industrial Robot: An International Journal, vol. 44, issue 4, 2017, pp. 532-541.

[10]. O.Yu. Sergiyenko. Optoelectronic system for mobile robot navigation. Optoelectronics, Instrumentation and Data Processing, vol. 46, no. 5, 2010, pp. 414-428.

[11]. O.Yu. Sergiyenko, M.V. Ivanov, V.V. Tyrsa et al. Data transferring model determination in robotic group. Robotics and Autonomous Systems, vol. 83, 2016, pp. 251-260.

[12]. M. Reyes-García, O. Sergiyenko, M. Ivanov et al. Defining the final angular position of DC motor shaft using a trapezoidal trajectory profile. In Proc. of the IEEE 28th International Symposium on Industrial Electronics (ISIE2019), 2019, pp. 1694-1699.

[13]. C.A. Sepúlveda-Valdez, O. Sergiyenko, D. Hernandez-Balbuena et al. Circular scanning resolution improvement by its velocity close loop control. In Proc. of the IEEE 28th International Symposium on Industrial Electronics (ISIE2019), 2019, pp. 244-249.

[14]. M. Ruderman and M. Iwasaki. Observer of nonlinear friction dynamics for motion control. IEEE Transactions on Industrial Electronics, vol. 62, no. 9, 2015, pp. 5941-5949.

[15]. О.Ю. Сергиенко, С.П. Павлов. Анализ природы атомно-молекулярного взаимодействия поверхностей трения. Вестник национального технического университета “ХПИ”, том 1, no. 6, 2002 г., стр. 39-43 / O.Yu. Sergienko and S.P. Pavlov. Analysis of the nature of the atomic-molecular interaction of the friction surfaces. Vestnik Natsionalnogo tekhnicheskogo universiteta: Kharkovskiy Politechnitscheskiy Institut, vol. 1, no.6, 2002, pp. 39-43 (in Russian)

[16]. M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. Diagnosis and Fault-Tolerant Control. Springer-Verlag, 2006, 672 p.

[17]. S. Ding. Data-driven design of fault diagnosis and fault-tolerant control systems. Springer-Verlag, 2014, 300 p.

[18]. I. Samy, I. Postlethwaite, and D. Gu. Survey and application of sensor fault detection and isolation schemes. Control Engineering Practice, vol. 19, 2011, pp. 658–674.

[19]. V. Utkin. Sliding Modes in Control Optimiztion. Springer, Berlin, 1992, 286 p.

[20]. K. Chandra, H. Alwi, and C. Edwards. Fault reconstruction for a quadrotor using an lpv sliding mode observer. IFAC-PapersOnLine, vol. 48, issue 21, pp. 374-379, 2015.

[21]. C. Edwards, S. Spurgeon, and R. Patton. Sliding mode observers for fault detection and isolation. Automatica, vol. 36, 2000, pp. 541–553.

[22]. C. Tan and C. Edwards. Sliding mode observers for robust detection and reconstruction of actuator and sensor faults. International Journal of Robust Nonlinear Control, vol. 13, 2003, pp. 443–463.

[23]. C.P. Tan and C. Edwards. Robust fault reconstruction using multiple sliding mode observers in cascade: development and design. In Proc. of the American Control Conference, 2009, pp. 3411–3416.

[24]. A. Zhirabok, A. Zuev, and A. Shumsky. Fault diagnosis in linear systems via sliding mode observers. International Journal of Control, vol. 94, issue 2, 2021, pp. 327-335.

[25]. A. Brahim, S. Dhahri, F. Hmida, and A. Sellami. Simultaneous actuator and sensor faults reconstruction based on robust sliding mode observer for a class of nonlinear systems. Asian Journal of Control, vol. 19, issue 1, 2017, pp. 362–371.

[26]. J. He and C. Zhang. Fault reconstruction based on sliding mode observer for nonlinear systems. Mathematical Problems in Engineering, vol. 2012, 2012, pp. 1–22.

[27]. X. Yan and C. Edwards. Nonlinear robust fault reconstruction and estimation using a sliding modes observer. Automatica, vol. 43, 2007, pp. 1605–1614.

[28]. J. Chan, C. Tan, and H. Trinh. Robust fault reconstruction for a class of infinitely unobservable descriptor systems. International Journal of Systems Science, vol. 48, issue 8, 2017, pp. 1646-1655.

[29]. H. Alwi and C. Edwards. Fault tolerant control using sliding modes with on-line control allocation. Automatica, vol. 44, 2008, pp. 1859–1866.

[30]. C. Edwards, H. Alwi, H. and C. Tan. Sliding mode methds for fault detection and fault tolerant control with application to aerospace systems. International Journal of Applied Mathematics and Computer Science, vol. 22, 2012, pp. 109–124.

[31]. V. Filaretov, A. Zuev, A. Zhirabok, and A. Protcenko. Development of fault identification system for electric servo actuators of multilink manipulators using logic-dynamic approach. Journal of Control Science and Engineering, vol. 2017, 2017, pp. 1–8.
[32]. H. Meziane, C. Labarre, S. Lefteriu, M. Defoort, and M. Djemai. Fault detection and isolation for a multi-cellular converter based on sliding mode observer. IFAC-PapersOnLine, volume 48, issue 21, 2015, pp. 164-170.
[33]. M. Mohamed, X-G. Yan, S. Spurgeon, and B. Jiang. Robust sliding mode observer design for interconnected systems with application to multimachine power systems. In Proc. of the IEEE Conference on Decision and Control, 2016, pp. 6246–6251.
[34]. K. Zhang, B. Jiang, X. Yan, and Z. Mao. Sliding mode observer based incipient sensor fault detection with application to high-speed railway traction device. ISA Transactions, vol. 63, 2016, Pages 49-59.
[35]. M. Corradini, G. Orlando, and G. Parlangeli. A fault tolerant sliding mode controller for accommodating actuator failures. In Proc. of the 44th IEEE Conference on Decision and Control, 2005, pp. 3091–3096.
[36]. C. Edwards, H. Alwi, H. and C. Tan. Sliding mode methods for fault detection and fault tolerant control. In Proc. of the Conference on Control and Fault-Tolerant Systems, 2010, pp. 106–117.
[37]. C. Edwards and C. Tan. A comparison of sliding mode and unknown input observers for fault reconstruction. European Journal of Control, vol. 12, 2006, pp. 245–260.
[38]. А.Н. Жирабок, А.Е. Шумский, С.В. Павлов. Диагностирование линейных динамических систем непараметрическим методом. Автоматика и телемеханика, no. 7, 2017, стр. 3-21 / A. Zhirabok, A. Shumsky, and S. Pavlov. Diagnosis of linear dynamic systems by the nonparametric method. Automation and Remote Control, vol. 78, 2017, pp. 1173–1188.
[39]. A. Zhirabok, A. Shumsky, S. Solyanik, and A. Suvorov. Fault detection in nonlinear systems via linear methods. International Journal of Applied Mathematics and Computer Science, vol. 27, no. 2, 2017, pp. 261–272.

Информация об авторах / Information about authors

Олег Юрьевич СЕРГИЕНКО – доктор технических наук, заведующий отделом прикладной физики Института инженерии. Область научных интересов: автоматизированная метрология, системы машинного зрения, быстрые электрические измерения, системы управления, роботизированная навигация и лазерные 3D-сканеры.

Oleg Yurievich SERGIYENKO, Doctor of Technical Sciences, Head of Applied Physics Department of Instituto de Ingenieria. Research interests: automated metrology, machine vision systems, fast electrical measurements, control systems, robot navigation and 3D laser scanners.

Алексей Нилович ЖИРАБОК – доктор технических наук, профессор, профессор кафедры автоматизации и управления. Область научных интересов: теория нелинейных систем, диагностика неисправностей, скользящие наблюдатели.

Alexey Nilovich ZHIRABOK, Doctor of Technical Sciences, Professor, Professor of the Department of Automation and Control. Research interests: nonlinear system theory, fault diagnosis, sliding mode observers.

DOI: 10.15514/ISPRAS–2021–33(1)–11

Смягчение неопределенности при разработке научных приложений в интегрированной среде

[1,2,3] *А.Н. Черных, ORCID: 0000-0001-5029-5212 <chernykh@cicese.mx>*
[4] *И.В. Бычков, ORCID: 0000-0002-1765-0769 <bychkov@icc.ru>*
[4] *А.Г. Феоктистов, ORCID: 0000-0002-9127-6162 <agf65@icc.ru>*
[4] *С.А. Горский, ORCID: 0000-0003-0177-9741 <gorsky@icc.ru>*
[4] *И.А. Сидоров, ORCID: 0000-0002-2398-5426 <ivan.sidorov@icc.ru>*
[4] *Р.О. Костромин, ORCID: 0000-0001-8406-8106 <kostromin@icc.ru>*
[5] *А.В. Еделев, ORCID: 0000-0003-2219-9754 <flower@isem.irk.ru>*
[6] *В.И. Зоркальцев, ORCID: 0000-0003-1001-1400 <zork@isem.irk.ru>*
[3,7,8,9] *А.И. Аветисян, ORCID: 0000-0002-0470-9944 <arut@ispras.ru>*

[1] *Центр научных исследований и высшего образования*
Мексика, 22860, Нижняя Калифорния, Энсенада, ш. Тихуана-Энсенада, 3918
[2] *Южно-Уральский государственный университет,*
454080, Россия, Челябинск, проспект Ленина, 76, Россия
[3] *Институт системного программирования им. В.П. Иванникова РАН,*
109004, Россия, Москва, ул. А. Солженицына, 25, Россия
[4] *Институт динамики систем и теории управления им. В.М. Матросова СО РАН,*
664033, Россия, Иркутск, ул. Лермонтова, 134, а/я 29, Россия
[5] *Институт систем энергетики им. Л.А. Мелентьева СО РАН,*
664033, Россия, Иркутск, ул. Лермонтова, 130, Россия
[6] *Лимнологический институт СО РАН,*
664033, Россия, Иркутск, ул. Улан-Баторская, 3, а/я 278, Россия
[7] *Московский Государственный университет имени М. В. Ломоносова,*
Москва, 119991, ГСП-1, Ленинские горы, д. 1
[8] *Московский физико-технический институт,*
141700, Московская область, г. Долгопрудный, Институтский пер., 9
[9] *НИУ "Высшая школа экономики",*
101000, Россия, г. Москва, ул. Мясницкая, д. 20

Аннотация. В статье представлены новые средства инструментария Orlando Tools. Он используется в качестве основы интегрированного программного окружения для разработки распределенных пакетов прикладных программ. Дополнительные средства Orlando Tools ориентированы на смягчение различных типов неопределенностей, возникающих при распределении заданий в интегрированной вычислительной среде. Они обеспечивают непрерывную интеграцию, доставку и развертывание прикладного и системного программного обеспечения пакетов. Это помогает существенно снижать негативное влияние неопределенности на время решения проблем, надежность вычислений и эффективность использования ресурсов. Экспериментальный анализ результатов решения практических задач наглядно демонстрирует преимущества применения инструментария.

Ключевые слова: распределенная среда; пакеты п программ; программное обеспечение; непрерывная интеграция

Для цитирования: Черных А.Н., Бычков И.В., Феоктистов А.Г., Горский А.С., Сидоров И.А., Костромин Р.О., Еделев А.В., Зоркальцев В.И., Аветисян А.И. Смягчение неопределенности при разработке научных приложений в интегрированной среде. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 151-172. DOI: 10.15514/ISPRAS–2021–33(1)–11

Благодарности. Исследование выполнено при поддержке РФФИ, проекты № 19-07-00097-а и № 18-07-01224-а. Разработка агентов метамониторинга и распределения ресурсов выполнена при частичной поддержке научного проекта IV.38.1.1 программы фундаментальных исследований СО РАН.

Mitigating Uncertainty in Developing Scientific Applications in Integrated Environment

[1,2,3] *A. Tchernykh, ORCID: 0000-0001-5029-5212 <chernykh@cicese.mx>*
[4] *I.V. Bychkov, ORCID: 0000-0002-1765-0769 <bychkov@icc.ru>*
[4] *A.G. Feoktistov, ORCID: 0000-0002-9127-6162 <agf@icc.ru>*
[4] *S.A. Gorsky, ORCID: 0000-0003-0177-9741 <gorsky@icc.ru>*
[4] *I.A. Sidorov, ORCID: 0000-0002-2398-5426 <ivan.sidorov@icc.ru>*
[4] *R.O. Kostromin, ORCID: 0000-0001-8406-8106 <kostromin@icc.ru>*
[5] *A.V. Edelev, ORCID: 0000-0003-2219-9754 <flower@isem.irk.ru>*
[6] *V.I.Zorkalzev, ORCID: 0000-0003-1001-1400 <zork@isem.irk.ru>*
[3,7,8,9] *A.I. Avetisyan, ORCID: 0000-0002-0470-9944 <arut@ispras.ru>*

[1] *Centro de Investigación Científica y de Educación Superior, 3918, Ensenada-Tijuana Highway, Ensenada, 22860, Mexico*
[2] *South Ural State University, Chelyabinsk, 76, Lenin prospekt, Chelyabinsk, 454080, Russia*
[3] *Ivannikov Institute for System Programming of the RAS, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia*
[4] *Matrosov Institute for System Dynamics and Control Theory of the Siberian Branch of the RAS, 134, Lermontov st., Irkutsk, postbox 292, 664033, Russia*
[5] *Melentiev Energy Systems Institute of the Siberian Branch of the RAS, 130, Lermontov st., Irkutsk, 664033, Russia*
[6] *Limnological Institute of the Siberian Branch of the RAS, 3, Ulan-Batorskaya st., Irkutsk, postbox 278, 664033, Russia*
[7] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia*
[8] *Moscow Institute of Physics and Technology (State University) 9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia*
[9] *National Research University Higher School of Economics (HSE) 11 Myasnitskaya Ulitsa, Moscow, 101000, Russia*

Abstract. The paper represents new means of the Orlando Tools framework. This framework is used as the basis of an integrated software environment for developing distributed applied software packages. The additional means are focused on mitigating various types of uncertainties arising from the job distribution in an integrated computing environment. They provide continuous integration, delivery, and deployment of applied and system software of packages. This helps to significantly reduce the negative impact of uncertainty on problem-solving time, computing reliability, and resource efficiency. An experimental analysis of the results of solving practical problems clearly demonstrates the advantages of applying these means.

Keywords: distributed environment; software packages; software; continuous integration

For citation: Tchernykh A., Bychkov I.V., Feoktistov A.G., Gorsky S.A., Sidorov I.A., Kostromin R.O., Edelev A.V., Zorkalzev V.I., Avetisyan A.I. Mitigating Uncertainty in Developing Scientific Applications in Integrated Environment. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 151-172 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–11

Acknowledgements. The study is supported by the Russian Foundation of Basic Research, projects no. 19-07-00097 and no. 18-07-01224. The development of meta-monitoring and resource allocation agents was supported in part by the Basic Research Program of SB RAS, project no. IV.38.1.1.

1. Введение

Анализ современных тенденций развития высокопроизводительных вычислений применительно к математическому моделированию сложных систем позволяет сделать ряд важных выводов. Как правило, такое математическое моделирование связано с решением NP-трудных задач, требующих применения высокопроизводительных вычислительных систем. Однако такие системы значительно различаются по своим вычислительным возможностям, аппаратным и программным платформам, системным архитектурам, интерконнектам, пользовательским интерфейсам и другим компонентам. Зачастую конечные пользователи вычислительных систем сталкиваются со следующими разнообразными трудностями их использования (особенно, при переходе пользователей на новую систему): учетом специфики предметной области, включая использование слабоструктурированных данных, наличие различных неопределенностей, изменение размерности решаемых задач и т.п.; проведением экспериментов в гетерогенной системе, например, адаптацией алгоритмов решения задачи на различные архитектуры узлов системы; прогнозированием производительности узлов и времени выполнения научных приложений.

Необходимо создание универсального информационно-вычислительного окружения, обеспечивающего инструментальную поддержку подготовки и проведения крупномасштабного эксперимента в гетерогенной среде и его специализацию под конкретную предметную область [1]. Такая среда позволит расширить набор используемых моделей, алгоритмов, приложений и баз данных. Кроме того, она обеспечит участие различных научных групп в экспериментах с использованием нужных ресурсов.

Облачные платформы, грид-системы и суперкомпьютерные центры коллективного пользования (ЦКП) отождествляют три организационно-функциональные формы параллельных и распределенных вычислений, которые востребованы научным сообществом.

Грид-системы поддерживают стандартное программное обеспечение (ПО) и сервисы для совместного использования и управления федеративными и географически распределенными ресурсами для вычислений и обработки данных [2].

Облачные платформы быстро и динамично предоставляют необходимые вычислительные ресурсы и хранилища данных, а также обеспечивают пропускную способность сети с требуемым качеством [3]. Тем не менее, конечные пользователи вынуждены самостоятельно доставлять и развертывать свое ПО и данные поверх выделенной инфраструктуры, а также управлять ими. Кроме того, использование облачных ресурсов и сервисов зачастую обходится дороже, чем проведение экспериментов в грид-системах.

Отличительными преимуществами суперкомпьютерных ЦКП является предоставление платных и бесплатных ресурсов и услуг в соответствии с принятыми административными политиками, предоставление базового системного и прикладного ПО, обеспечение унифицированного и жестко регламентированного доступа к вычислительным ресурсам, оценка эффективности вычислений и использования ресурсов, техническая и методическая поддержка конечных пользователей.

В общем случае ресурсы ЦКП являются невыделенными и разделяемыми всеми пользователями центра. Однако по желанию ресурсы могут быть выделены пользователям для решения сложных задач с использованием облачных или грид-вычислений [4]. Различия в способах доступа к вышеперечисленным ресурсам и их использования существенно ограничивают сферу применения высокопроизводительных вычислений исследователями. Таким образом, интеграция облачных и грид-вычислений, включая ресурсы ЦКП, для

поддержки решения трудно решаемых задач обеспечит широкие возможности для проведения массовых масштабных научных экспериментов в такой среде [5].

В статье представлены новые возможности инструментального комплекса Orlando Tools (OT) [6]. Он является основой интегрированной вычислительной среды для разработки и применения распределенных пакетов прикладных программ (РППП). Предполагается, что среда создается на основе трех вышеперечисленных форм организации вычислений.

РППП представляют специальный класс научных приложений [7]. ПО таких пакетов имеет модульную структуру. Модули пакета реализуют алгоритмические знания о процессе решения задач. Набор модулей, используемых для решения задачи, и порядок их выполнения описываются схемой решения задачи. Направленный ациклический граф применяется для отражения отношений и потоков данных между модулями. Понятие схемы решения задачи соответствует понятию абстрактного научного рабочего процесса (workflow) [8]. Спецификация схемы решения задачи (задание) для мета-планировщиков среды описывает информацию об используемых модулях пакета, их входных и выходных данных, требованиях к ресурсам, системам хранения данных и т.п. В отличие от систем поддержки научных рабочих процессов применение РППП связано с решением класса задач в некоторой предметной области с учетом интенсивного изменения алгоритмических знаний.

Применение подсистемы, реализующей новые возможности OT, обеспечивает обработку слабоструктурированных предметных данных, автоматизацию сборки, отладки и совместного тестирования модулей РППП, прогнозирование времени выполнения этих модулей, их классификацию относительно разнородных ресурсов среды и, как следствие, повышение качества распределения ресурсов. Преимущества применения подсистемы показаны на примере разработки двух версий РППП для исследования направлений устойчивого развития топливно-энергетического комплекса (ТЭК) Вьетнама с применением подсистемы [9] и без нее [10]. В первом случае, экспериментальные результаты показали существенное сокращение общего времени подготовки и проведения экспериментов.

2. Связанные исследования

Вопросы создания интегрированного программного окружения для поддержки крупномасштабных научных экспериментов с использованием различных форм организации высокопроизводительных вычислений не очень широко рассматриваются в научной литературе. Тем не менее, можно выделить ряд интересных проектов, имеющих большую практическую значимость организации совместных вычислений.

Модульный инструментарий Distributed and Unified Numerics Environment (DUNE) для решения уравнений в частных производных представлен в [11]. Он обеспечивает использование высокопроизводительных вычислений научными приложениями.

Интеграция приложений, например, популярных систем Computer Aided Engineering, в распределенную вычислительную среду может быть выполнена на основе сервис-ориентированного подхода, предложенного в [12]. Разработан новый алгоритм управления вычислениями в такой среде [13].

Проект SPPEXA направлен на поддержку исследований в области интегрированной разработки вычислительных алгоритмов, системного и прикладного ПО, исследования данных и управления ими, среды программирования и инструментальных средств [14].

В [15] описывается веб-ориентированная платформа Galaxy для научного анализа больших биомедицинских наборов данных в области геномики, протеомики, метаболомики и визуализации. Большое число ученых активно использует ее по всему миру в своих крупномасштабных экспериментах.

Гибкая облачная платформа Globus [16] может быть использована для реализации, развертывания и эксплуатации сервисов при поддержки подобных экспериментов. Аналогичные средства, реализованные с использованием агентов, предложены в [17].

Зачастую исследователям необходимо выполнение композиции доступных сервисов с определенными критериями качества. Как правило, задача построения такого композитного сочетания сервисов является NP-трудной. Подход к ее решению представлен в [18].

Концепция и принципы работы интегрированного программного окружения предложены в [1]. Они предполагают разработку инструментальных средств, входящих в такое окружение, для реализации основных этапов крупномасштабных экспериментов, включая поддержку длительного жизненного цикла прикладного и системного ПО. Однако вышеперечисленные и другие существующие проекты оставляют многие проблемы нерешенными в полной мере. В их числе можно выделить следующие проблемы [1, 14]: исследование процессов работы приложений в разнородных ресурсах среды; проектирование приложений с учетом архитектуры среды; применение связующего ПО для поддержки взаимодействия между приложениями и операционными системами, используемыми в узлах среды; интеграция связующего ПО и приложений по управлению программно-аппаратными ресурсами среды; интеллектуализация управления заданиями; автоматизация контроля, сборки и тестирования системного и прикладного ПО; извлечение, анализ и преобразование предметных данных.

Эти проблемы также актуальны для научных рабочих процессов [19]. Их решение способствует смягчению неопределенности в распределении рабочих процессов по ресурсам.

Системы поддержки научных рабочих процессов и РППП имеют ряд отличий. Как правило, научные рабочие процессы, такие как LIGO, Montage, CyberShake, SIPHT или Epigenomics, ориентированы на решение одной задачи в некоторой предметной области. Пакет предназначен для решения класса задач. Схемы решения задач отражают многообразие используемых в пакете алгоритмов. Научные рабочие процессы не претерпевают существенных изменений при их использовании [20]. Пакеты, напротив, активно развиваются. Их алгоритмические знания интенсивно изменяются. В пакетах могут разрабатываться новые схемы решения задач или модифицироваться существующие схемы.

Анализ состояния исследований в данной области показывает, что в настоящее время нет средств для реализации непрерывной интеграции ПО применительно к РППП в полной мере [21]. Требуется разработка новых средств для решения данной проблемы. Такие средства должны обеспечивать выполнение специальных функций непрерывной интеграции, обусловленных спецификой пакетов, и взаимодействие с существующими средствами непрерывной интеграции, реализующими традиционные функции. Проведенный сравнительный анализ возможностей известных средств непрерывной интеграции показывает, что для реализации подобного взаимодействия наиболее подходит GitLab [22].

В отличие от большинства научных рабочих процессов [23], пакеты создаются для выполнения в гетерогенной распределенной вычислительной среде. Ресурсы, интегрируемые в рамках среды, могут быть выделенными и невыделенными. Возможностей традиционных средств виртуализации, обычно использующихся системами поддержки научных рабочих процессов [24], недостаточно для интеграции вышеупомянутых ресурсов в единую среду.

Сравнительный анализ систем виртуализации представлен в [9]. Он показал, что в динамической среде необходимо применять связующее ПО для поддержки эффективного взаимодействия с различными гипервизорами и системами управления контейнерами. Такой системой, которая может использоваться в выделенных ресурсах, является платформа OpenStack [25]. Однако для невыделенных ресурсов необходима разработка специального средства запуска виртуальных машин (ВМ).

Вышеупомянутые отличия приводят к различным типам неопределенностей, которые возникают при планировании вычислений и распределении ресурсов. В их числе

неопределенность в возможностях ресурсов, свойствах заданий, предпочтениях владельцев и пользователей ресурсов, качестве услуг и др. С точки зрения неопределенности, основными проблемами в масштабируемых научных приложениях являются планирование вычислений и распределение ресурсов. Активно развиваются исследования по улучшению планирования и распределения за счет уменьшения перечисленных выше неопределенностей.

Методы анализа структур рабочего процесса и их декомпозиции рассмотрены в [26]. Анализ рабочего процесса позволяет уточнить свойства его компонентов. Декомпозиция проводится с целью рационального распределения компонентов по ресурсам. Кроме того, автоматизация планирования рабочих процессов (схем решения задач) для непроцедурных формулировок задач позволяет значительно упростить выбор приложений, необходимых для решения задач, при наличии неопределенности в использовании программного обеспечения [27].

Алгоритм уменьшения неопределенности распределения заданий проблемно-ориентированных приложений, учитывающий специфику решаемых задач, предложен в [28].

Адаптивное распределение ресурсов в условиях неопределенности исследуется в [29]. Предложены модель задержки выполнения заданий с целью прогнозирования их времени выполнения и стратегия адаптивного стохастического распределения заданий. Предложен подход к моделированию облачных вычислений с неопределенностью [30], учитывающий предоставление ресурсов в гибридной среде и обеспечивающий повышение эффективности предоставления ресурсов с учетом их конфиденциальности, целостности и доступности.

Неопределенность качества услуг, связанная с координацией предпочтений владельцев и пользователей ресурсов, может быть смягчена путем применения агентных технологий [31].

3. Orlando Tools: основа для интегрированного программного окружения

Архитектура интегрированной вычислительной среды показана на рис. 1. Подсистемы ОТ представляют ее основу. Внешние информационно-вычислительные системы и ресурсы используются в процессе разработки и применения РППП.

Выделяются следующие основные подсистемы ОТ: *пользовательский интерфейс*, обеспечивающий доступ различных категорий пользователей РППП к их компонентам, подсистемам ОТ и ресурсам среды; *конструктор концептуальной модели среды*, реализующий спецификацию знаний пользователей РППП о предметной области решаемых задач; *конструктор библиотек модулей*, поддерживающий разработку и модификацию прикладного ПО РППП; *менеджер ПО*, осуществляющий автоматизацию контроля версий, тестирование, доставку и развертывание как прикладного и системного ПО РППП среды; *менеджер вычислений*, реализующий управление вычислениями на разнородных ресурсах среды; *менеджер состояния среды*, выполняющий мониторинг ее ресурсов; *API* для доступа к информационно-вычислительным системам и ресурсам.

Знания о предметных областях РППП, информация о ресурсах среды и сведения об информационно-вычислительных системах и ресурсах сохраняются в соответствующих разделах базы знаний ОТ. Для извлечения таких знаний могут быть использованы различные базы предметных данных. Исходные данные и результаты вычислений хранятся в базах расчетных данных инструментального комплекса. В процессе вычислений могут использоваться дополнительные системы хранения данных.

Пользовательский интерфейс. Компоненты подсистем ОТ представлены сервисами. Для пользователей реализован веб-ориентированный доступ к их операциям. Сервисы, представляющие компоненты, к которым могут обращаться другие программные сущности (другие компоненты, модули пакетов или внешние программные системы), имеют дополнительное описание на языке WSDL.

Конструктор концептуальной модели. Концептуальная модель среды, формируемая в рамках конструктора, состоит из трех частей. Первая часть описывает алгоритмические знания о решении задач предметной области РППП. Вторая часть содержит информацию об инфраструктурных объектах среды. Знания о процессах непрерывной интеграции, доставки и развертывания ПО на ресурсах среды формируют третью часть модели.

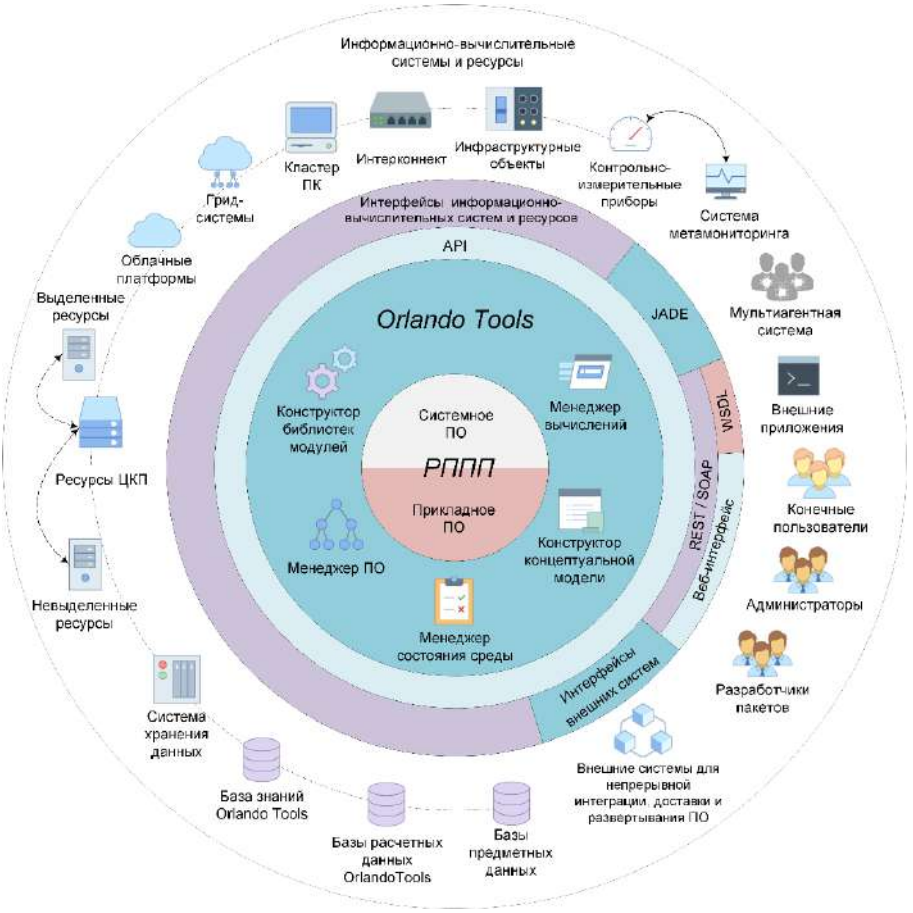


Рис. 1. Архитектура интегрированной вычислительной среды
Fig. 1. Architecture of integrated computing environment

Алгоритмические знания являются результатом структурного анализа предметной области. В ОТ она описывается с помощью специального набора концептуальных понятий. В их числе такие концепты, как параметры, операции, продукции, модули и схемы решения задач.

Параметры представляют характеристики предметной области, необходимые в процессе решения задач. Над множеством параметров определяются операции. Они ставят в соответствие подмножеству входных параметров, значения которых должны быть известны, другое подмножество выходных параметров, значения которых нужно вычислить. Связи между операциями определяются на основе их входных и выходных параметров. Продукции определяют логические условия применения операций в процессе решения задач в зависимости от промежуточных результатов вычислений и состояния ресурсов среды.

Схемы решения задач создаются разработчиками РППП и их конечными пользователями по процедурной или непроцедурной постановкам задач на основе информационных связей

между операциями. Процедурная постановка задачи представляет собой список операций, требуемых для решения задачи и частично-упорядоченных в рамках ярусно-параллельной формы направленного ациклического графа. Непроцедурная постановка задачи формулируется в следующем виде: «По исходному подмножеству параметров, значения которых известны, вычислить значения параметров целевого подмножества». В случае такой постановки задачи, нахождение операций, требуемых для решения задачи, и их упорядочение выполняются автоматически. Модули ассоциируются с программными реализациями операций. Каждая схема решения задачи через свои операции определят набор модулей, необходимых для ее выполнения. Описание модуля может включать текст модуля, язык программирования, тип и назначение входных, выходных и транзитных параметров, методы передачи параметров и обработки нестандартных ситуаций, требуемый компилятор и его опции, формат вызова, и другая необходимая информация.

В рамках данного исследования представлены новые возможности ОТ по автоматизации контроля версий и тестирования ПО РППП. В связи с этим описание модулей расширено дополнительными сведениями о репозиториях с их исходными и бинарными файлами, сценариях их сборки и тестирования в различных узлах среды как в отдельности, так и в составе схем решения задач, тестовых данных, способах доставки и развертывания модулей на ресурсах среды. В частности, конструктор концептуальной модели обеспечивает разработку необходимых скриптов для внешних систем. Эти скрипты реализуют сценарии непрерывной интеграции, доставки и развертывания ПО.

Информация об инфраструктурных объектах среды включает характеристики узлов, каналов связи, сетевых устройств, топологии сети и других объектах. Кроме того, эти характеристики включают в себя статистику сбоев и отказов программного обеспечения и оборудования. Дополнительно учитываются сведения об административных политиках, определенных для ресурсов среды. Такие сведения включают данные о пользователях и их заданиях, правах доступа к ресурсам и квотах на их использование, а также системах управления ресурсами, их характеристиках и дисциплинах диспетчеризации вычислительных работ.

Информация об инфраструктуре среды и административных политиках извлекается с помощью различных информационно-вычислительных и управляющих систем, контрольно-измерительных приборов и систем мониторинга. Она отображается в концептуальной модели с помощью системных параметров, операций и модулей. Описание концептуальной модели хранится в базе знаний ОТ. Ее разделы распределены между ОТ, системой метамониторинга и внешними системами непрерывной интеграции, доставки и развертывания ПО.

Конструктор библиотек модулей. Прикладное ПО РППП может формироваться на основе существующих программных библиотек или создаваться разработчиками пакета. В обоих случаях описание модулей включается в концептуальную модель среды. ОТ предоставляет редактор исходного кода модулей и средства работы с внешними системами их сборки, отладки и тестирования. Системное ПО формируется на основе системных библиотек ОТ.

Менеджер ПО является новой подсистемой ОТ. Он предназначен для выявления и частичного устранения потенциальных проблем при непрерывной интеграции, доставки и развертывания новых версий как прикладного, так и системного ПО РППП.

Его основные функции включают: работу с репозиториями исходных и бинарных файлов ПО, включая поддержку контроля их версий; автоматизацию сборки и тестирования модулей в отдельности; создание образов ВМ для выполнения модулей; автоматизацию тестирования модулей в составе схем решения задач; прогнозирование времени выполнения модулей в узлах среды; автоматизацию доставки и развертывания модулей в среде.

Новые средства для мета-описания процессов выполнения всех перечисленных выше операций впервые реализованы в конструкторе концептуальной модели среды. Общая схема непрерывной интеграции, доставки и развертывания прикладного и системного ПО представлена на рис. 2.

Сведения о производительности узлов, полученные при тестовом выполнении модулей, позволяют установить степень соответствия производительности ресурсов критериям эффективности выполнения модулей. Такое соответствие устанавливается администраторами среды заблаговременно, до начала назначения ресурсов в процессе проведения вычислительных экспериментов. Тем самым, обеспечивается существенное смягчение неопределенности в последующем процессе распределения заданий по ресурсам.

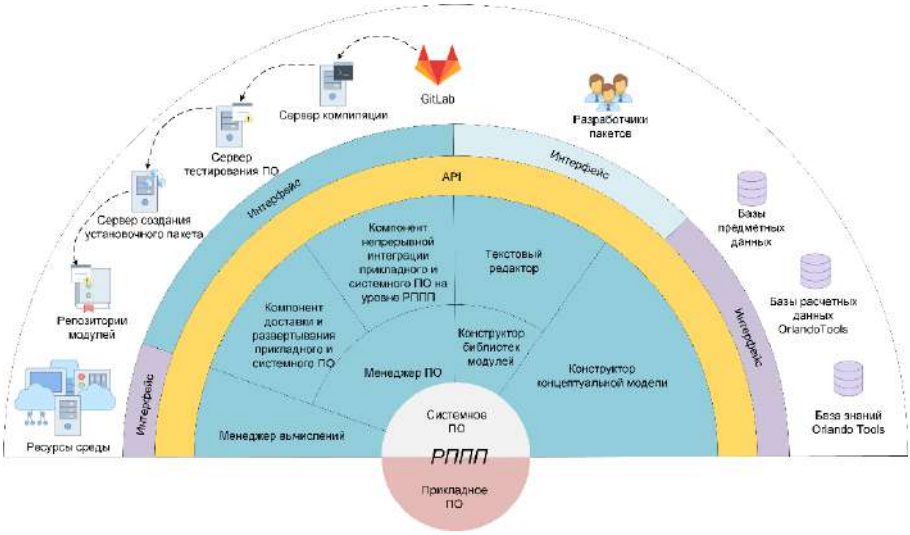


Рис. 2. Схема непрерывной интеграции, доставки и развертывания ПО
Fig. 2. Scheme of continuous integration, delivery, and deployment of software

Менеджер вычислений включает компоненты для формулировки постановок задач и построения схем их решения, генератор заданий для выполнения построенных схем, планировщик и интерпретатор. Планировщик производит декомпозицию исходных схем решения задач, представленных в заданиях, на подсхемы с целью оптимизации распределения вычислительной и коммуникационной нагрузки на ресурсы среды. Интерпретатор осуществляет выполнение схем или подсхем решения задач.

Дополнительно менеджер вычислений может передать задание на обработку мультиагентной системе (МАС) [32]. Она реализована на основе Java Agent Development Framework (JADE) [33]. МАС относится к классу метапланировщиков, таких как GridWay [34] или Condor DAGMan [35]. В то же время, в отличие от подобных средств, она позволяет управлять потоками заданий в среде, интегрирующей ресурсы облачных платформ, грид-систем и ЦКП.

МАС включает агентов классификации заданий, мониторинга и диспетчеризации ресурсов среды, а также агентов, непосредственно представляющих интегрированные ресурсы (ресурсных агентов). Ресурсные агенты объединяются в новое виртуальное сообщество для выполнения каждого задания. Условием включения агента в сообщество для выполнения конкретного задания является наличие у него ресурсов, обладающих необходимыми возможностями для выполнения модулей данного задания.

Схема распределения ресурсов агентами представлена на рис. 3. Агент-классификатор определяет класс задания и формирует виртуальное сообщество ресурсных агентов. Система классификации детально рассмотрена в [36]. Классифицированное задание передается в очередь агенту-диспетчеру. При обработке очереди заданий агент-диспетчер производит их декомпозицию на подзадания, которые затем распределяются по сегментам интегрированной вычислительной среды, представляющим облачные платформы, грид-системы и ЦКП.

Для выполнения подзаданий запускаются ВМ, образы которых предварительно созданы менеджером ПО. ВМ объединяются в единую виртуальную среду выполнения РППП.

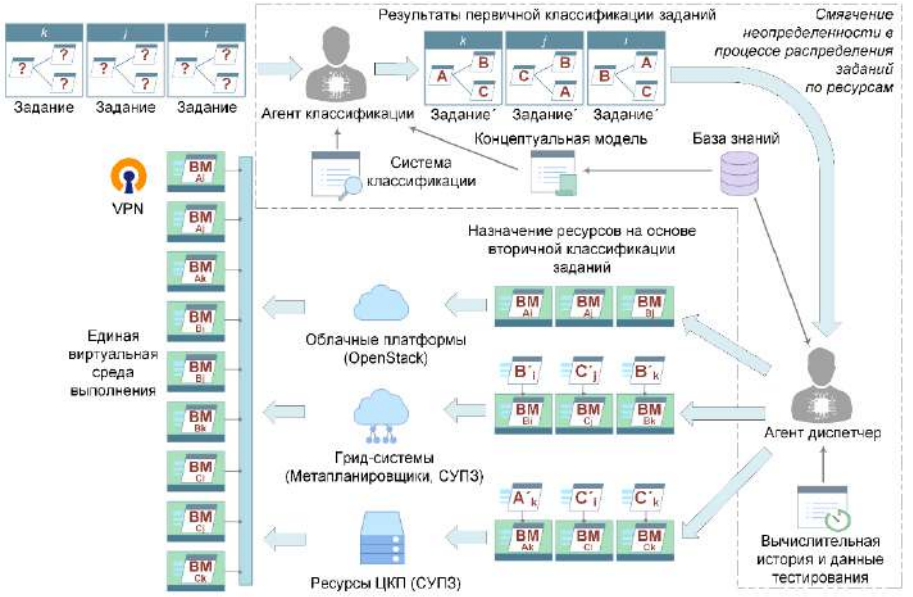


Рис. 3. Назначение ресурсов агентами
Fig. 3. Resource allocation by agents

Управление облачными ресурсами осуществляется с помощью OpenStack. При запуске ВМ на ресурсах грид-систем и ЦКП формируются задания для систем управления прохождением заданий (СУПЗ), установленных на ресурсах. Эти задания отправляются в очереди СУПЗ. Для ускорения прохождения заданий выполняется поиск узлов со свободными слотами в их расписаниях обслуживания заданий. Каждый слот определяет число ядер, которые могут быть использованы для выполнения задания в течение некоторого интервала времени.

В целом агенты обеспечивают смягчение неопределенности в распределении заданий по ресурсам. Оно осуществляется автоматически в четыре этапа.

На первом этапе осуществляется первичная классификация задания путем анализа его спецификации с помощью специальной характеристической функции. Проверяется формальное соответствие характеристик задания и их значений характеристикам предопределенных классов и допустимым областям их значений. В результате первичной классификации задание может принадлежать нескольким классам. Выбор ресурсов, соответствующих классу задания, производится на втором этапе. В рамках третьего этапа выполняется вторичная классификация заданий на основе анализа статистической информации о вычислительной истории выполнений задания и их соответствия имеющимся классам. Результаты анализа применяются для корректировки данных первичной классификации. На последнем этапе результаты вторичной классификации уточняются на основе полученных ранее данных тестирования прикладного ПО соответствующим менеджером или сведений о прогнозируемом времени выполнения модулей.

Первые два этапа выполняются агентом классификации заданий. Последние два этапа осуществляются агентом-диспетчером. Предварительно администраторы определяют необходимые классы заданий в системе классификации и затем устанавливают соответствие ресурсов классам на основе своего экспертного опыта. Окончательное распределение заданий по ресурсам выполняется ресурсными агентами с помощью тендера вычислительных работ (заданий) [37]. В рамках тендера торги реализованы на основе закрытого

однораундового аукциона Викри второй цены [38]. В процессе торгов учитываются дополнительные условия выполнения заданий. В их числе предпочтения владельцев ресурсов и критерии эффективности решения задач, определяемые пользователями среды.

Менеджер состояния среды получает информацию о состоянии среды от системы метамониторинга, реализованной на основе агентного подхода [32]. Эта система осуществляет сбор сведений о состоянии инфраструктурных объектов среды с помощью контрольно-измерительных приборов (Sensors, APC PowerChute, IPMI, SMART и др.) и локальных систем мониторинга (Ganglia, Nagios, Zabbix, ZenOSS и др.). Дополнительно она осуществляет контроль показателей функционирования систем управления этими объектами.

На основе собранной информации система метамониторинга выполняет оценку текущей вычислительной ситуации, прогнозирует ее развитие и формирует управляющие воздействия на инфраструктурные объекты и системы управления с целью предупреждения или частичного устранения сбоев и отказов программно-аппаратного обеспечения. В случае критической ситуации, когда такие действия невозможно выполнить в автоматическом режиме, система метамониторинга отправляет уведомление администратору среды.

Доступ к информационно-вычислительным системам и ресурсам. Веб-ориентированный доступ пользователей к информационно-вычислительным системам и ресурсам среды обеспечивается средствами ОТ. Данный комплекс взаимодействует с системами и ресурсами по протоколам, поддерживаемыми ими.

4. Смягчение неопределенности

В статье неопределенность рассматривается с точки зрения недостаточности имеющихся знаний относительно знаний, необходимых для принятия субоптимального решения. В нашем случае такими знаниями являются время выполнения модулей в разнородных узлах среды, свойства заданий и возможность выполнения заданий на тех или иных ресурсах.

Оценка времени выполнения модуля. Разработан алгоритм оценки времени выполнения модулей. Модуль запускается из среды профилирования программ на выбранном эталонном узле. В процессе его выполнения определяется время его работы с различными компонентами узла. Общее время выполнения модуля на целевом узле прогнозируется на основе анализа различий характеристик узлов, влияющих на ускорение (замедление) вычислений. Алгоритм включает следующие этапы:

- определение эталонного и целевого узлов;
- определение множества $CR = \{cr_1, cr_2, \dots, cr_m\}$ характеристик эталонного узла, используя знания о ресурсах концептуальной модели среды;
- определение множества $CT = \{ct_1, ct_2, \dots, ct_m\}$ характеристик целевого узла на основе тех же знаний;
- подготовка данных d в расчетной базе знаний ОТ;
- размещение профилировщика, модуля и подготовленных данных на эталонном узле с использованием менеджера ПО;
- запуск профилировщика на эталонном узле;
- запуск модуля из среды профилировщика;
- выполнение модуля на эталонном узле;
- получение значений счетчиков профилировщика;
- определение реального времени $T_r(d)$ выполнения модуля;
- вычисление значений $p_1(d)$, $p_2(d)$, …, $p_n(d)$ из множества P, которые отражают вычислительную нагрузку узла (число выполненных целочисленных операций, число

операций с плавающей запятой, число обращений к основной памяти и кэш-памяти разных уровней, число пропущенных обращений и др.) при выполнении модуля в зависимости от данных d;

- оценка времени $\hat{T}_r(d) = \sum_{l=1}^{k} f_l\big(CR, P, g(d)\big)$ выполнения модуля на эталонном узле;
- оценка времени $\hat{T}_t(d) = \sum_{l=1}^{k} f_l\big(CT, P, g(d)\big)$ выполнения модуля на целевом узле;
- определение погрешности $\varepsilon = T_r(d) - \hat{T}_r(d)$;
- оценка времени $\hat{T}_t'(x)\hat{T}_t(d) + \varepsilon\frac{\hat{T}_r(d)}{\hat{T}_t(d)}$ выполнения модуля на целевом узле.

В алгоритме характеристики cr_1, cr_2, …, cr_m, ct_1, ct_2, …, ct_m, компоненты данных d, время T_r, оценки $\hat{T}_r$ и $\hat{T}_t'$, а также погрешность ε являются системными параметрами. Функции p_1, p_2, …, f_1, f_2, …, f_k и g, а также выражения для вычисления $\hat{T}_r$, $\hat{T}_t'$, и ε реализуются системными модулями. Системные объекты описываются в концептуальной модели среды.

Смягчение неопределенности в распределении заданий по ресурсам. Для оценки степени смягчения неопределенности в процессе распределения заданий по ресурсам в статье используется информационная двоичная энтропия. Данный показатель часто используется для оценки степени неопределенности сложной системы. Предполагается, что уменьшение энтропии ведет к смягчению неопределенности.

Пусть по результатам первичной классификации найдены m классов, характеристикам которых удовлетворяет задание. Среда включает n разнородных ресурсов. Одному классу может соответствовать k ресурсов, $k \in \overline{1,n}$. Информация о соответствии между классами и ресурсами представлена булевой матрицей Y размерности $n \times m$. Элемент матрицы $y_{ji} = 1$ ($y_{ji} = 0$) показывает, что j-й ресурс соответствует (не соответствует) i-му классу. Определим энтропию $E^c(x_c) = \sum_{i=1}^{m} p_i(x_c)E_i^c(x_c)$, отражающую степень неопределенности принадлежности задания одному из классов, где x_c – случайное событие, которое обуславливает принадлежность задания одному из m классов, $p_i(x_c)$ – вероятность такого события относительно i-го класса, $\sum_{i=1}^{m} p_i(x_c) = 1$, $E_i^c(x_c)$ – энтропия, соответствующая данному событию. Значение $E_i^c(x_c)$ вычисляется следующим образом:

$$E_i^c(x_c) = \begin{cases} 0, \text{если } p_i(x_c) = 0, \\ -log_2 p_i(x_c) \text{ в противном случае.} \end{cases}$$

Тем же способом определим энтропию $E^r(x_r) = \sum_{j=1}^{n} p_j(x_r)E_j^r(x_r)$, отражающую степень неопределенности принадлежности задания одному из классов, где x_r – случайное событие, которое обуславливает возможность назначения заданию одного из n ресурсов, $p_j(x_r)$ – вероятность такого события относительно j-го ресурса, $E_j^r(x_r)$ – энтропия, соответствующая данному событию. Значения $p_j(x_r)$ и $E_j^r(x_r)$ вычисляются следующим образом:

$$p_j(x_r) = \sum_{\forall l: y_{jl}=1} p_l(x_C) \Big/ \sum_{j=1}^{n} \sum_{\forall l: y_{jl}=1} p_l(x_C), \quad l \in \overline{1,m}, \quad \sum_{j=1}^{n} p_j(x_r) = 1,$$

$$E_j^r(x_r) = \begin{cases} 0, \text{если } p_j(x_r) = 0, \\ -log_2 p_j(x_r) \text{ в противном случае.} \end{cases}$$

Рассмотрим иллюстративный пример смягчения неопределенности в процессе распределения заданий по ресурсам. Пусть по результатам первичной классификации найдены 8 классов, характеристикам которых удовлетворяет задание, поступившее в среду для выполнения указанного в задании модуля. Эти классы различаются между собой областью допустимых значений одной характеристики, определяющей допустимое время выполнения заданий. Значение данной характеристики не указано в спецификации задания.

Среда включает 3 разнородных ресурса (3 кластера, узлы которых различаются по вычислительным характеристикам). Соответствие ресурсов классам представлено матрицей

$$Y = \begin{bmatrix} 1\,1\,1\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,1\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0 \end{bmatrix}.$$

Имеется вычислительная история выполнения такого задания на ресурсах среды.

В табл. 1 приведены значения $p_1(x_c)$, $p_2(x_c)$, …, $p_8(x_c)$, E^c, $p_1(x_r)$, …, $p_3(x_c)$ и E^c для каждого этапа смягчения неопределенности в распределение заданий по ресурсам.

Табл. 1. Вероятности событий и показатели энтропии
Table 1. Event probabilities and entropy parameters

№	$\boldsymbol{p_1(x_c)}$	$\boldsymbol{p_2(x_c)}$	$\boldsymbol{p_3(x_c)}$	$\boldsymbol{p_4(x_c)}$	$\boldsymbol{p_5(x_c)}$	$\boldsymbol{p_6(x_c)}$	$\boldsymbol{p_7(x_c)}$	$\boldsymbol{p_8(x_c)}$	E^c	$p_1(x_r)$	$p_2(x_r)$	$p_3(x_r)$	E^r
1	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	3.00	0.333	0.333	0.333	1.06
2	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125	3.00	0.444	0.556	0	0.99
3	0.115	0.741	0.132	0.012	0	0	0	0	1.14	0.988	0.012	0	0.09
4	0	0.936	0.064	0	0	0	0	0	0.34	1.000	0	0	0

На первом этапе вероятности принадлежности задания одному из 8 классов равны между собой, так же как равны и вероятности назначения заданию одного из 3 ресурсов. В этом случае $E^c = 3.00$ и $E^r = 1.06$. На следующем этапе выбираются два ресурса, наиболее подходящие для выполнения задания с точки зрения экспертного опыта администраторов ресурсов. При этом степень неопределенности назначения ресурсов заданию понижается до $E^r = 0.99$. Вторичная классификация задания на основе анализа статистической информации о вычислительной истории конкретизирует вероятности принадлежности задания классам, полученные на третьем этапе. Значение E^c становится равным 1.14. В соответствие с этим изменяются вероятности назначения ресурсов заданию. Степень неопределенности назначения ресурсов заданию уменьшается до $E^r = 0.09$. На последнем этапе результаты вторичной классификации уточняются на основе прогнозного времени выполнения модуля или полученных ранее данных его тестирования. В результате этого $E^c = 0.34$ и $E^r = 0.09$. Таким образом, ресурс 1 назначается для выполнения задания.

5. Экспериментальный анализ

РППП для решения задач поиска направлений устойчивого развития топливно-энергетического комплекса. Поиск направлений устойчивого развития как ТЭК в целом, так и отдельных систем энергетики является типовой задачей. Необходимость ее решения многократно возникает в процессе их эксплуатации и развития. Для решения подобной задачи разработан РППП. Развитие ТЭК исследуется с учетом всех комбинаций крупных возмущений, которые могут привести к недостаточному обеспечению конечных потребителей энергоресурсами. Такое исследование приводит к возникновению большого числа возможных состояний комплекса и затрудняет их анализ классическими методами.

В пакете используется новый подход к исследованию развития ТЭК с помощью комбинаторного моделирования. Он основан на рассмотрении различных комбинаций возможных состояний комплекса и переходов между ними для заданного периода T. Поскольку множество возможных состояний является достаточно большим, рассматривается только подмножество допустимых состояний. Множество состояний представлено ориентированным графом $G=<V, E>$, где V – множество вершин (возможных состояний ТЭК), а E – множество ребер (переходов между состояниями). Дуги направлены из состояния s_0 в начальный момент времени τ_0 в направлении состояний в следующий момент времени τ_1 и далее в состояния в следующие моменты времени $\tau_2, \tau_3, \ldots, \tau_n$ (рис. 4). Каждая вершина отражает состояние взаимосвязанных объектов комплекса. Такое состояние включает эксплуатацию, реконструкцию, модернизацию и создание объектов. Каждый объект выполняет одну или несколько технологических операций (хранение, производство, обработку, преобразование и транспортировку энергоресурсов). Комбинаторный анализ

проводится путем изучения различных комбинаций состояний объектов и переходов между состояниями за период T. Цель состоит в том, чтобы найти подграф графа G, который включает пути с минимальными затратами на развитие ТЭК.

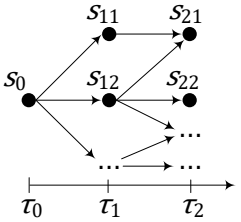


Рис. 4. Фрагмент графа G
Fig. 4. Fragment of the graph G

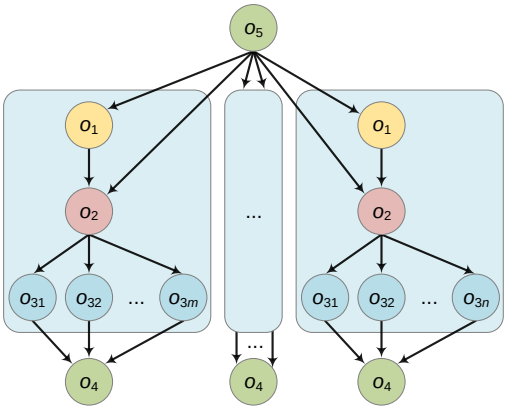


Рис. 5. Схема решения задачи
Fig. 5. Problem-solving scheme

Общая схема решения задачи приведена на рис. 5. Операция o_5 осуществляет подготовку данных сценариев развития ТЭК Вьетнама. Она формирует критерии оценки функционирования комплекса, список мероприятий по его развитию, множество организационно-функциональных ограничений, продолжительность моделирования и др.

Затем на основе сформированных данных операция o_1 выбирает сценарий и строит соответствующий ему вариант модели развития ТЭК. Операция o_2 производит декомпозицию модели на подмодели с учетом логических и балансовых ограничений. Подмодели параллельно исследуются с помощью экземпляров операции o_3. При этом решается следующая задача линейного программирования: $f(S) = CX + P(R - Y) \to min$, $S =< A, X, Y, C, P, D, R, U, W >$, $AX - Y \geq 0$, $X \leq D$, $Y \leq R$, где S – обобщенное состояние ТЭК, C – вектор удельных затрат по технологическим способам работы действующих, реконструируемых или модернизируемых, а также вновь сооружаемых объектов ТЭК, A – матрица технологических коэффициентов производства (добычи, переработки, преобразования) и транспорта ресурсов, $a_{ij} = q_{ij}(U, W)$, D – вектор, определяющий технически возможные интенсивности применения технологических способов работы объектов (производства и транспорта ресурсов), $d_i = h_i(U)$, R – вектор, задающий требуемые объемы потребления ресурсов, P – вектор удельных ущербов из-за дефицита ресурсов, U – вектор мероприятий по развитию ТЭК, W – вектор возмущений. Выражения CX и $P(R - Y)$ в целевой функции соответственно отражают издержки, связанные с работой комплекса, и ущерб от недопоставки ресурсов потребителям вследствие воздействия возмущения. В качестве возмущений рассматриваются такие чрезвычайные ситуации, как крупномасштабные технологические аварии, стихийные бедствия и террористические акты.

Операция o_4 агрегирует результаты моделирования, выполняет их многокритериальный анализ и формирует множество рациональных путей развития ТЭК для каждого сценария.

Модули $m_1 - m_5$ реализуют операции $o_1 - o_5$. Их частая модификация обусловлена развитием алгоритмических знаний РППП. В частности, она обуславливается применением новых критериев оценки работы комплекса, расширением списка мероприятий по его развитию и использованием дополнительных организационно-функциональных ограничений. Кроме того, развитие программно-аппаратного обеспечения среды зачастую приводит к необходимости перекомпиляции, сборки и тестирования модулей с учетом этих

обновлений. В связи с этим рассматриваются две базовые версии v_1 [9] и v_2 [10] пакета для решения задачи поиска направлений устойчивого развития ТЭК Вьетнама. Вторая версия разработана с использованием нового менеджера ПО ОТ.

Вычислительные ресурсы среды. Решение задач с помощью обеих вышеупомянутых версий пакета выполнялось в интегрированной вычислительной среде, включающей следующие ресурсы: облачные виртуализированные ресурсы под управлением OpenStack – 5 выделенных узлов (2x16 cores CPU AMD Opteron 6276, 2.3 GHz, 64 GB RAM) высокопроизводительного кластера ЦКП «Иркутский суперкомпьютерный центр СО РАН» (ИСКЦ); невыделенные ресурсы – узлы (2×18 cores CPU Intel Xeon X5670, 2.1 GHz, 128 GB RAM) высокопроизводительного кластера ЦКП ИСКЦ; грид-ресурсы – удаленные ПК. Выделенные ресурсы предназначены только для решения рассматриваемой задачи. Невыделенные ресурсы являются общими для всех пользователей ЦКП. Они используются в соответствии с установленными для них административными политиками и квотами. Пользователи, которым уже выделены ресурсы ЦКП, могут использовать только те невыделенные ресурсы, на которых есть свободные слоты в расписании обслуживания СУПЗ. В процессе решения задачи в интегрированной вычислительной среде ее грид-ресурсы могут использоваться их владельцами.

Подготовка эксперимента. Модификация ПО включает следующие этапы:

1) модификацию исходного кода прикладного и системного ПО, включая поиск его нужной версии в репозиториях на сервере GitLab, клонирование репозитория на сервер ОТ или сервер разработчика, запуск среды программирования, редактирование кода и сохранение модифицированного кода на сервер GitLab;
2) сборку ПО в соответствии со сценариями, предопределенными в конструкторе концептуальной модели для внешних систем непрерывной интеграции;
3) модификацию тестовых данных, включая их поиск на сервере GitLab или сервере тестирования, запуск системы обработки данных и их редактирование;
4) модификацию сценариев непрерывной интеграции, доставки и развертывания ПО с помощью конструктора концептуальной модели;
5) тестирование ПО в соответствии с предопределенными сценариями;
6) построение установочных пакетов ПО в соответствии со сценариями его непрерывной интеграции, доставки и развертывания;
7) модификацию концептуальной модели (параметров, операций, модулей, продукций, схем решения задач и связей между ними) среды, трансляцию полученной модификации в XML и проверку его полноты и корректности;
8) размещение ПО на ресурсах с помощью установочных пакетов;
9) тестирование ПО на ресурсах, включая оценку их производительности;
10) тестирование схем решения задач РППП по предопределенным сценариям.

Как правило, данные этапы многократно повторяются в процессе модификации ПО. При разработке версии v_1 все они выполнялись в неавтоматизированном режиме. Оценки средних временных затрат в минутах на модификацию ПО пакета показаны на рис. 6. Они включают оценки затрат общего времени (рис. 6 а и 6 в) и разработчика (рис. 6 б и 6 г) на этапах модификации одного модуля с использованием менеджера ПО (версия v_2) и без него (версия v_1). Рис. 6 а и 6 в отражают временные затраты на этапах, связанных с модификацией алгоритмических знаний. Их выполнение базируется на совместном использовании средств ОТ и внешних систем непрерывной интеграции. Остальные оценки, относящиеся к модификации пакета только средствами ОТ, показаны на рис. 6 б и 6 г.

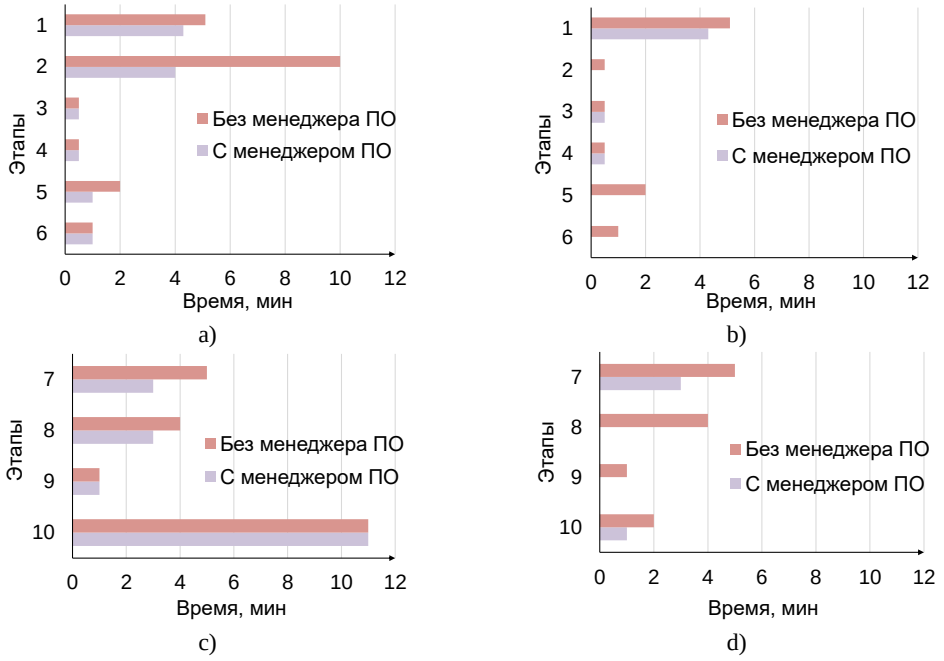


Рис. 6. Оценки затрат общего времени (a, b) и разработчика (c, d) на этапах модификации ПО
Fig. 6. Evaluations of the total time overheads (a, b) and developer (c, d) at the stages of software modification

Результаты на рис. 6, показывают, что применение менеджера ПО обеспечивает существенное сокращение как временных затрат разработчика, так и общих временных затрат на подготовку эксперимента. Они обусловлены следующими факторами: сокращением времени поиска исходного кода ПО в репозиториях; автоматизацией сборки, тестирования, построения установочных пакетов, размещения и тестирования ПО на ресурсах, тестирования схем решения задач; сокращением времени сборки, тестирования и построения установочных пакетов за счет переноса выполнения этих этапов на более производительные ресурсы; повышением надежности вычислений в процессе решения задачи путем предварительного тестирование ПО на разнородных ресурсах; автоматизацией процесса модификации концептуальной модели среды при изменении ПО в репозиториях.

На рис. 7 показаны оценки временных затрат разработчика ПО и общих временных затрат в минутах на подготовку и проведение экспериментов для версии v_2 с менеджером ПО и без него. Здесь 11-й этап – это проведение эксперимента. Оценки времени даны нарастающим итогом в соответствии с порядком выполнения этапов. Очевидно, что применение менеджера ПО обеспечивает сокращение времени подготовки и проведения экспериментов.

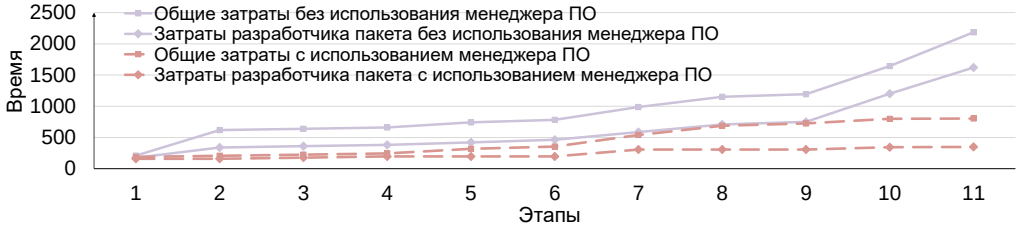


Рис. 7. Оценки временных затрат
Fig. 7. Time overheads evaluations

Вычислительный эксперимент. Мы сравниваем показатели выполнения модулей двух версий v_1 и v_2 при решении типовых задач с одинаковой вычислительной сложностью. При этом использованы данные одной и той же структуры. Они получены на основе статистической информации о работе ТЭК Вьетнама в разные периоды. В табл. 2 приведены следующие показатели решения задачи для одного сценария развития ТЭК: n_s – число запусков модуля, t_{avg} – среднее время выполнения модуля в секундах, n_f – число отказов модуля, n_r – число рестартов модуля, k – коэффициент полезного использования компонентов узла.

Табл. 2. Показатели выполнения модулей
Table 2. Module execution parameters

Модуль / Module	n_s		t_{avg}			n_f		n_r		k
	v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2	v_1	v_2
m_1	1	1	10.22	10.17	0	0	0	0	90.01	90.70
m_2	1	1	172.64	166.42	0	0	0	0	94.35	94.74
m_3	531442	531497	10.17	9.74	1371	0	1678	0	88.27	98.46
m_4	1	1	130.04	129.23	0	0	0	0	30.00	30.02
m_5	1	1	11.62	10.55	0	0	0	0	27.27	30.63

Время параллельной обработки подмоделей развития ТЭК экземплярами модуля m_3 составляет основную долю общего времени решения задачи. Результаты, приведенные в табл. 2, показывают, что число отказов экземпляров модуля m_3 версии v_2, сократилось. Соответственно, число их рестартов уменьшилось. Оптимизация работы модуля m_3 версии v_2 и устранение отказов в его работе привело к улучшению коэффициента полезного использования ресурсов данным модулем на 10.19%. Коэффициенты полезного использования ресурсов при выполнении остальных модулей версии v_2 так же улучшены.

На рис. 8 приведено время t_2 и t_3 вычислений в минутах при использовании модулей версий v_1 и v_2. В обоих случаях совместно использовались выделенные и невыделенные ресурсы. Распределение заданий выполнялось под управлением MAC.

Дополнительно приведено оценочное время t_1 решения задачи только на выделенных ресурсах при использовании модулей версии v_1. В этом случае распределение заданий осуществлялось под управлением OpenStack. В случае версии v_1 использование дополнительных невыделенных ресурсов позволило сократить время решения задачи на 18% по сравнению с оценочным временем ее решения только на невыделенных ресурсах. Применение версии v_2 обеспечило сокращение этого времени еще на 5%.

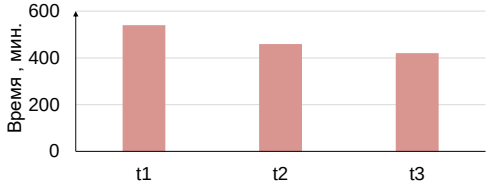


Рис. 8. Время решения задачи
Fig. 8. Problem-solving time

7. Заключение

ОТ предоставляет возможность организовывать унифицированную среду, интегрирующую облачные платформы, грид-системы и суперкомпьютерные ЦКП. Он включает средства для

разработки и использования РППП с использованием современных методов и средств параллельных и распределенных вычислений, а также их виртуализации.

В статье представлены новые возможности инструментального комплекса ОТ, который является основой интегрированной вычислительной среды для разработки и применения РППП. Новая подсистема непрерывной интеграции, доставки и развертывания прикладного и системного программного обеспечения нацелена на сокращение временных затрат на подготовку эксперимента, повышение надежности распределенных вычислений и ускорение процесса решения задачи с учетом неопределенности.

Эти цели достигаются путем смягчения неопределенности в распределении заданий по ресурсам за счет извлечения дополнительных необходимых знаний о времени выполнения модуля в разных узлах гетерогенной среды, классификации заданий в соответствии с их характеристиками, использования знаний администраторов среды о соответствии ресурсов классам заданий, а также сведений, получаемых в процессе непрерывной интеграции, доставки и развертывания программного обеспечения. Преимущества ее применения показаны на примере разработки двух версий РППП для решения важных практических задачи исследования направлений устойчивого развития ТЭК Вьетнама.

Список литературы / References

[1]. Il'in V. Artificial Intelligence Problems in Mathematical Modeling. Communications in Computer and Information Science, vol. 1129, 2019, pp. 505-516.

[2]. Wang L., Jie W., Chen J. Grid computing: infrastructure, service, and applications. CRC Press, 2018, 528 p.

[3]. Varshney S., Sandhu R., Gupta P.K. QoS Based Resource Provisioning in Cloud Computing Environment: A Technical Survey. Communications in Computer and Information Science, vol. 1046, 2019, pp. 711-723.

[4]. Б.М. Шабанов, О.И. Самоваров. Принципы построения межведомственного центра коллективного пользования общего назначения в модели программно-определяемого ЦОД. Труды ИСП РАН, том 30, вып. 6, 2018 г., стр. 7-24. DOI: 10.15514/ISPRAS-2018-30(6)-1 / B.M. Shabanov and O.I. Samovarov. Building the Software-Defined Data Center. Programming and Computer Software, vol. 45, no. 8, 2019, pp. 458-466.

[5]. Mateescu G., Gentzsch W., Ribben C.J. Hybrid computing – where HPC meets grid and cloud computing // Future Generation Computer Systems, 2011, vol. 27, no. 5, pp. 440-453. DOI: 10.1016/j.future.2010.11.003.

[6]. Feoktistov A., Gorsky S., Sidorov I. et al. Orlando Tools: Energy Research Application Development through Convergence of Grid and Cloud Computing. Communications in Computer and Information Science, vol. 965, 2019, pp. 289-300.

[7]. Feoktistov A., Kostromin R., Sidorov I., Gorsky S. Development of Distributed Subject-Oriented Applications for Cloud Computing through the Integration of Conceptual and Modular Programming // In Proc. of the 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, 2018, pp. 256-261.

[8]. Yu J., Buyya R. A taxonomy of workflow management systems for grid computing. Journal of Grid Computing, vol. 3, no. 3-4, 2005, pp. 171-200.

[9]. Feoktistov A., Sidorov I., Tchernykh A. et al. Multi-Agent Approach for Dynamic Elasticity of Virtual Machines Provisioning in Heterogeneous Distributed Computing Environment In Proc. of the International Conference on High Performance Computing and Simulation (HPCS-2018), 2018, pp. 909-916.

[10]. Bychkov I., Oparin G., Feoktistov A. et al. Subject-oriented computing environment for solving large-scale problems of energy security research. Journal of Physics: Conference Series, vol. 1368, 2019, pp. 052030-1-052030-12.

[11]. Burri A., Dedner A., Klofkorn R., Ohlberger M. An efficient implementation of an adaptive and parallel grid in DUNE. Notes on Numerical Fluid Mechanics and Multidisciplinary Design, vol. 91, 2006, pp. 67-82.

[12]. Radchenko G., Hudyakova E. A service-oriented approach of integration of computer-aided engineering systems in distributed computing environments. In Proc. of the UNICORE Summit, 2012, pp. 57-66.
[13]. Shamakina A. Brokering service for supporting problem-oriented grid environments. In Proc. of the UNICORE Summit, 2012, pp. 67-75.
[14]. Bungartz H.J., Neumann P., Nagel W.E. Software for Exascale Computing-SPPEXA 2013-2015. Lecture Notes in Computational Science and Engineering, vol. 113, 2016, 565 p.
[15]. Afgan E., Baker D., Batut B. et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. Nucleic Acids Research, vol. 46, no. W1, 2018, pp. W537-W544.
[16]. Ananthakrishnan R., Blaiszik B., Chard K., Chard R. Globus platform services for data publication. In Proc. of the Practice and Experience on Advanced Research Computing, 2018, pp. 1-7.
[17]. Sukhoroslov O. Supporting Efficient Execution of Workflows on Everest Platform. Communications in Computer and Information, vol. 1129, 2019, pp. 713-724.
[18]. Gavvala S.K., Chandrasheka J., Gangadharan G.R., Buyya R. QoS-aware cloud service composition using eagle strategy. Future Generation Computer Systems, vol. 90, 2019, pp. 273-290.
[19]. Deelman E., Peterka T., Altintas I., Carothers C.D. The future of scientific workflows. The International Journal of High Performance Computing Applications, vol. 32, no. 1, 2018, pp. 159-175.
[20]. Wangsom P., Lavangnananda K., Bouvry P. Multi-Objective Scientific-Workflow Scheduling with Data Movement Awareness in Cloud. IEEE Access, vol. 7, 2019, pp. 177063-177081.
[21]. Feoktistov A., Gorsky S., Sidorov I., Tchernykh A. Continuous Integration in Distributed Applied Software Packages. In Proc. of the 42st International Convention on Information and Communication Technology, Electronics and Microelectronics, 2019, pp. 1775-1780.
[22]. Gruver G. Start and Scaling Devops in the Enterprise. BookBaby, 2016. 100 p.
[23]. Talia D. Workflow Systems for Science: Concepts and Tools. ISRN Software Engineering, 2013, vol. 2013, Article ID 404525.
[24]. Deelman E., Vahi K., Juve G et al. Pegasus, a workflow management system for science automation // Future Generation Computer Systems, vol. 46, 2015, pp. 17-35.
[25]. Bumgardner V.K. OpenStack in Action. Manning Publications, 2016. 384 p.
[26]. Hirales-Carbajal A., González-García J. L., Tchernykh A. Workload Generation for Trace Based Grid Simulations. In Proc. of the 1st International Supercomputer Conference in Mexico (ISUM–2010), 2010, pp. 1-10.
[27]. Bychkov I., Oparin G., Tchernykh A. et al. Conceptual Model of Problem-Oriented Heterogeneous Distributed Computing Environment with Multi-Agent Management. Procedia Computer Science, vol. 103, 2017, pp. 162-167.
[28]. Соколинский Л.Б., Шамакина А.В. Методы управления ресурсами в проблемно-ориентированных вычислительных средах. Программирование, том 42, no. 1, 2016 г., стр. 26-38 / Sokolinsky L.B., Shamakina A.V. Methods of resource management in problem-oriented computing environment. Programming and Computer Software, vol. 42, no. 1, 2016, pp. 17-26.
[29]. Ramírez-Velarde R., Tchernykh A., Barba-Jimenez C. et al. Adaptive Resource Allocation with Job Runtime Uncertainty Journal of Grid Computing, vol. 15, no. 4, 2017, pp. 415-434.
[30]. Tchernykh A., Schwiegelshohn U., Talbi E.-G., Babenko M. Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability. Journal of Computational Science, vol. 36, 2019, 100581. DOI: 10.1016/j.jocs.2016.11.011.
[31]. Kalyaev A.I., Kalyaev I.A. Method of multiagent scheduling of resources in cloud computing environments. Journal of Computer and Systems Sciences International, vol. 55, no. 2, 2016, pp. 211-221.
[32]. Bychkov I.V., Oparin G.A., Feoktistov A.G. et al. Multiagent control of computational systems on the basis of meta-monitoring and imitational simulation. Optoelectronics, Instrumentation and Data Processing, vol. 52, no. 2, 2016, pp. 107-112.
[33]. Java Agent DEvelopment Framework, available at: https://jade.tilab.com, accessed 30.05.2020.
[34]. Herrera J., Huedo E., Montero R., Llorente I. Porting of Scientific Applications to Grid Computing on GridWay. Scientific Programming, vol. 13, no. 4, 2005, pp. 317-331.
[35]. Tannenbaum T., Wright D., Miller K., Livny M. Condor – A Distributed Job Scheduler. In Beowulf Cluster Computing with Linux. The MIT Press, 2002, pp. 307-350.
[36]. Feoktistov A., Tchernych A., Kostromin R., Gorsky S. Knowledge Elicitation in Multi-Agent System for Distributed Computing Management. In Proc. of the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, 2017, pp. 1350-1355.

[37]. Feoktistov A., Kostromin R., Sidorov I. et al. Multi-Agent Algorithm for Re-Allocating Grid-Resources and Improving Fault-Tolerance of Problem-Solving Processes. Procedia Computer Science, vol. 150, 2019, pp. 171-178.

[38]. Vickrey W. Counterspeculation, Auctions, and Competitive Sealed Tenders. Journal of Finance, vol. 16, no. 1, 1961, pp. 8-37.

Информация об авторах / Information about authors

Андрей Николаевич ЧЕРНЫХ получил степень кандидата наук в Институте точной механики и вычислительной техники РАН. Он является профессором CICESE. В научном плане его интересуют многоцелевая оптимизация распределения ресурсов в облачной среде, проблемы безопасности, планирования, эвристики и метаэвристики, интернет вещей и т.д.

Andrei Nikolaevitch TCHERNYKH received his PhD degree at the Institute of Precision Mechanics and Computer Engineering of the Russian Academy of Sciences. He is holding a full professor position in computer science at CICESE. He is interesting in grid and cloud research addressing multiobjective resource optimization, both, theoretical and experimental, security, uncertainty, scheduling, heuristics and meta-heuristics, adaptive resource allocation, and Internet of Things.

Игорь Вячеславович БЫЧКОВ – академик РАН, доктор технических наук, профессор, директор Института динамики систем и теории управления им. В.М. Матросова СО РАН (ИДСТУ СО РАН). Области исследования: искусственный интеллект, геоинформационные системы, веб-технологии, математическое моделирование и облачные вычисления.

Igor Vyacheslavovich BYCHKOV – Academician of RAS, Ph.D., Professor, Director of the Matrosov Institute for System Dynamics and Control Theory of SB RAS (ISDCT SB RAS). Fields of research: artificial intelligence, geoinformation systems, web-technologies, mathematical modeling, and cloud computing.

Александр Геннадьевич ФЕОКТИСТОВ – кандидат технических наук, доцент, заведующий лабораторией параллельных и распределенных вычислительных систем ИДСТУ СО РАН. Области исследования: распределенные пакеты программ, мультиагентные технологии.

Alexander Gennadevich FEOKTISTOV – Ph.D., Associate Professor, Head of the Laboratory of Parallel and Distributed Computing Systems of ISDCT SB RAS. Fields of research: distributed software packages and multi-agent technologies.

Сергей Алексеевич ГОРСКИЙ – кандидат технических наук, научный сотрудник ИДСТУ СО РАН. Сфера научных интересов: параллельные и распределенные вычисления, математическое моделирование и сервис-ориентированное программирование.

Sergei Alexeevich Gorsky – Ph.D., Research Officer of ISDCT SB RAS. Research interests: parallel and distributed computing, mathematical modeling, and service-oriented programming.

Иван Александрович СИДОРОВ – кандидат технических наук, научный сотрудник ИДСТУ СО РАН. Его научные интересы включают системное администрирование, параллельные и распределенные вычисления, агенты и мониторинг.

Ivan Alexandrovich SIDOROV – Ph.D., Research Officer of ISDCT SB RAS. His research interests include system administration, parallel and distributed computing, agents, and monitoring.

Роман Олегович КОСТРОМИН – младший научный сотрудник ИДСТУ СО РАН. Область исследования: мультиагентные средства управления распределенными вычислениями.

Roman Olegovich KOSTROMIN – Junior Researcher ISDCT SB RAS. Field of research: multi-agent tools for distributed computing management.

Алексей Владимирович ЕДЕЛЕВ – кандидат технических наук, старший научный сотрудник Института систем энергетики им. Л.А. Мелентьева СО РАН. Его научные интересы включают моделирование развития энергетики и системы поддержки принятия решений.

Aleksey Vladimirovich EDELEV – Ph.D. of Engineering Sciences, Senior Researcher of the Melentiev Energy Systems Institute of SB RAS. His research interests include energy development modeling and decision-making support systems.

Валерий Иванович ЗОРКАЛЬЦЕВ – доктор технических наук, профессор, главный научный сотрудник Лимнологического института СО РАН. Области исследования: теория оптимизации, математическая экономика и методика моделирования.

Valeriy Ivanovich ZORKALTSEV – Doctor of Technical Sciences, Professor, Chief Researcher of the Limnological Institute of SB RAS. Field of research: optimization theory, mathematical economics, and simulation technique.

Арутюн Ишханович АВЕТИСЯН – академик РАН, доктор физико-математических наук, профессор РАН, директор ИСП РАН, заведующий кафедрами системного программирования ВМК МГУ, МФТИ и факультета компьютерных наук ВШЭ. Сфера научных интересов: анализ и трансформация программ, безопасность программного обеспечения, технологии параллельных и распределенных вычислений.

Arutyun I. AVETISYAN – Academician of RAS, Doctor of Physics and Mathematics, Professor of RAS, Director of the ISP RAS, Head of the Departments of System Programming, the faculty of Computational Mathematics and Cybernetics of Lomonosov Moscow State University, Moscow Institute of Physics and Technology, and the Faculty of Computer Science at the Higher School of Economics. Research interests: program analysis and transformation, software security, parallel and distributed computing.

DOI: 10.15514/ISPRAS–2021–33(1)–12

Путеводитель по проектированию удобных Web-API

М. Тельо-Родригес, ORCID: 0000-0002-7424-9912 <maritello797@gmail.com>
Х.О. Очаран-Эрнандес, ORCID: 0000-0002-2598-1445<jocharan@uv.mx>
Х.К. Перес-Арриага, ORCID: 0000-0003-2354-2462 <juaperez@uv.mx>
К. Лимон, ORCID: 0000-0003-4654-636X <hlimon@uv.mx>
А.Х. Санчес-Гарсия, ORCID: 0000-0002-2917-2960 <angesanchez@uv.mx>
Университет Веракрузана
Мексика, 91000, Веракрус, Халапа

Аннотация. Направления развития облачных вычислений, такие как Software as a Service (SaaS), позволяют поставщикам размещать сложные приложения через Интернет, делая их доступными для внешних потребителей через интерфейсы прикладного программирования (API). Успех SaaS, как и в некотором смысле любой распределенной системы, во многом зависит от ее API. Наличие очень удобных в использовании API повышают эффективность и качество процесса разработки, хотя, конечно для программистов остаются существенными и другие аспекты API. Различные исследования показывают, что в процессе разработки API наиболее подходящим для решения проблем удобства использования является этап проектирования. При проектировании API удобство использования должно являться явным критерием качества. В настоящей статье мы предлагаем путеводитель по проектированию Web-API с акцентом на удобство использования, опираясь на лучшие методы проектирования удобных Web-API. Наш путеводитель по проектированию основано на адаптации методологии *проектного подхода к исследованиям (*Design Science Research Methodology, DSRM) и дополнено систематическим обзором литературы, а также анализом серой литературы по методам, методикам и инструментам, используемым для разработки удобных API.

Ключевые слова: интерфейс прикладного программирования; API; Web-API; удобство использования API; путеводитель по проектированию

Для цитирования: Тельо-Родригес М., Очаран-Эрнандес Х.О., Перес-Арриага Х.К., Лимон К., Санчес-Гарсия А.Х. Путеводитель по проектированию удобных Web-API. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 173-188. DOI: 10.15514/ISPRAS–2021–33(1)–12

Благодарности. Авторы выражают признательность за поддержку и финансирование, предоставленные PRODEP 2018 (Programa para el Desarrollo Profesional Docente en Educación Superior, 48058 511-6/18-9245/PTC-888).

A Design Guide for Usable Web APIs

M. Tello-Rodríguez, ORCID: 0000-0002-7424-9912 <maritello797@gmail.com>
J.O. Ocharán-Hernández, ORCID: 0000-0002-2598-1445 <jocharan@uv.mx>
J.C. Pérez-Arriaga, ORCID: 0000-0003-2354-2462 <juaperez@uv.mx>
X. Limón, ORCID: 0000-0003-4654-636X <hlimon@uv.mx>
Á.J. Sánchez-García, ORCID: 0000-0002-2917-2960 <angesanchez@uv.mx>
Universidad Veracruzana,
Xalapa, Veracruz, México, 91000

Abstract. Cloud computing trends such as Software as a Service (SaaS) enable providers to host complex applications over the Internet, making them available to external consumers through an Application Programming Interface (API). The success of a SaaS, and in some sense any distributed system, is greatly influenced by its API. Highly usable APIs improve the efficiency of the development process and its quality, ensuring that programmers continue to appreciate other aspects of the API while increasing their productivity. Different studies state that the design phase within the development process of an API is the most appropriate to address usability issues. Therefore, usability should be considered as an explicit criterion in the design of an API. In this paper, we propose a design guide for web APIs with an emphasis on usability, using the best practices of usable web APIs design. Our design guide is based on an adaptation of the design science research methodology (DSRM), and it is complemented with a systematic literature review and gray literature analysis concerning methods, techniques, and tools used to develop usable APIs.

Keywords: Application Programming Interface; API; Web API; API Usability; Design Guide.

For citation: Tello-Rodríguez M., Ocharán-Hernández J.O., Pérez-Arriaga J.C., Limón X., Sánchez-García Á.J. A Design Guide for Usable Web APIs. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 173-188 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–12.

Acknowledgments. The authors are grateful for the support and funding granted by PRODEP 2018 (Programa para el Desarrollo Profesional Docente en Educación Superior, 48058 511-6/18-9245/PTC-888).

1. Введение

Web-API обеспечивают эффективный и расширяемый подход для взаимодействия приложений [1] в гетерогенных и распределенных средах, таких как облачные вычисления. В качестве интерфейсов API-интерфейсы позволяют пользователю (разработчику клиентской части приложения) воспринимать программы, контролировать их и взаимодействовать с ними [2]. Точно так же, как в любом пользовательском интерфейсе отражается опыт пользователей по взаимодействию с приложениями, в API отражается опыт разработчиков приложений. Удобство использования является абсолютно необходимым минимумом для того, чтобы разработчик использовал данный API [5].

В своей работе мы руководствовались определением удобства использования, представленеым в стандарте ISO / IEC 25010 [6] (который заменил стандарт ISO / IEC 9126). Стандарт определяет удобство использования как подмножество модели качества в использовании (quality-in-use). Эта модель определяется пятью характеристиками: результативность, производительность, удовлетворенность, покрытие контекста и свобода от риска. Удобство использования является подмножеством этих характеристик и включает следующие три характеристики:

1) *результативность* отражает возможность для конкретных пользователей использовать систему в определенном контексте и достигать своих целей полностью и точно.
2) *производительность* оценивает, как конкретный пользователь использует ресурсы для эффективного достижения своих целей.
3) *удовлетворенность* отражает уровень удовлетворенности пользователей использованием системы в определенном контексте; эта вспомогательная характеристика далее делится на четыре атрибута: *полезность* (когнитивное

удовлетворение), *доверие* (уверенность в поведении системы), *удовольствие* (эмоциональное удовлетворение) и *комфорт* (физическое удовлетворение).

В ISO / IEC 25010 определяется вторая модель качества, в которой удобство использования классифицируется как одна из восьми характеристик качества, и определяется как «степень, в которой программный продукт может успешно пониматься, изучаться, применяться пользователем и являться для него наиболее привлекательным при использовании в соответствии с требованиями». Внутренние и внешние свойства удобства использования моделируются с помощью следующих шести подхарактеристик [7]:

1) определимость пригодности;
2) изучаемость;
3) управляемость;
4) защищенность от ошибок пользователя;
5) эстетика пользовательского интерфейса;
6) доступность.

Биль (Matthias Biehl) в [3] отмечает, что удобство использования является желательным свойством любого API, включающим в себя такие характеристики, как ориентация на потребителя, простота, понятность без дополнительных пояснений, интуитивность и предсказуемость. Сторонники удобства использования API, такие как Джошуа Блох (Joshua Bloch) из Google, подчеркивают, что «хорошие API повышают удовольствие от работы и продуктивность разработчиков […], а также качество программного обеспечения, создаваемое с их использованием» [9].

По мнению Хеннинга (Michi Henning) [10], последствиями применения плохо спроектированных API являются снижение надежности программного обеспечения и продуктивности программиста, использующего API. Проблемы с удобством использования могут вызвать ненужную сложность в коде клиентской части приложений и, следовательно, во всей распределенной системе.

Как следует из литературы, во всем процессе разработки наиболее подходящим для решения проблемы удобства использования является этап проектирования API [12, 13]. Кроме того, удобство использования должно быть добавлено как явный критерий проектирования и оценки, чтобы исключить в ходе разработки возможность случайного создания API, неудобного для использования [11].

Нашей целью является разработка высокоуровневого руководства по проектированию Web-API на основе мирового опыта, артефактов проектирования и методов повышения удобства использования Web-API. Мы предлагаем не строгую пошаговую процедуру, а скорее набор общих рекомендаций, которые разработчики API могут применять независимо от того, какую методологию или процедуру разработки.

В своей работе мы используем адаптацию методологии проектного подхода к исследованиям (Design Science Research Methodology, DSRM) [14], которая включает пять этапов. На сегодняшний день мы сделали следующее: путем систематического обзора литературы (systematic literature review, SLR) и анализа серой литературы определили методы, приемы и инструменты, используемые для разработки удобных для использования API; сравнили и смоделировали процессы разработки API, а также выбрали и интегрировали эти элементы для разработки первой версии путеводителя по проектированию. В будущем мы намерены завершить демонстрацию и оценку руководства, выполнив специальное исследование, в котором с использованием предлагаемого руководства будет спроектирован реальный Web-API, и удобство его использования будет оцениваться с помощью различных методов. С отчетом по проведенным нами SLR и обзору серой литературы можно познакомиться в [15], настоящая статья является продолжением указанной работы.

Статья организована следующим образом. В разд. 2 приводится обзор некоторых процессов разработки и проектирования, ориентированных на удобство использования и восприятие

пользователями (User Experience, UX). В разд. 3 представлен метод разработки нашего руководства по проектированию API, ориентированного на удобство использования, на основе методологии, предложенной Пефферсом (Ken Peffers) и др. [14]. Разд. 4 описывает предлагаемое руководство по проектированию и следующую из него модель. Наконец, в разд. приводятся заключение и описываются направления будущей работы.

2. Родственные работы

Несколько авторов представили исследования, которые затрагивают вопросы обеспечения удобства использования API во время разработки, но мало кто пытался это сделать на этапе проектирования. В проанализированной литературе мы нашли методы, процессы, руководства, описание опыта, эвристики, артефакты, методы и инструменты, в которых прямо или косвенно учитывается удобство использования. Ниже в этом разделе некоторые из этих работ детализированы, начиная от общих процессов разработки и заканчивая конкретными артефактами и методами, используемыми в различных проектах.

Авторы работы [5] описывают ориентированный на пользователя процесс, в котором учитываются удобство использования и другие аспекты, такие как безопасность и масштабируемость. Биль [3] описывает полный процесс разработки, но без упоминания практических аспектов удобства использования. С другой стороны, Хантер [16] описывает дизайн-ориентированную методологию, с упором на учет опыта разработчиков и RESTful API – Web-службы на основе архитектуры REST [17].

Кроме того, мы нашли руководства, в которых описываются процессы разработки и проектирования, специально предназначенные для повышения удобства использования API. В 2008 году Стилос (Jeffrey Stylos) и др. в [18] описали свою работу по редизайну, ориентированному на пользователя, для повышения удобства использования существующего API. В 2014 году Ли (Sunghoon Lee) и др. [19] предложили процесс проектирования в расчете на обеспечение удобства использования вместе с рекомендациями по оценке API, ссылаясь на критерии удобства использования, «простые в применении и количественно измеримые». Еще одним примером является проект API Craft на сайте Google Groups, в котором успех разработчика ставится выше любого другого принципа проектирования [4].

В отличие от упомянутых процессов разработки и проектирования, некоторые авторы ставят на передний план артефакты дизайна, практические приемы и методы, нацеленные на повышение удобства использования. Прототипирование – это один из наиболее рекомендуемых методов, поскольку прототипы могут использоваться для привлечения пользователей (программистов клиентских частей приложений) и получения обратной связи на ранних этапах, что помогает улучшить пользовательское восприятие [13, 18, 20].

Часто упоминаются методы проверки и оценки удобства использования. В [16] приемочные тесты, основанные на описании требований заказчиков, предлагаются для проверки того, что тестируемый вариант использования является простым, понятным и функциональным. Другим примером является «эвристическая оценка» Нильсена для проверки дизайна API. Хотя рекомендации Нельсона можно использовать как отдельное руководство по проектированию API, разработчикам API следует иметь их в виду, даже если они используют какой-либо другой подход [11].

Робиллард (Martin P. Robillard) и Делайн (Robert DeLine) [21] отмечают, что документирование API не может быть четко отделено от обязательств по проектированию API, даже если для этого требуются другие навыки. Документация часто используется, чтобы помочь заполнить разрыв между дизайном API и пониманием разработчика программного обеспечения того, как успешно использовать API [22]. Мы нашли несколько работ, посвященных документации API и удобству использования. Например, имеется исследование, авторы которого убеждают в том, что, следуя их рекомендациям по поводу

документирования REST API, разработчики API могут уменьшить количество ошибок, повысить уровень успеха и удовлетворенность разработчиков, использующих API [23].

Таким образом, мы смогли найти только исследования, в которых предлагаются руководства по проектированию и соответствующие процессы для повышения удобства использования API-интерфейсов, но не с упором на Web-API. Руководства по проектированию, которые включают Web-API, не предназначены для решения проблем удобства использования, но охватывают другие аспекты проектирования, например, документирование API. В отличие от этого, в настоящей работе представлено руководство по проектированию удобных Web-API, в котором собраны предложения авторов по улучшению удобства использования. Нашей целью является обеспечение полного набора, методов и артефактов проектирования, чтобы каждый разработчик API мог применить то, что лучше всего подходит для его проекта.

3. Метод исследования

Применяемый нами метод исследования был инспирирован методологией проектного подхода к исследованиям Пефферса и др. [14]. Авторы [14] утверждают, что методология работает как проводник для исследований в области науки проектирования (Design Science, DS), помогая в признании и легитимизации целей исследования, процессов и результатов, помогая исследователям представить свою работу. На рис. 1 показаны пять последовательных фаз используемого метода, фазы нашего исследования подробно описаны в последующих параграфах.

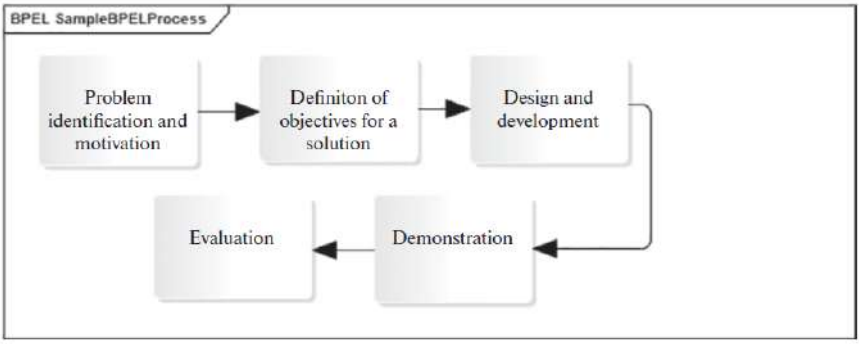


Рис.1. Этапы методологии, описанной Пефферсом и др.
Fig. 1. The phases in the methodology described by Peffers et al.

3.1 Выявление и мотивация проблемы

С целью выявления проблемы был выполнен обзор литературы с ориентацией на определение Web-API, удобство использования как атрибут качества, важность удобства использования API и отрицательное влияние неудобных API. Цель состояла в том, чтобы обосновать полезность рассмотрения удобства использования как значимого атрибута при проектировании Web-API.

Согласно информации, полученной при работе над обзором, проблему можно резюмировать следующим образом: на удобство использования API может влиять несколько факторов, таких как учебные ресурсы, контекст использования, технические ограничения и проектные решения на низком и высоком уровнях абстракции [13, 24, 25]. Пренебрежение удобством API в процессе не позволяет избежать аномалий, которые могут привести к снижению показателей надежности и соответствия требованиям [26]. Человеческий фактор имеет отношение и к разработчикам API, а отсутствие опыта разработки API может негативно влиять на конечный продукт. Кроме того, не хватает методологий, нацеленных на удобство использования, учитывая, что литература по этому вопросу технически неоднородна, и

многие авторы вообще не затрагивают или обсуждают только поверхностно некоторые аспекты удобства использования [20, 26].

3.2 Определение целей решения

Чтобы определить инструменты, методы и приемы, используемые для повышения удобства использования API на любом этапе разработки, мы выполнили систематический обзор литературы (SLR). Отчет об этой работе [15] (на испанском языке) был подготовлен на основе руководства, предложенного Барбарой Китченхэм (Barbara Kitchenham) и др. в [27], в котором описывается процесс систематических обзоров в области инженерии программного обеспечения..

Для достижения цели SRL мы сформулировали четыре исследовательских вопроса (Research Questions, RQ), которые перечислены в табл. 1. Эти RQ легли в основу создания поискового запроса, которая использовалась при поиске статей в семи базах данных: Scopus, Web of Science, Science Direct, Wiley, SpringerLink, IEEE Xplore Digital Library и ACM Digital Library.

Табл.1. Исследовательские вопросы для SRL
Table 1. Research questions for the SRL

№	*Вопрос*
RQ 1	Какие методы, практики, эвристики, техники, методологии или процессы, найденные в литературе, посвящены разработке удобных в использовании API?
RQ 2	Какие существуют инструменты, поддерживающие практики, техники, методы, методологии и процессы?
RQ 3	Каковы характеристики удобных в использовании API?
RQ 4	Какие существуют доказательства эффективности этих методов, техник, практик, методологий, процессов и инструментов?

По поисковому запросу было найдено 19112 статей. Мы применили к полученным статьям шесть критериев включения и исключения, в результате чего у нас осталось 65, а после удаления дубликатов мы сократили список до 42 статей. Для ознакомления с этим списком, а также с более подробным описанием процесса SLR мы адресуем читателя к нашей предыдущей публикации [15]. Из найденных работ были извлечены следующие данные:

- название;
- автор;
- год публикации;
- источник;
- ключевые слова;
- ответы на RQ;
- определяемые метод, техника, эвристика, практика и соответствующее описание;
- этапы и шаги метода, техники, эвристики или практики;
- желательные свойства в используемых API;
- характеристики неудобных API;
- рассмотренные аспекты удобства использования;
- конкретные примеры метода, техники, эвристики или практики;
- отношение инструмента к методу, технике или практике;
- проверяется или измеряется ли эффективность метода, техники, эвристики или практики, и каким образом?

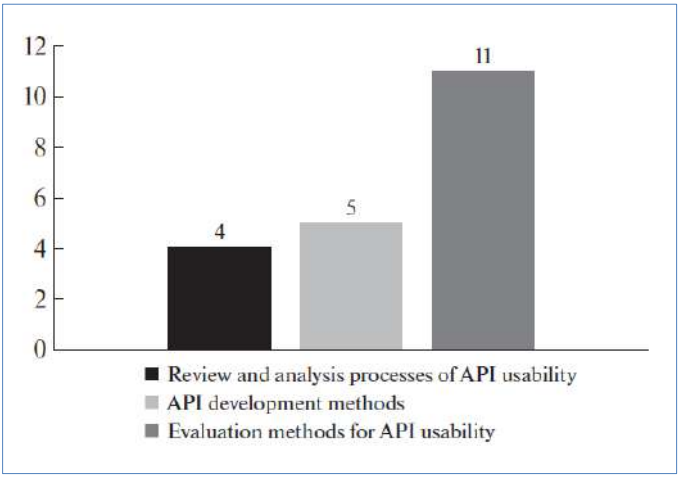


Рис. 2. Статьи, посвященные различным вопросам удобства использования API
Fig. 4. Papers addressing different API usability topics

На рис. 2 мы можем увидеть, что количество найденных методов оценки удобства использования API превышает количество найденных процессов обзора и анализа, а также методов разработки.

После извлечения данных был произведен синтез данных, следуя методу метаагрегации [29], который приводит к извлечению найденных результатов и их классификации по категориям. Всего извлечено 158 результатов. В итоге мы получили следующие четырнадцать категорий:

- определение удобства использования;
- факторы, влияющие на удобства использования;
- аспекты удобства использования;
- оценка и измерение удобства использования;
- проблемы удобства использования;
- удобные в использовании API:
  - важность и преимущества;
  - обычная практика;
  - характеристики и свойства;
  - методы разработки;
    - инструменты поддержки;
  - документация:
    - характеристики;
- неудобные API:
  - характеристики и свойства;
  - последствия.

Чтобы дополнить полученную информацию, был проведен обзор серой литературы с учетом книг и онлайн-ресурсов, ссылки на которые можно найти в [15]. Процесс обзора заключался в быстром чтении тематического указателя, аннотации и содержания каждого ресурса, чтобы определить, будет ли это интересно.

Учитывалось, упоминалось ли явно удобство использования или был ли описанный процесс разработки ориентирован на восприятие пользователей (User Experience, UX). После отбора ресурсов были прочитаны разделы, соответствующие нашему исследованию, и были извлечены существенные данные о самом процессе разработки и его артефактах, этапах, методах.

SLR помог также выявить различные методы оценки удобства использования, которые обычно применяются после некоторого этапа цикла разработки для получения обратной

связи по поводу уровне удобства использования API; например, метод пошагового разбора (API Walkthrough method) [30] или групповой экспертной оценки (Peer review API) [24, 31]. Оба указанных метода используются для обнаружения недостатков удобства использования при проектировании API. Мы также заметили, что в литературе почти не упоминаются инструменты, поддерживающие разработку удобных в использовании API.

3.3 Проектирование и разработка

Для проектирования и разработки нашего путеводителя была принята во внимание информация, полученная в упомянутых выше обзорах литературы. Из таблиц извлеченных данных (одна с данными SRL, другая с данными из серой литературы) наиболее актуальная и полезная информация была собрана в таблицу, в которой мы детализировали выявленные ранее методы, инструменты и артефакты.

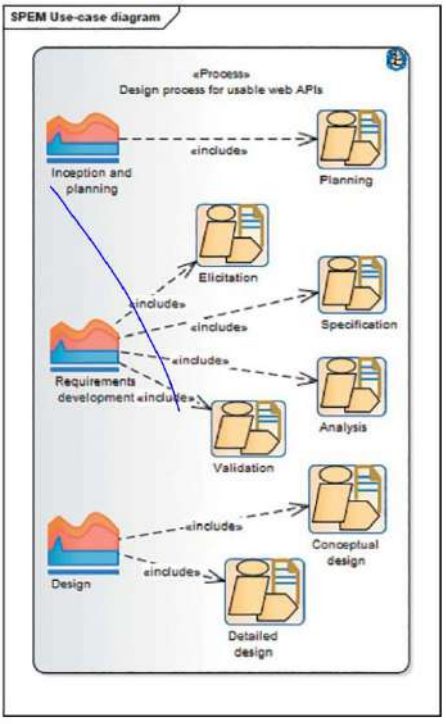


Рис. 3. Этапы создания и планирования, разработки требований и проектирования, а также соответствующие области деятельности
Fig 3. Phases of inception and planning, requirements development and design, and its corresponding areas of activity

3.4 Оценка и демонстрация

Для оценки и демонстрации мы планируем провести специальное исследование, которое будет включать разработку Web-API в соответствии с процессом, основанным на предлагаемом руководстве. Затем удобство использования разработанного API будет оцениваться с помощью различных методов анализа и оценки удобства использования, ранее выявленных в SRL. Можно использовать любой из следующих методов, все они применимы к проектированию API: метод пошагового разбора API [30], Apiness API design review [24], Think Aloud Evaluation [34], Cognitive Dimensions Framework [13, 26, 35, 36], групповые

экспертные оценки [31], метод Concept Maps [9] и исследовательский опросник (Research Questionnaire) [37].

4. Предлагаемое руководство по проектированию

В руководстве основное внимание уделяется представлению артефактов, методов, задач, инструментов и передовых практик, используемых в дизайне веб-API для повышения удобства использования конечного API, но также рассматриваются этапы начала/создания и планирования, а также разработки требований. Дизайнеры веб-API могут выбрать использование артефактов, методов, задач, инструментов и практик, которые они считают подходящими для своего проекта, поскольку не все из них могут быть применимы. Руководство призвано предоставить предложения, но не является методом разработки, поэтому мы не рекомендуем использовать его как таковой.

Для моделирования процессов, рассматриваемых в руководстве, была использована SPEM 2.0 (Software & Systems Process Engineering Metamodel, метамодель разработки программного обеспечения и систем), модель для определения конкретных программных процессов. На рис. 3 можно увидеть схему рассматриваемых этапов данного руководства и соответствующие области деятельности. К этим этапам мы обратимся отдельно в последующих подразделах.

4.1 Этап создания и планирования

Выявленные нами ориентированные на пользователя процессы с самого начала процесса разработки учитывают интересы клиентского программиста и отдают приоритет его нуждам, что помогает разработчикам API принимать решения, которые в конечном итоге приносят пользу тому же клиентскому программисту [5]. Поэтому в области планирования деятельности мы предлагаем принять на вооружение задачи и артефакты, перечисленные в табл. 2.

Табл. 2. Задачи и артефакты, предлагаемые на этапе создания и планирования
Table 2. Tasks and artifacts proposed within the phase of inception and planning

Название	Тип	Ссылка
Определение проблемы и степени воздействия	Задача	[38]
План	Артефакт	[38]
Функциональная спецификация	Артефакт	[16]
Определение бизнес-ценности	Задача	[16]
Установка метрик	Задача	[16]

Мы заметили, что некоторые из предложенных задач и артефактов тесно связаны друг с другом. Например, задача определения проблемы и степени ее воздействия может быть учтена в плане Web-API, и то же самое может произойти с оценкой бизнес-ценности.

Кроме того, Хантер [16] предлагает использовать метрики, которые могут быть связаны с удобством использования, такие как коэффициент внедрения. Этот же автор рассматривает функциональную спецификацию в рамках управления проектом и планирования, а не разработку требований, как мы обнаружили у большинства авторов.

4.2 Этап разработки требований

Фаза разработки требований включает в себя такие области деятельности, как выявление, анализ, спецификация и валидация. Из рис. 4 мы видим, насколько итеративным является этот процесс [39], и приходим к пониманию того, что предлагаемые задачи, артефакты и методы в этих областях деятельности можно использовать гибким образом.

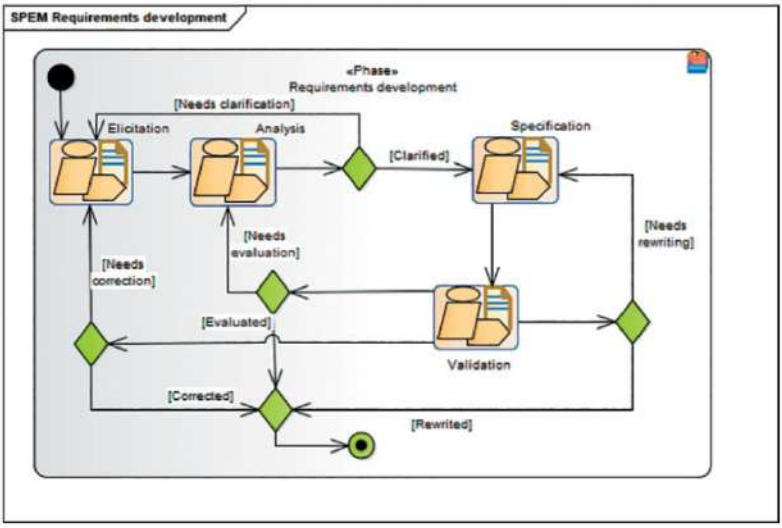


Рис. 4. Этап разработки требований и соответствующие области деятельности
Fig. 4. Phase of requirements development and its corresponding areas of activity

Несколько примеров артефактов и методов приведены в табл. 3, включая область деятельности для их предлагаемого использования. Картина целей – это артефакт, который может быть использован в двух областях деятельности – анализе и валидации; как и определение профилей программистов, картина целей может быть использована для извлечения и анализа информации. Методы создания сценариев использования и описания требований пользователей предложены несколькими авторами, что является логичным, поскольку эти методы помогают выявить потребности клиентских программистов, использующих API, и осуществлять проектирование с учетом их нужд [3, 13]. Предлагаемые методы сбора информации с помощью интервью и опросников помогают определить, что пользователи могут делать, как они это сделают, входные и выходные данные и другую важную информацию. Методы валидации особенно полезны для достижения работоспособности, подхарактеристики удобства использования в соответствии с ISO/IEC 25010, поскольку работоспособность включает в себя удовлетворение требований пользователя, учитывая все возможные сценарии [40]. Таким образом, методы и артефакты помогают лучше понять точку зрения программиста, использующего API, что является решающим для обеспечения удобства использования [41].

Табл. 3. Методы и артефакты, предлагаемые в рамках разработки требований
Table 3 Techniques and artifacts proposed within the phase of requirements development

Название	**Тип**	**Область деятельности**	**Ссылки**
Требуемые документы	Артефакт	Спецификация	[19]
Высокоуровневые сценарии использования	Метод	Анализ	[5, 20]
Интервью с заинтересованными сторонами	Метод	Сбор информации	[18, 20]
Пользовательские истории	Метод	Анализ	[3, 5, 13]
Картина целей	Артефакт	Анализ, валидация	[41]
Опросник	Метод	Сбор информации	[41]
Функциональная спецификация	Artifact	Спецификация	[16]

4.3 Этап проектирования

На этом этапе мы можем выявить места использования большинства найденных артефактов, методов, инструментов и даже руководств. На рис. 5 можно видеть области деятельности в рамках проектирования API, как это предлагается в литературе: все начинается с концептуального проекта, за которым следует его валидация; затем подготавливается детальный проект, после чего производится еще одна валидация. Несмотря на то, что поток работ между областями деятельности может казаться в некотором роде последовательным, этап проектирования на практике может быть очень гибким, и разработчики API могут вернуться к концептуальному проекту после выполнения деятельности по детальному проектированию.

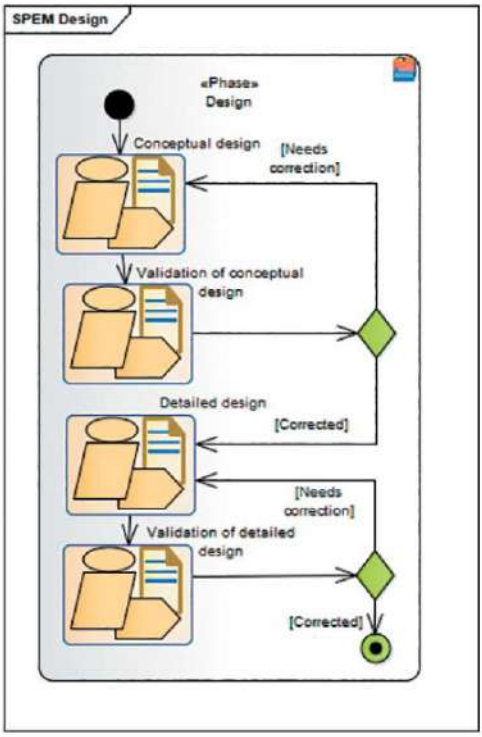


Рис. 5. Этап проектирования и соответствующие области деятельности
Fig 5. The phase of design and its corresponding areas of activity

В табл. 4 мы демонстрируем несколько техник, артефактов, рекомендаций и инструментов, используемых при проектировании Web-API для содействия в улучшении удобства использования результирующего API. Мы обнаружили, что инструмент для создания эскизов при концептуальном проектировании предложен только в одной работе [32], в то время как методы прототипирования предлагаются многими авторами и по-разному. Например, Рамакришнан (Lavanya Ramakrishnan) и др. [20] предлагают использовать нефункциональный прототип для получения обратной связи от пользователей. Джин (Brenda Jin), Сахни (Saurabh Sahni) и Шеват (Amir Shevat) [5] напротив, предлагают использовать реалистичные прототипы как подготовку к реализации, что связано с техникой валидации на основе приемочного тестирования: предполагается, что созданные прототипы будут тестироваться потребителями API. Другой метод валидации – это использование оценок удобства использования, таких как метод Think Aloud Evaluation, цель которого состоит в нахождении областей для улучшения проекта [18], или Cognitive Dimensions Framework, где несколько измерений могут использоваться для оценки различных проблем удобства

использования [13]. В рекомендациях Нильсона по «эвристической оценке» описываются десять свойств, которые разработчик может использовать для проверки своего проекта API, включая предотвращение ошибок, эстетику и эффективность использования. Заметим, что эти свойства являются подхарактеристиками удобства использования в соответствии с ISO/IEC 25010 [11].

Табл. 4. Методы, артефакты, рекомендации и инструменты, предлагаемые на этапе проектирования
Table 4. Techniques, artifacts, guidelines, and tools proposed within the phase of design

Название	Тип	Область деятельности	Ссылки
Эскиз	Метод, инструмент	Концептуальный дизайн	[32]
Прототипирование	Метод	Концептуальный дизайн	[3, 5, 18, 20]
Оценка удобства использования методом «think aloud”	Метод	Валидация	[18]
Руководства на основе критерия удобства использования	РуководствА	Концептуальный и детальный дизайн	[19]
Оценка удобства использования с помощью CDF	Метод	Валидация	[13]
Таблица ресурсов	Артифакт	Детальный дизайн	[41]
Блок-схема для проверки данных параметров	Артифакт	Детальный дизайн	[41]
Описание API	Артифакт, руководство	Концептуальный и детальный дизайн	[3, 5, 16, 20, 41, 42]
Тест приемки	Метод	Валидация	[16]
Рейтинги Нильсена по степени удобства использования	Метод	Валидация	[38]
Рекомендации по «эвристической оценке» Нильсена	Метод	Валидация	[11]

Один из наиболее упоминаемых артефактов – это описания API, использование которых полезно для общения с потребителями API и получения от них отзывов. Кроме того, этот артефакт служит для валидации, поскольку описание может использоваться при моделировании, чтобы можно было ответить на некоторые вопросы, связанные с удобством использования, такие как «является ли API по-прежнему простым в использовании?» или «это все еще небольшой, гибкий и удобный API, или же мы создали чудовищный API?» [42].

Мы ожидаем, что предлагаемое руководство может служить набором методов, задач, инструментов и артефактов, чтобы разработчики Web-API могли использовать некоторые из этих элементов в процессе своих собственных разработок для создания удобных в использовании веб-API.

5. Выводы и дальнейшая работа

В настоящей работе представлены результаты проекта, направленного на создание руководства по разработке Web-API с акцентом на удобство использования. Проведен систематический обзор литературы с целью выявления методов, практик, серий руководств,

артефактов, техник и инструментов, с ориентацией на разработку удобных в использовании API, а также обзор серой литературы в дополнение к результатам SLR. Из всех обнаруженных методов и процессов выбраны элементы, которые включены в первый вариант руководства. Документация не была рассмотрена в явной форме, учитывая, что наш подход был сосредоточен на артефактах для разработки самого API, а не сопроводительной документации, но важно признать документацию решающим фактором, влияющим на удобство использования API.

В будущем предполагается продолжить работу над этапами демонстрации и оценки метода, описанного в разд. 4. На демонстрационном этапе будет проведено тематическое исследование, которое заключается в применении предлагаемого руководства при создании, планировании, разработке требований и проектировании проекта Web-API. На заключительном этапе с использованием методов оценки удобства использования мы оценим, как использование руководства в процессе проектирования Web-API влияет на удобство использования разработанного API. При этом предполагается приблизиться к ответам на такие вопросы, как «Улучшит ли сочетание каких-либо артефактов удобство использования разрабатываемого API», и, применяя руководство по проектированию к тематическому исследованию реальных примеров, проверить, может ли это общее руководство быть полезным для конкретных ситуаций. Наконец, цель оценки предлагаемого руководства заключается в том, чтобы узнать о его эффективности и определить возможности по совершенствованию для последующих проектов.

Список литературы / References

[1] Espinha T., Zaidman A., Gross H.G. Web API growing pains: Loosely coupled yet strongly tied. Journal of Systems and Software, vol. 100, 2015, pp. 27-43.

[2] Зосимов В.В., Христодоров А.В., Булгакова А.С. Программные решения для динамического изменения пользовательского интерфейса на основе автоматически собранной информации о пользователе. Труды ИСП РАН, том 30, вып. 3, 2018 г., стр. 207-220. DOI:10.15514/ISPRAS-2018-30(3)-1 / Zosimov V.V., Khrystodorov O.V., Bulgakova O.S. Dynamically Changing User Interfaces: Software Solutions Based on Automatically Collected User Information. Programming and Computer Software, vol. 44, no. 6, 2018, pp. 492-498 (2018).

[3] Biehl M. API Architecture: The Big Picture for Building APIs. CreateSpace, 2015, 107 p.

[4] Mulloy B. Web API Design – Crafting Interfaces that Developers Love. Apigee, 2012, 36 p.

[5] Jin B., Sahni S., Shevat A. Designing Web APIs: Building APIs That Developers Love. O'Reilly Media, 2018, 232 p.

[6] ISO/IEC 25010:2011(en). Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. 2011.

[7] Speicher M. What is Usability? A Characterization based on ISO 9241-11 and ISO/IEC 25010. arXiv:1502.06792, 2015, 10 p.

[8] Piccioni M., Furia C.A., Meyer B. An empirical study of API usability. In Proc. of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2013, pp. 5-14.

[9] Gerken J., Jetter H.C., Zöllner M., Mader M., Reiterer H. The concept maps method as a tool to evaluate the usability of APIs. In Proc. of the SIGCHI Conference on Human Factors in Computing Systems, 2011, pp. 3373-3382.

[10] Henning M. API design matters. Communications of the ACM, vol. 52, no. 5, 2009, pp. 46–56.

[11] Myers B.A., Stylos J. Improving API usability. Communications of the ACM, vol. 59, no. 6, 2016, pp. 62-69.

[12] Zibran M.F., Eishita F.Z., Roy C.K. Useful, but usable? Factors affecting the usability of APIs. In Proc. of the 18th Working Conference on Reverse Engineering, 2011, pp. 151–155.

[13] Bhaskar R.K., Anslow C., Brosz J., Maurer F. Developing usable APIs with XP and cognitive dimensions. In Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing, 2016, pp. 101–105.

[14] Peffers K., Tuunanen T., Rothenberger M.A., Chatterjee S. A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems, vol. 24, issue 3, 2007, pp. 45-77.

[15] Tello-Rodríguez M., Ocharán-Hernández J.O., Pérez-Arriaga J.C., Estadística F. De, Veracruzana U. Hacia una guía de diseño para APIs Web con un enfoque en la usabilidad. Abstraction & Application, vol. 25, 2019, pp. 36-60 (in Spanish).
[16] Hunter K.L. Irresistible APIs: Designing web APIs that developers will love. Manning Publications, 2016, 232 p.
[17] Eassa A.M., Elhoseny M., El-Bakry H.M., Salama A.S. NoSQL Injection Attack Detection in Web Applications Using RESTful Service. Programming and Computer Software, vol. 44, vol. 6, 2018, pp. 435-444.
[18] Stylos J., Graf B. et al. A case study of API redesign for improved usability In Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing, 2008, pp. 189-192.
[19] Lee S., Lee S., Lim S. et al. An API design process in terms of usability: A case study on building more usable apis for smart TV platform. In Proc. of the IEEE 38th International Computer Software and Applications Conference Workshops, 2014, pp.567-571.
[20] Ramakrishnan L., Poon S., Hendrix V. et al. Experiences with user-centered design for the tigres workflow API. In Proc. of the IEEE 10th International Conference on e-Science, 2014, pp. 290-297.
[21] Robillard M.P., Deline R. A field study of API learning obstacles. Empirical Software Engineering, vol. 16, 2011, pp.703-732.
[22] Watson R. Applying the Cognitive Dimensions of API Usability to Improve API Documentation Planning. In Proc. of the 32nd ACM International Conference on The Design of Communication, 2014, pp. 1-2.
[23] Sohan S.M., Maurer F., Anslow C., Robillard M.P. A study of the effectiveness of usage examples in REST API documentation. In Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing, 2017, pp. 53-61.
[24] Macvean A., Maly M., Daughtry J. API Design Reviews at Scale. In Proc. of the Conference Extended Abstracts on Human Factors in Computing Systems, 2016, pp. 849-858.
[25] Robillard M.P. What Makes APIs Hard to Learn? Answers from Developers. IEEE Software, vol. 26, no. 6, 2009, pp. 27-34,
[26] Munir M.B., Mushtaq A. A framework for extending usability engineering: API usability essentials: Extending usability via component-based platform. In Proc. of the IEEE Conference on Open Systems, 2012, pp. 1-6.
[27] Kitchenham B., Pretorius R., Budgen D. et al. Systematic literature reviews in software engineering-A tertiary study Information and Software Technology, volume 52, issue 8, 2010, pp. 792-805.
[28] Stylos J., Myers B.A. The implications of method placement on API learnability. In Proc. of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering, 2008, pp. 105-112.
[29] Pearson A. Balancing the evidence: incorporating the synthesis of qualitative data into systematic reviews. JBI Reports, vol. 2, issue 2, 2004, pp. 45–64.
[30] O'Callaghan P., O'Callaghan P. The API Walkthrough Method: A Lightweight Method for Getting Early Feedback about an API. In Proc. of the Workshop on Evaluation and Usability of Programming. Languages and Tools, 2010, pp. 1-6.
[31] Farooq U., Welicki L., Zirkler D. API Peer Reviews: A Method for Evaluating Usability of Application Programming Interfaces. In Proc. of the ACM Conference on Computer Supported Cooperative Work, 2010, pp. 207–210.
[32] Mitra R. Rapido : A Sketching Tool for Web API Designers. In Proc. of the 24th International Conference on World Wide Web, 2015, pp. 1509–1514.
[33] Murphy-Hill E., Sadowski C., Head A. et al. Discovering API Usability Problems at Scale. In Proc. of the 2nd International Workshop on API Usage and Evolution, 2018, pp. 14-17.
[34] Beaton J.K.K., Myers B.A.A., Stylos J. et al. Usability Evaluation for Enterprise SOA APIs. In Proc. of the 2nd International Workshop on Systems Development in SOA Environments, 2008, pp. 29–34.
[35] Zghidi A., Hammouda I., Hnich B. Towards a formal API assessment. In Proc. of the 40th International Conference on Software Engineering: Companion Proceeedings, 2018, pp. 398–399.
[36] Scheller T., Kühn E. Automated measurement of API usability: The API Concepts Framework. Information and Software Technology, vol. 61, 2015, pp. 145-162.
[37] López-Fernández L., García B., Gallego M., Gortázar F.: Designing and evaluating the usability of an API for real-time multimedia services in the Internet. Multimedia Tools and Applications, vol. 76, 2017, pp. 14247–14304.
[38] Arquitectura y estrategia de API Un enfoque coordinado. White paper. CA technologies, 2015, 23 p. (in Spanish).

[39] Wiegers K., Beatty J. Software Requirements (Developer Best Practices), 3rd Edition. Microsoft Press, 2013, 672 p.
[40] Mosqueira-Rey E., Alonso-Ríos D. et al. A systematic approach to API usability: Taxonomy-derived criteria and a case study. Information and Software Technology, vol. 97, 2018, pp. 46-63.
[41] Lauret A. The Design of Web APIs. Manning Publications, 2019, 392 p.
[42] Webber J., Parastatidis S., Robinson I. REST in Practice: Hypermedia and Systems Architecture. O'Reilly Media, 2010, 448 p.

Информация об авторах / Information about authors

Марибель ТЕЛЛО-РОДРИГЕС – бакалавр в области программной инженерии. Научные интересы: программная инженерия, разработка API.

Maribel TELLO-RODRÍGUEZ, Bachelor in Software Engineering. Research interests include software engineering, API design.

Хорхе Октавио ОЧАРАН-ЭРНАНДЕС, кандидат компьютерных наук, профессор факультета статистики и информатики. Область научных интересов: разработка программного обеспечения, архитектура программного обеспечения, разработка требований, разработка API.

Jorge Octavio OCHARÁN-HERNÁNDEZ, Doctor in Computing Sciences, Professor at the School of Statistics and Informatics. Research interests: software engineering, software architecture, requirements engineering, API design.

Хуан Карлос ПЕРЕС-АРРИАГА, магистр компьютерных наук, разработчик программного обеспечения. Область научных интересов: архитектура программного обеспечения, инженерия программного обеспечения, показатели программного обеспечения, программные инструменты, качество программного обеспечения.

Juan Carlos PÉREZ-ARRIAGA, Master in Computer Science, Software Developer. Research interests include software architecture, software engineering, software metrics, software tools, software quality.

Ксавье ЛИМОН, кандидат наук в области искусственного интеллекта, доцент факультета статистики и информатики. Область научных интересов: распределенные системы, архитектуры программного обеспечения, мультиагентные системы, машинное обучение.

Xavier LIMÓN, Doctor of Artificial Intelligence, Associate Professor of the Statistics and Informatics Faculty. Research interests: Distributed Systems, Software Architectures, Multi-agent systems, Machine Learning.

Анхель Х. САНЧЕС-ГАРСИЯ, кандидат наук в области искусственного интеллекта, штатный профессор факультета статистики и информатики Университета Веракруса. Область научных интересов: искусственный интеллект, машинное обучение, эволюционные вычисления, инженерия программного обеспечения.

Ángel J. SÁNCHEZ-GARCÍA, Doctor of Artificial Intelligence, Full-time Professor of the Faculty of Statistics and Informatics, University of Veracruz. Research interests: Artificial Intelligence, Machine Learning, Evolutionary Computing, Software Engenieering.

DOI: 10.15514/ISPRAS–2021–33(1)–13

A reflection on the design and user acceptance of *Tamil talk*

R. Ramachandran, ORCID: 0000-0003-0355-698X <raj.ramachandran@uwe.ac.uk>
E. Ogunshile, ORCID: 0000-0002-6276-188X <Emmanuel.ogunshile@uwe.ac.uk>
University of West of England,
Coldharbour Ln, Bristol, BS16 1QY UK

Abstract. *Tamil talk* is a speech to text application and was designed from a perspective of language and philosophy. This paper takes an indigenous approach in reflecting on the design and user acceptance of *Tamil talk*. The paper makes use of literature in critically reflecting on the design and the potential user acceptance of the application. It takes a multi-disciplinary approach and explores the influence of factors like language shift, language maintenance and philosophy in the context of user acceptance of speech to text. The application may appeal to a section of the native Tamil speakers as suggested in the literature but there are complex challenges that needs further research. Further research shall be in developing the application that conforms to the conceptual framework and widely test with the native speakers to arrive at a more precise prediction of user acceptance.

Keywords: Speech to text; Use acceptance; Philosophy; Tamil; Software Engineering

For citation: Ramachandran R., Ogunshile E. A reflection on the design and user acceptance of *Tamil talk*. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 189-208. DOI: 10.15514/ISPRAS–2021–33(1)–13.

Размышление о дизайне и восприятии пользователями приложения *Tamil talk*

Р. Рамачандран, ORCID: 0000-0003-0355-698X <raj.ramachandran@uwe.ac.uk>
Э. Огуншиле, ORCID: 0000-0002-6276-188X <Emmanuel.ogunshile@uwe.ac.uk>
Университет Западной Англии,
Великобритания, BS16 1QY, Бристоль, Колдхарбор Лейн

Абстрактный. *Tamil talk* – это приложение для преобразования устной речи в текст, разработанное с позиций языка и философии. В этой статье используется автохтонный подход к осмыслению дизайна и принятия пользователями приложения *Tamil talk* на основе анализа литературных источников. Используется междисциплинарный подход и исследуется влияние таких факторов, как смена языка, приверженность языку и философия в контексте принятия пользователем преобразования устной речи в текст. Как полагают авторы литературных источников, такое приложение может импонировать части носителей тамильского языка, но имеются сложные проблемы, которые требуют дальнейшего исследования. Дальнейшие исследования должны быть направлены на разработку приложения, соответствующего концептуальной модели и испытываемого большим числом носителей языка, чтобы прийти к более точному пониманию принятия пользователями этого приложения.

Ключевые слова: преобразование речи в текст; принятие пользователями; философия; тамильский язык; программная инженерия.

Для цитирования: Рамачандран Р., Огуншиле Э. Размышление о дизайне и восприятии пользователями приложения *Tamil talk*. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 189-208 (на английском языке). DOI: 10.15514/ISPRAS–2021–33(1)–13

1. Introduction

Speech is the ability to express thoughts and feelings by articulating sounds. It is a key component of communication for humans. Each language uses phonetic compositions of a limited set of vowels and consonant sounds that form words. Speech recognition is the interdisciplinary subfield of computational linguistics that develops methodologies and technologies that enables the recognition and translation of spoken language into text [1]. There are over 6000 languages spoken in the world. Everett suggests that humans can learn all the sounds they want when they are young. The articulation of sounds gives rise to speech forms. Some languages share similarities whilst some don't. Accuracy of pronunciation by the speaker could be argued as a requirement in languages where there is a one to one correspondence between the sound and orthography. In cultures that consider philosophy as a way of life, it is even more important to consider the philosophical views and practices on how a language ought to be spoken amongst other factors. The research explores the feasibility to develop a speech to text application using a native approach and a conceptual framework of 'what you speak is what you get' with a focus on user acceptance.

This paper is an extension of [2] and builds upon the work of [3]. This paper reflects on Tamil talk – an application that was developed to convert spoken Tamil speech into Tamil orthography with a conception view of 'What you speak is what you get'. It attempts to discuss the user acceptance of Tamil talk using secondary sources such as ethnography, in particular, the paper explores the link between the use of language at social sphere and the behaviour intention to use that language in the technology such as speech to text. It sets focus on areas like language shift, language proficiency and the challenges associated with it in using Tamil talk.

Tamil is a syllabic language with almost one to one correspondence to the sound and orthography [4]. The work of [5], point to unique syllables of Tamil often mispronounced especially the syllable 'zha' by the native speakers for various reasons as also identified and raised in [3] in the context of not just predicting the user acceptance of speech to text but also to be able to use speech to text in Tamil owing to the nature of the language. There are very few studies on recognition of 'zha' as seen in [6], and [7]. It is argued that the views as in [8] on language variation do not apply in this context due to the syllabic nature of the language.

The first successful speech recognition machine was released by three Bell Labs researchers in 1952 and it was called Audrey. This machine was able to recognize spoken digits with 90% accuracy; however, it only recognized the inventor's voice [9]. The technology then developed into understanding English words about 10 years later. Raj Redy was one the first to research on continuous speech recognition as a graduate student in the late 1960s. Systems prior to this, required users to pause after every word.

Speech recognition technology has grown in sophistication and accessibility leading to the ability to convert speech into text for hundreds of languages and dialects. This includes languages like Tamil, which uses a script that is different to Roman script. Nowadays, almost everyone has a smartphone which is more than capable of recognizing spoken language using the most basic of hardware. The research conducted in this paper points to different design possibilities and details of developing a speech to text application, more specifically in a non Roman script. An application was built to convert spoken Tamil to Tamil text and display what has been spoken onto the screen of a given device. Example: A smartphone or a laptop. The paper makes use of literature as a framework to reflect on the design and potential acceptance of the application on the basis of the literature primarily from the perspective of philosophy and cultural anthropology.

The aim of this research is to predict the user acceptance of Tamil talk developed on the conceptual framework as in [2] 'what you speak is what you get'

The application itself was designed to work as a teaching tool for native and non-native Tamil speakers to confirm correct pronunciation of syllables and words.

The structure of the paper is as follows:

- The first section deals with the research carried out into the Tamil language and investigates any

speech to text applications that are currently available.

- The second section describes the method used to design, develop, implement and test the application developed.
- The third section documents the testing carried out on the developed application.
- The final section provides the discussion and conclusion for the project and suggested work that could be conducted in the future.

2. Literature Survey

Speech to text is the conversion of spoken words into text. ASR applications are very widely available and help assist millions of people every day, however, many of these applications have language-based limitations by only supporting a handful of languages. Developing an application that supports "minority languages" would provide great technological benefit and help develop more inclusive and accessible technology [10]. Minority language is defined as a language spoken by less than 50% of the given region, state, or community [11]. This research views the definition of minority language as seen in [12] outside of the context and region where Tamils are in majority, for example, in the Indian state of Tamil Nadu and Puducherry.

2.1 Introduction to Tamil

Tamil is a Dravidian language spoken by eighty million people in South Asia and across fifty-five diaspora countries but predominantly by the Tamil people of Tamilnadu [12]. It is one of the longest surviving classical languages in the world and recorded Tamil literature has been documented for over 2000 years [13].

The Tamil script consists of 12 vowels (Uyir Ezhuthukkal), 18 consonants (Mei Ezhuthukkal) and one special character, the Aytha Ezhuthu (see fig.1). The vowels and consonants combine to form 216 compound characters (Uyir Mei Ezhuthukkal), giving a total of 247 characters [14]. Unlike other South Asian scripts, such as Gujarati, Tamil does not have script to represent voiceless aspirated (such as "kh"), voiced ("g") and voiced aspirated stops ("gh").

Fig 1. Basic Tamil script [14]

2.2 Speech to Text in Tamil

There are many existing speech-to-text software and applications that support the Tamil language. The most commonly used system is Google Translate [15]; which could potentially be used to translate from one language to another, such as French to English, but could also arguably used for speech to text. Unfortunately, there are shortcomings with the existing application. The existing system does not incorporate the concept of 'what you speak is what you get'. The concept as seen in [3] is based on the structure of the language as opposed to an application that accommodates language and pronunciation variation of the users.

This kind of predictive speech-to-text will give the users a sense of pronouncing the words correctly, where in reality the words are mis-pronounced. This makes users believe they speak correctly and will teach them the language incorrectly which is a significant issue [16].

2.3 Design considerations for Tamil talk

In a standard speech recognition system (fig. 2) the raw speech is typically sampled at a high frequency (16KHz – 8KHz) this produces a sequence of amplitude values over time. This raw speech data is then transformed and compressed to enable simplified processing [17]. Many signal analysis techniques are available which can extract useful features and compress the data by a factor of ten, such as Fourier analysis, Perceptual Linear Prediction, and Lineal Predictive Coding.

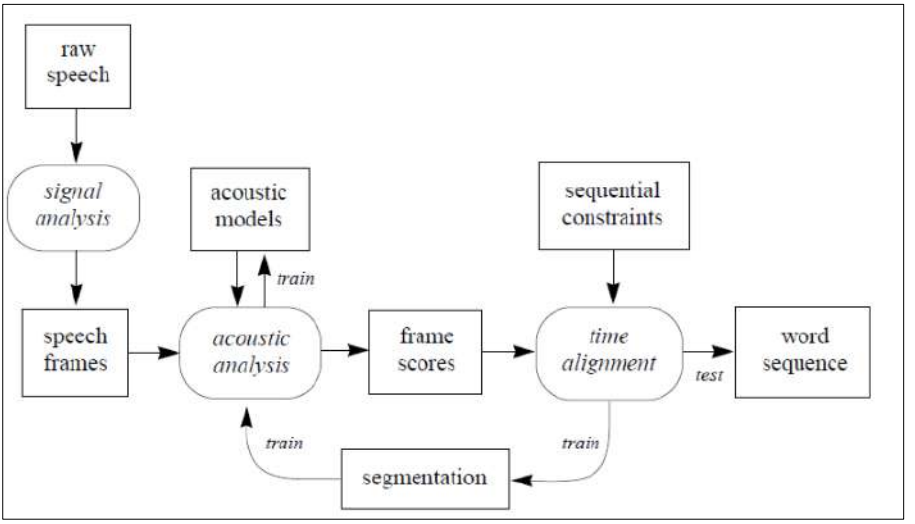


Fig. 2. Structure of standard Speech Recognition System [17]

The speech-to-text transformation is one of the most difficult tasks in computer science because it consists of many difficult problems as seen in [18] but it is an important technology to develop well. Speech recognition has many potential applications including command and control, dictation, transcription of recorded speech, searching audio documents, and interactive spoken dialogues [19]. At the core of all speech recognition systems consists of a set of statistical models representing the various sounds of the language to be recognized.

The task of speech recognition is to find the best matching word-sequence ($\hat{W}$) given the data of an utterance (O). O is a sequence of input vectors generated from the raw speech data. According to Bayes' Theorem the task can be formulated as seen below [20]:

$$\hat{W} = \arg\max_{W} P(W|O) = \arg\max_{W} \overbrace{P(O|W)}^{AM} \cdot \overbrace{P(W)}^{LM},$$

To decode the sequence of words spoken you have to calculate the next probability where W is the decoded sequence and O is the observed sequence or incoming feature vector [20]:

$$\hat{W} = \arg\max_{W} P(W|O).$$

This splits the task into two components $P(O|W)$, also known as the acoustic model and $P(W)$,which is also known as the language model. In most speech recognition systems, the acoustic model is represented by the HMM or the Hidden Markov Model [21]. Each HMM is a finite state machine with n states whereby each state, besides the first and last, has specific output probabilities

and each arc between states is associated with a transition probability [19]. Previously, the output probabilities were modelled by multivariate Gaussian Mixture Model or GMM [22]. Given the restraints in computational power, the GMM is restricted to have a diagonal co-variance matrix and hence require independence between the input dimensions.

Artificial neural networks (ANNs) on the other hand, do not need this requirement of independence, which is why they are more currently used for acoustic modelling and give increased performance to speech recognition, particularly Convolutional Neural Networks and Recurrent Neural Networks [23]. There are several possible ways to exploit ANNs in automatic speech recognition systems, such as the Hidden Markov Model-Artificial Neural Network hybrid system, which takes the advantage of ANN's strong representation learning power and HMM's sequential modelling ability.

There are researches around Connectionist Temporal Classification (fig. 3) which is very similar to the standard HMM-ANN approach however a single-state HMM is used, usually with three times lower output frame-rate and after each single-state HMM a "blank" state is allowed [24]. Further, this model does not need alignment of the training data, which is an advantage, but only starts to show benefits starting with very large quantities of data [25].

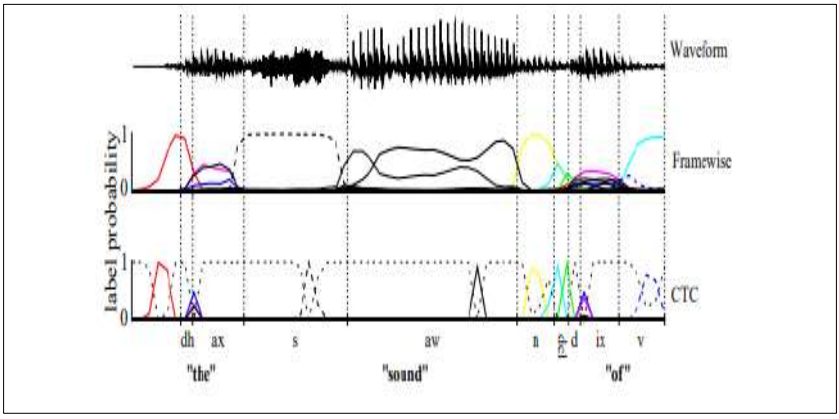


Fig 3. Frame wise and CTC networks classifying a speech signal [24]

All these options were considered at the beginning but we eventually worked on two options that was thought would provide a complete solution of creating a Speech-to-Text application in Tamil that uses the conceptual framework of "what you speak is what you get". While it was deemed the option to develop an application using the Dictionary Model would be the best fit for this project, both options have been described in sections 2.5 and 2.6, allowing for the information to be used for any enhancement to the solution at later date.

2.4 Application Programming Interfaces

An Application Programming Interface (API) is a set of subroutine definitions, protocols, and tools for building application software. In general terms, it is a set of clearly defined methods of communication between various software components. Many API's are built with the intention to allow third party developers to build interesting applications and designed to expand the reach of an organization [26].

All speech recognition engines/API's initially work in the same way as the user's voice is passed through the microphone input to reach the recognition system. There are a number of options for Automatic Speech Recognition API's available that can be divided into closed source code and open source code (fig. 4).

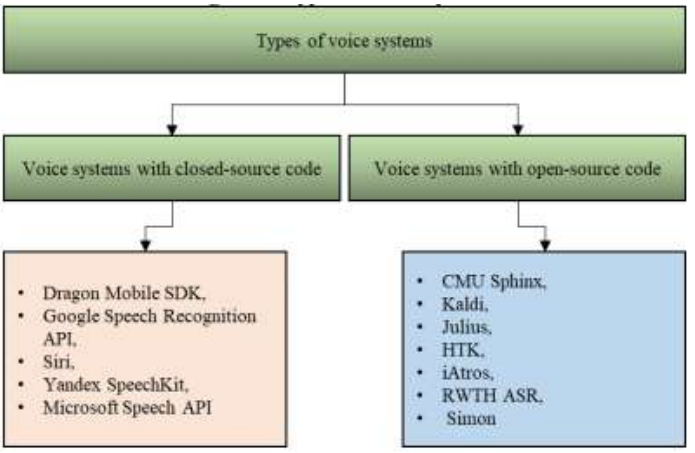


Fig 4. Types of voice systems [29]

Closed source code means that there is no physical access to the code and user will be unable to manipulate and modify it. Two of the biggest companies building voice-powered applications are Google and Microsoft; however, these API's code is inaccessible [27].

CMU Sphinx is one of the most renowned systems and is completely open source. Sphinx enables developers to download and use their code freely as in [28] which includes speech recognizers and acoustic model trainers [29].

There are many benefits of using API's when building applications, especially if the API meets the specific needs of an application, then avoiding the reinvention of the proverbial wheel is a standard piece of wisdom in software development circles [30]. If it is not a good fit for a proposed application but the API is open source and open code then users will be able to access and manipulate the code to fit with an application. Additionally, there are also disadvantages to API's that need to be considered. Anyone could have written the code and if it is closed source there is almost no way of knowing what it actually does and that limits the opportunity and ability to improve.

Google Speech Recognition API is a technology used widely in different research fields for different languages. It allows the user to voice search and its technology is integrated into many smartphones and computers. Originally it only supported a short request of a maximum 40 words and has only recently improved its speech recognition by using a new technology, deep learning neural networks [31].

Part of Googles improvement is a major new feature in the Speech-to-text API that now allows developers to select between different machine learning models [32]. The speech API now supports over 100 languages, including ancient languages such as Georgian (first spoken in 430AD), as well as Swahili, Gujarati and Tamil in a bid to make the internet more inclusive [33].

Google acquired several deep learning companies over the years such as, DeepMind, DNNresearch and JetPac and using these deep learning neural networks Google achieve an 8 percent error rate in 2015, a reduction of more than 23 percent from 2013 [34].

In 1986 Sphinx was launched and became one of the most successful open source systems developed for research purposes using HMM [35]. Developed at Carnegie Mellon University (CMU) it currently has one of the largest vocabularies and speaker independent recognition codebase. The speaker independent speech recogniser uses HMM and n-gram statistical language model, which is able to recognise continuous speech with a big vocabulary [29].

Since its initial release, Sphinx has gone through a number of different modifications. In the past, the decoding strategy of the Sphinx systems tended to be deeply entangled with the rest of the system. As a result of these constraints, the systems were difficult to modify for experiments in other areas [36].

However, the newest version, Sphinx-4, released in 2010, works with various kinds of language specifications such as grammars, statistical language models or blends of both [37]. This means a major benefit of the Sphinx system is that researchers are given more flexibility in the way they incorporate acoustic models allowing constraints to be imposed on the input from the user as seen in [38], making it a great way for identifying particular phonetic sounds for set phrases.

Sphinx has now developed into a toolkit that can be used to develop powerful speech recognition applications and includes several parts as in [39]:

1) PocketSphinx is small fast program, processing sound, acoustic models, grammars and dictionaries.
2) The library Sphinxbase is necessary for PocketSphinx work.
3) Sphinx4 is the recognition library.
4) Sphinxtrain which is for acoustic models training.

Microsoft's Speech API has been around since 1993, developed by three of the four people responsible for the CMU Sphinx-II speech recognition system [31]. Its system is very similar to Google Speech API but uses a server application programming interface (SAPI) for its data. It includes a set of effective methods and its data is well integrated into the .NET framework [40].

Microsoft has continued to develop the powerful speech API and has released a series of increasingly powerful speech platforms as seen in [41], focusing on increasing the emphasis on speech recognition systems and improved Speech API by using a context dependent deep neural network hidden Markov model [42].

432.5 User acceptance consideration

The model proposed by [3] to predict the user acceptance of speech to text application is based on the UTAUT (Unified Theory of Acceptance and Use of Technology) model. The work on language maintenance in Singapore provides a critical insight on the potential consequences arising from language shift. [43] also provides an insight on the relationship between identities and script. Arguably, the identities apart from Tamil speaking Muslims, the concept of script and identity could also be related to the Tamil speaking Brahmans [44] who speak a highly Sanskritised Tamil as pointed in [43], [44], [6]. The used of the term 'Sanskritised Tamil' suggests presence of Sanskrit vocabulary in Tamil arguably by a minority of the Tamil society. There are compelling reasons to consider the aspect of code mixing, code switching, accuracy of pronunciation, choices of orthography as emphasised in [3]. Code mixing, according to [45] are of three types: insertion or embedding, alternation between the structures, congruent lexicalisation.

On the other hand, Creole of the Indian ocean such as Mauritius presents interesting insight on the value of language, and the kind of transformation, the society undergoes over time. The usefulness of English in the economic front, motivated them to retain it as an official language unlike the Reunion which is typically French [46]. The work of [47] suggest the indigenisation of the West in Asian culture such as in Malaysia and Singapore. More importantly, it discusses, the negotiation of identity which we argue is crucial from the point of user acceptance of a language based technology as it is inextricably linked to the choice of language, orthography literacy and preference as seen in [3].

[48] provides example of Ramanujam who lived in the United States from 1959, wrote primarily in English despite deeply rooted in South Indian Brahman culture. The work of [44] provides an interesting insight to the choices made by Brahmans, Tamil Brahmans in their case. [49] presents an interesting studies around language, education from the global southern contexts. In their work, they explore the question "Can English function as the national tongue of the once-colonised nations?". The work of [50] brings out the struggle between the Tamil purists and 'others'. Mandarin, Tamil and Malay are important identity markers for the Chinese, Indians and Malays within Malaysia [51]. The work of [52] and [46] provide perspectives on Standard Spoken Tamil.

2.6 The Dictionary-Based Approach

The second available option was the Dictionary-based approach. This section investigates and discusses it based on the project. Everyday speech by human is referred to as spontaneous speech; it is not scripted and therefore adds a variety of phenomena to a speech recognition task with false starts, human and non-human noises, new words and alternative pronunciations. All of this has to be tackled when creating a system for spontaneous speech recognition [53]. Rather than looking for the "correct" pronunciation of a word the system should automatically expand and adapt the phonetic dictionary and choose a word according to its frequency.

A dictionary-based approach is a basic approach for a speech-to-text system [18]. The pronunciation and sounds are looked up in a sound wave dictionary, these dictionaries are usual modified by hand or by applying phonological rules to a given dictionary [53].

As seen in [53], the Dictionary Learning Algorithm was proposed for using both a speech and phoneme recognizer:

1) Collect all occurrences of each word/tuple in the database and run the phoneme recognizer on them using the smoothed phoneme LM
2) Compute statistics of the resulting phonetic transcriptions of all words/tuples
3) Sort the resulting pronunciation candidates using a confidence measure and define a threshold for rejecting statistically irrelevant variants
4) Reject variants that are homophones to already existing dictionary entries
5) Reject variants which only differ in confusable phonemes
6) Add the resulting variants to the dictionary
7) Test with the modified dictionary on the cross validation set (optional)
8) Retrain the speech recognizer, allowing the use of multiple pronunciations during training.
9) As an optional step corrective phoneme training can be performed
10) Test with the resulting recognizer and the modified dictionary on the cross validation set
11) Create a new smoothed language model for the phoneme recogniser, incorporating all new variants.
12) Optional second pass.

There are disadvantages to the Dictionary Approach especially when entries are added by hand as this usually focuses on single occurrences of a word and it can introduce a number of errors into the system [18]. When modifying words in a dictionary by hand it is important to be aware that experts tend to use the "correct" phonetic transcription of a word which is not necessarily the most frequent or even the most likely transcription for a given task. The actual pronunciations may also be very different for the pronunciation deemed as "correct", and in spontaneous speech there are a lot of alternative pronunciations and they are not easy to predict. The phonetic dictionary is one of the main knowledge-sources for speech recognition but it is still regarded as being less important at acoustic or language modelling. As we have seen in speech recognition research the emphasis is often on the correct pronunciation of a word as it can be found in a lexicon, but this correct pronunciation does not have to be the main feature and does not provide the best recognition accuracy.

The Dictionary-Based method proposes a solution to reduce the system complexity, but this method will fail if a word looked up cannot be found in the systems dictionary. However, for this application this method should provide the user with the accuracy of their speech and not the "correct" spelling of the word as specified in the requirements.

3 *Methods*

Selecting the most appropriate development methodology is not easy. For this project there were two considerations that were priority, the type and complexity of application that was being developed, stakeholders, timescale and the experience of the development team.

3.1 Requirements and design

This subsection identifies the requirements expected to be met to produce a successful product for the end client. The requirements must be clear, complete and understandable to the stakeholders so that all parties are aware of what the system should be doing and any constraints to the software development process.

The method of project management chosen by the team was agile, so the approach to the development was iterative. The application itself was quite simple in design, meaning there are very functions:

a) It should be a web application that is able to run on a PC, smartphone or a tablet.
b) The application must be able to print Tamil orthography to represent the spoken Tamil word.
c) The application must be able to interpret and understand sound waves produced by the speaker in order to pick up the original word or sound.
d) The application must be able to interpret and understand real words and ignore words with errors or mispronunciation.
e) As the application uses a microphone, it must understand what to ignore and what to accept, meaning if there is 2 or more people speaking or there is background noise, it must focus on just the user.

The above are the functional requirements that entail the capabilities, appearance and interactions this system has with the users also known as the target audience for this project. They were measured during the testing and verification stage which is detailed further on in the paper.

The following are the non-functional requirements for the application:

a) The application must be able to recognize the spoken Tamil word to the written Tamil word in real time.
b) The application must be available for use by several users simultaneously.
c) As the application is web based, users should be able to access the application on any internet browser on any device, whether it be Linux, Windows, iOS or Android.
d) The application should achieve a high accuracy rate for translation from speech to text.
e) The application will provide the user with clear options of how to use the features.
f) The application shall have a simple design that only displays the necessary features.

The design process started with each member researching applications already available. This research was discussed during initial meetings with the full group and the sponsor. A high-level idea of the layout was put together. From this a selection of mockups were created using photoshop, and other design software, and an initial back-end prototype was developed calling on Google's speech recognition software. Further web design mockups were created and then compiled to create a high-fidelity prototype which was demonstrated to the sponsor where feedback was obtained and designs further modified.

This helped us steer in the right direction as to how visually the application should look and this was consistently implemented throughout the design and development stages. This technique proved to be the beneficial as it relied on the perspective of many people.

To visualize the application a series of use cases/user stories were created, to map out the user journey and information flow through the application.

3.2 Implementation

The tools used for initial implementation on this application were CMU Sphinx Python-based libraries, SQL Database and Javascript for the front end. These tools were chosen be the structure as they are able to interlink with one another to create a functioning application.

CMU Sphinx and Pocket Sphinx are the best source for this application as it was designed to support such a project.

3.3 Architecture of *tamil talk*

The following section deals with the application from both the back end and client front end.

3.3.1 System Architecture

Originally the system needed to be in a format that was shareable with others and to provide this, it was decided the software will be created in a mobile application format sharable on the Android Play Store or iOS app store. Subsequently, the requirements changed and the application was designed to be web-based. The following technologies were decided to be used:

- Python;
- Ionic Hybrid web application framework;
- CMU Sphinx PocketSphinx API;
- Apache Server;
- MySQL;
- MVC Design Pattern.

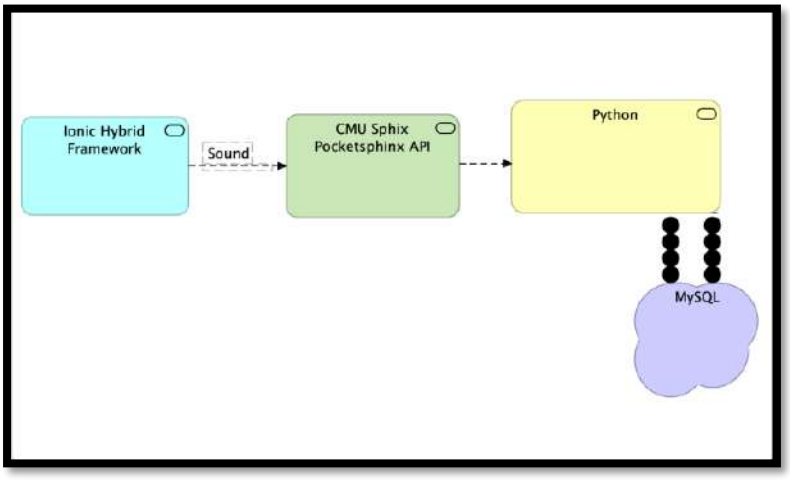


Fig 5. TamilTalk Architecture

To summarize fig. 8, the initial design of the web-based application had a MVC design with an Ionic based front end that communicates with a back-end written in Python, while using the PocketSphinx API to convert the sound into text.

3.3.2 HTML, CSS & Javascript

Due to unforeseen circumstances the original design model was not able to be fully developed. To ensure that a working software was developed for testing another approach had to be adapted. Time constraints placed an urgency on the development process, with this in mind a simple API implementation was used for the application. The Google Web Speech API was the choice of developers as it allowed for an efficient build of a working web application. This API provides a prebuilt dictionary of Tamil syllables, words, and orthography unique to the four major regions where the language is spoken (India, Sri Lanka, Malaysia, Singapore).

The main functionality was written in JavaScript, so it allowed easy implantation into a web browser. The graphical user interface was transferred over from the original design as it was built using simple HTML and CSS. Several constraints would arise while using the Web Speech API to provide the back-end functionality. The Web Speech API's sever-side language is not open source, so ensuring the correct process of conversion from speech to text became a difficult task. Additionally, this API

would require all potential users to have an internet connection as well as use the latest version of Google Chrome, as it is the only web browser that works with this API.

4. Testing and verification

Testing was carried out to help define whether the requirements were achieved and this was implemented in the final stages of the development iteration.

Design protocol testing is imperative in software development as it will help entail how achievable the requirements are. The non-functional requirements were required to be successful and to demonstrate the quality level of the project.

This verification method was conducted by individuals of the team that were not associated with the development of the 'coding aspect' of the software to achieve an unbiased result. Other forms of verification were to monitor the risks associated with the development of the software on a continual basis to measure the success of the application; more risks entailed a higher failure rate. Feedback on initial design, storyboards were provided. The feedback was then incorporated into the design of the application showing the potential that it could be taken much further in the future.

In the testing phase the application was subjected to a single round of black-box testing. In other words, the user testing the application had no prior knowledge or explanation of the application, only a user manual to aid in navigation.

The testing itself was performed on a MacBook Pro. The user was given a set of instructions for tasks to be accomplished while researchers observed and recorded the outcomes. Initially two words were chosen for testing purpose – '*Vanakkam'* and '*Tamizh'.* The rationale for the choice of these words is on the basis of the presence of the unique syllable '*zha*' in the word 'Tamizh' and '*Na'* in the word '*Vanakkam'*. Upon completion of a working prototype of the application, it was identified that it did not satisfy the conceptual framework of '*what you speak is what you get'.* We are not basing our conclusion almost entirely on these two words. However, since the language is syllabic in nature and the one of the main focus within the language were on syllables unique to the language such as '*zha', 'La'* etc, a couple of words such as '*Vanakkam*' and '*Tamizh'* would be sufficient to indicate whether or not the application fulfills the conceptual framework. This, we argue is dependent on the target language and may not be applicable to other languages.

The outcomes of these tasks were measured on a success/failure scale. Four separate test cases were completed during the testing phase with varying tasks for both the application and the user to complete successfully which allowed for testing of the interface design usability and the application functionality at the same time.

5. Results

The user was able to successfully navigate the application to accomplish the primary task of speaking into the microphone and the application converting that to Tamil orthography. Usability was high throughout the majority of the application with a few features, such as the copy and paste button, providing the user with problems.

Specific to application performance the test cases provided mixed results. In Test case one, voice-input data, the application was successful in 100% (4 of 4) of tasks tested and 66.67% (4 of 6) of total tasks. Two tasks in test case one were unable to be completed due to testing environment. Test case two showed the application's accuracy at 66.67% (2 of 3) of the tasks tested and 50% (2 of 4) of the total tasks. One task of test case two was unable to be tested due to lack of access to API server-side functionality. In test case three the application successfully completed 100% (2 of 2) of the tasks tested and 50% (2 of 4) of the total tasks. Again, the two untested tasked were due to lack of access to server-side API access. Finally, in test case four the application successfully completed 33% (1 of 3) of tasks tested and 25% (1 of 4) of the total tasks. One task was not able to be tested due to testing environment. In total, the application successfully completed 75% (9 of 12) of tasks tested and 47.9% (9 of 19) of the total tasks in the test cases.

6. Discussion

This section discusses the implications of the results and further development of the application. The results of the testing of this application provide mixed feedback. The main functionality of this application was developed successfully using available industry standard tools. The user was able to speak into the application and receive accurate feedback in Tamil orthography. One of the main requirements of this application was not met, though. When the user would pronounce words or syllables incorrectly, the application would not provide the intended result. This application, like many other applications, would predict what it anticipated the user meant to say instead of displaying the result exactly as uttered by the user, even if pronounced incorrectly. Several reasons led to these results such as using a closed source API, development abilities, and development time due to circumstances that arose during the process.

The model proposed in [3] to predict the user acceptance of speech to text application is based on the UTAUT model. Keane (2017) discusses the methodology for decolonisation. [43] also provides an insight on the relationship between identities and script. Arguably, the identities apart from Tamil speaking Muslims, the concept of script and identity could also be related to the Tamil speaking Brahmans [44] who speak a highly Sanskritised Tamil as pointed by [43], [44], [5]. The used of the term 'Sanskritised Tamil' suggests presence of Sanskrit vocabulary in Tamil arguably by a minority of the Tamil society. There are compelling reasons to consider the aspect of code mixing, code switching, accuracy of pronunciation, choices of orthography as emphasised in [3]. Code mixing, according to [45] are of three types: insertion or embedding, alternation between the structures, congruent lexicalisation.

On the other hand, Creole of the Indian ocean such as Mauritius presents interesting insight on the value of language, and the kind of transformation, the society undergoes over time. The usefulness of English in the economic front, motivated them to retain it as an official language unlike the Reunion which is typically French [54]. The work of [47] suggest the indigenisation of the West in Asian culture such as in Malaysia and Singapore. More importantly, it discusses, the negotiation of identity which we argue is crucial from the point of user acceptance of a language based technology as it is inextricably linked to the choice of language, orthography literacy and preference as seen in [3]. [48] provides example of Ramanujam who lived in the United States from 1959, wrote primarily in English despite deeply rooted in South Indian Brahman culture.

The work of [44] provides an interesting insight to the choices made by Brahmans, Tamil Brahmans in their case. [49] presents an interesting studies around language, education from the global southern contexts. In their work, they explore the question "Can English function as the national tongue of the once-colonized nations?". The work of [49] brings out the struggle between the Tamil purists and 'others'. Mandarin, Tamil and Malay are important identity markers for the Chinese, Indians and Malays within Malaysia [51]. The work of [50] and [53] provide perspectives on Standard Spoken Tamil.

Tamil talk is an application that was designed on the conceptual framework of 'What you speak is what you get' [2]. This meant that any incorrect pronunciation by the user would also result in incorrect orthography. The design of Tamil Talk and the expectation for the users to pronounce syllables accurately comes from the Vedic philosophy and practices of teaching and learning. However, anecdotal evidences of contemporary usage of language meant that aspects such as code-mixing, code switching had to be considered. In a language based application like speech to text, the ability to speak the language or in other words proficiency in the target language inevitably becomes a key requirement for the user to use the application. In this case, the proficiency of language cannot be assumed for a variety of reasons.

The prediction of user acceptance of Tamil talk shall be explored on the basis of the following comments [3] shown in table 1.

Table 1. Comments to explore user acceptance of Tamil talk

On pronunciation	On orthography	Empirical link between society, language and use of technology
Language and spelling cannot be changed.	I am comfortable with Roman orthography	Society influences the usage of Tamil.
You have to pronounce the word correctly.	Tamil must be written in Tamil orthography. You either write English in English or Tamil. Roman orthography is unacceptable.	What do I gain by learning Tamil?
People must change their pronunciation	Tamil words in Tamil, English in Roman	What we speak is not at all Tamil! Sangam Tamil is pure Tamil.

The Tamil talk application as developed by [2] does not solve the issue of code mixing and code switching as well as the issue of mispronounced syllables displaying 'correctly'. Code mixing and code switching is a common phenomenon that occurs in a multi-lingual or bi-lingual environment. Anecdotal evidences suggest that bth code-mixing and code-switching are common occurrences in Tamil speech. An example of code-mixing is

Tableல இருக்கு (It's on the table)

An example of code switching is:

So இதுல compulsion வர்ரதில்ல this is how you are conditioned. [3]

In designing an application, the output orthography plays a vital role and based on the indigenous approach, these decisions need to be made by the designers of the application rather than by the users.

The output orthography of a code mixed sentence is rather unpredictable. The result is contrary to the views expressed in [3]. However, if we were to consider the work of [55], Tamil talk might be appealing to a section of Tamil speaking Brahmans who extensively code-mix, code-switch into English. The work in [3] questions the necessity of a speech to text application in Tamil, in the event of extensive code-switching and code-mixing [3].

The user acceptance model is broadly divided into two: language and technology. We argue that, to predict the user acceptance for a language based technology, particularly for applications like Tamil talk which is speech to text, the ability to speak, read and write should be a non negotiable component embedded within the user acceptance model. Language maintenance, opportunity to use the language, perceived usefulness of the language, government policy on language are factors that arguably contribute towards the usage of language in the social sphere.

The attribution of lack of English proficiency is also perceived to be a sign of being 'backward and uncivilised' could perhaps be one of the strong factors in language shift towards English [56]. [57] points that Tamils notably the Brahmans within the Malaysian context have largely shifted to English and retain Tamil only for cultural and religious purpose. Whilst this is also true in other contexts as seen in [58], [59]. Language maintenance in country like Malaysia where it is not official unlike its neighbour Singapore where it is official, has been challenging [60].

But, the issue of language maintenance is not unique to Tamil. [61] presents a case study on the struggle of the Malayalee community in Singapore in maintaining their language. The argument of Non-Tamil Indian communities in Singapore (Malayalees in this case), is the lack of recognition of Non Tamil Indian languages in Singapore combined with numerically minority contributes towards shift in other official languages such as English and Tamil. Since, Malayalam is closely related to Tamil, it could be argued that the Malayalam speakers could learn Tamil as a second language with

ease. Second language speakers of Tamil such as the Malayalees of Singapore would have even less motivation to use Tamil in technology.

One should assume that language maintenance in a country or region where it has an official status should relatively be easy. Interestingly, the study by [62] suggest otherwise. In his study, the respondents showed a shift towards English with a belief that the language can replace all domains even in temples where Sanskrit is used. The situation in the Indian state of Tamil Nadu is not encouraging either. It is interesting to observe that the native Tamil speakers struggle to maintain their language even in spaces where there is some form institutional support but at the same time complain about repressive policies in countries like Myanmar. The Burmese of Myanmar also recognized the importance of English and its role in socio-economic upliftment [63]. In fact this seems to have been the trend in many for colonies including African countries [64].

In fact, one could draw parallels between the introduction of English in Africa and Macaulay's proposal of education system in India. In both cases, they were linked with social mobility, progress and language of the intellect. Post-independence many of the African and Asian colonies continued the use of English. Some of them perceive English as a unifying and level playing language that unites various ethnic groups in the country. Could these be the effects of colonialism and language imperialism? Tamils in Myanmar were sandwiched between Burmese and English. The former was often associated with the national identity as in countries like Malaysia, Sri Lanka where Malay and Sinhala respectively are usually perceived by the majority as identity markers of the nationality. As a result, code-mixing by Tamil parents were actively encouraged with a view that it might contribute towards fluency in English.

Although, we cannot generalize, the tendency to switch to English could also be attributed to the colonialism [65]. The studies suggest that the preferential attitude of a native Tamil towards English is not very different in native space. This attitude is problematic because it contributes towards the expectation when designing a speech to text application like Tamil talk. This is because the application was intended to be designed from the perspective of language and philosophy as opposed to the user's usage. The argument of perceived usefulness of the language as a motivation factor to retain in social sphere which subsequently could potentially be the language used in technology merits consideration and further research.

Although, there are debates over the Standard form of Tamil, and in some cases, on the notion of what constitutes Tamil in terms of language. This could be observed in the following comment from [3]:

"*What we speak is not at all Tamil. Sangam Tamil is pure Tamil"*.

This provides an empirical link between the individual perception on what is Tamil, how should it be spoken. Whilst this might fit in the narrative of Tamil purist movement, in practice, the attitude is somewhat different. The pure Tamil movement as seen in the work of [48], was initially directed towards any or perhaps all foreign words and sounds including English, Sanskrit. However, the proponents of pure Tamil movement such as Vedhachalam himself wrote frequently in Tamil. Whilst Vedhachalam's writing in English provides a different view to the people who resonate with the idea of pure Tamil movement or atleast with the concept of speaking pure Tamil. In this context, it is compelling to consider some of the genre that a section of Tamil society seem to identify with.

For example, the Manipravalam (Tamil with a disproportionately large amount of Sanskrit vocabulary) predominantly used by the Tamil speaking Brahmans [44]. On the other hand, Araputamil is a form of Tamil that makes use of liberal Arabic vocabulary [43]. Arguably, there are commonalities in both. The preference of genre by a section of Tamil speaking community could arguably be attributed to religion and philosophy [43]. If we were to consider the orthography, there would be considerable differences in expectation between the two groups.

For example, Tamil- Sanskrit code-switching would involve either the Grantha or the Devanagari script, Tamil-Arabic code switching would involve Tamil-Arabic script or Tamil written in Arabic script as in [43]. Neither of these would be acceptable by the purists as [50] seem to suggest.

Moreover, the Tamil language makes use of six borrowed characters called the 'Grantha'. From a purist point of view, the Grantha would serve no purpose in representing 'pure Tamil'. In fact, the Grantha script was accommodated to facilitate representation of Sanskrit sounds. [3], suggests that clearly there are different expectations amongst users in terms of how their speech ought to appear in text. For example, one of his participants preferred:

"*Tamil words in Tamil, English in Roman*".

Whilst, it is possible to employ bi-lingual and multi-lingual speech recognition, it is appearance of the output that is more challenging from the perspective of readability. The choice of orthography could ideally be classified into one or more of the following reasons:

1) Inability to read, write Tamil either by the user or the person receiving the message.
2) Poor readability as a result of code mixing and code-switching.

The choice of script, code-switching and code-mixing could also arguably be attributed to the users religious or philosophical affiliations. For example, the 'Arabic-Tamil' was predominant amongst Tamil speaking Muslims while the use of disproportionately large Sanskrit vocabulary in Tamil was predominant amongst the Tamil speaking Brahmans. Tamil, Sanskrit and Arabic are all different languages, with different script and sounds, possibly with some common sounds. Due to the complexities and dilemma it presents on the output script of a speech to text application, we are compelled to consider the view of a purist as seen in [50] from a perspective of application design.

Although there are many dimensions and contexts to purism, we restrict our view to the use of language in the context of designing a speech to text application alone. The work of [44] and [50] suggest that identity and language are intertwined. Further, the ethnography of [58], [59] the work of [60], [64] further suggests different perspectives on language by various social groups within the Tamil society. The priorities of learning Tamil, its relevance and maintenance of language appears to have been challenged by the native speakers. There are different notions on purism or in other words what constitutes pure Tamil. For example,

"*Sri Lankan Tamil is very pure. It has not got polluted. I am coming from Chennai. It is the worst place to speak Tamil"* [59].

There are different views and debates on language competency and the ability to speak pure Tamil amongst the Indian and Sri Lankan Tamils. Although a few claim that Jaffna Tamil is pure, in practice, the following example from [59] suggest otherwise:

"Okay naan deti eluthirukkiren".

In our view, pure Tamil of the above example should have been:

"*Sei naan thethiyai ezhuthirukkiraen"(okay I have written the date)*.

We argue that the perception of pure Tamil in the context of Indian and Sri Lankan Tamil are relative. Both the variety code mix and code-switch perhaps may be a little less in the Sri Lankan variety. From the perspective of the application, colloquial Tamil or Literary Tamil matters very little since the pronunciation of Tamil syllables would be the same. But what matters much is the borrowing of words from non- Tamil sources such as English, Sanskrit or Arabic. The motivation of speech to text application like *Tamil talk* from the point of view of preserving one's linguistic heritage is slightly deceptive if we were to consider the following example of the extent of code-switching observed in a conversation:

"I will tell you a time when I was compelled to use technology in Tamizh- Never! By default it was English- there wasn't any compulsion. Its like this, you go to a school, and you graduate out, you be a chemist, you be a pharmacist, you be even a civil engineer, English is your bread and butter, you have to follow English. So இதுல compulsion வர்ரதில்ல this is how you are conditioned." [4]

The above could be seen in parallel to the contexts of Singapore and Myanmar where a similar perception could be observed in terms of the perceived usefulness of the language. The notion of pure language could also be seen in languages like Sinhala. The author points out the '*correct*' pronunciation of /æ/ instead of the modern usage of /a/. In the case of Sinhala (like the Tamil purists),

the emphasis was not only on the pronunciation but also on the grammar. A similar attitude can be seen in [55] work, that centres around the *sabhas* (Theatre) culture of Chennai. The work focuses on the Tamil Brahmans who from the [55] point of view assumes and expects the knowledge of Tamil and English to be able to appreciate the subtle *jokes*. The author further observes that the Tamil Brahmans are notorious for their social conservativism and has an appreciation for traditional values and education.

The practice of code-mixing of two languages at a social level could be argued against the norm of linguistic conservatism. Nevertheless, the Brahman Tamil is an interesting variety to study because of its inherent code switching between Tamil and Sanskrit (*Manipravalam).* However, the perception of Sanskrit-Tamil variety as quintessential Tamil may be problematic. The expectation of the users using the system may vary quite drastically as seen in [3]. Nevertheless, Sanskrit- Tamil code switching (*Manipravalam*), provides an interesting perspective on the script complexity. The work of Ganesan suggest that *Grantha script* accommodates both the Sanskrit and Tamil sounds.

The same could be argued for *ArwiTamil* where the Arabic script is claimed to accommodate both the Arabic and Tamil sounds. However, over a period, attitude towards Grantha has influenced a shift in script to Devanagari for Sanskrit and Tamil script for Tamil. This has contributed increasing illiteracy of the script and has contributed to the perception that Tamil script cannot handle Sanskrit sounds and the need to use Devanagari script for Sanskrit and Tamil script for Tamil. The question of economic value of the language along with the equation on Sanskrit as a '*philosophical language'* meant that the Tamil speaking Brahmans need to acquire proficiency in three scripts namely Tamil, Sanskrit and English.

The work of [65], [44] point to the choices of profession and increasing emigration of Tamil Brahmans to English speaking countries like the United States and the work of [56] suggest that Tamil in non native space refer to a collective cultural identity that share similar values as opposed to the language and the necessity to maintain it. Transliteration is common and positively contributes to the issue of script illiteracy. The dictation experiment in [3] provides an insight on the issues of transliteration which could be argued also contributes to mispronunciation or perceived pronunciation variants in guise of dialects and accents. The *ArwiTamil* also has similar challenges in terms of handling both the Arabic and Tamil sounds in script. In both cases, it is arguably difficult to satisfy users with diverse linguistic preference both in terms of speech and script.

Whilst the rationale to move towards 'pure Tamil' might seem a sensible choice to solve much of the design issue of a language based application such as speech to text, it might neither promote the usability of application nor the acceptance of application due to perceived language shift and differing notions of what Tamil is to individual speaker. Therefore, for a native Tamil speaker who frequently code-switches or code mixes into English, we assume that *Tamil talk* application would be fairly well received. However, this needs to be further investigated under controlled conditions. Nevertheless, we take the view that the design of Tamil talk, although not fully achieved, could be argued in favour of preserving linguistic heritage of a society or at least facilitate in the education process of preserving one's linguistic heritage:

"*Language is like water. Learned men are like filters. Impurities from the outside gets mixed up with water. However, before being used for drinking the water is strained and the impurities are filtered out and discarded In language too, coarse and uncivilised usage seep in. Scholars examine and discard them" [66].*

The design of application from a language perspective as opposed to the user's usage of the language might contradict the traditional approaches to requirements and user acceptance. Nevertheless, the design view merits consideration when a native view is adopted in designing an application. Language, identity and cultural considerations are areas that need to be further explored in the context of user acceptance, requirement elicitation and feasibility study of software engineering. Since these are inextricably linked to each other, careful design considerations and acceptance framework needs to be established. A deeper understanding of how language and culture operate in

a given space, particularly in the native space is undoubtedly important to aid the process of requirement elicitation, feasibility and user acceptance.

In this case, in addition to the language and script, there are a number of other areas that contributes to the challenge of design and acceptance such as language shift, language maintenance, illiteracy in script, institutional policies and so on. Although, there are different views on what constitutes a 'pure form of a language', there appears to be a unanimous agreement. For example, the *Sangam Tamil* or Literary Tamil in the case of Tamil as can be seen in the following excerpt:

"*What we speak is not at all Tamil. We take some words from English, some from Sanskrit. If you talk about proper Tamil, then we need to go to the Sangam Tamil"* [4]

7. Conclusion and further work

This paper has explored the potential acceptance of Tamil talk using relevant literature and framework as proposed in [3]. To be able to predict user acceptance of Tamil Talk, more data is needed. Further work would be to widely test Tamil talk with the native Tamil speakers. In order to provide the best outcome in future development of Tamil speech to text applications we believe that a phonetic dictionary would provide the best result for Tamil speakers. Developing a unique back-end, ideally with the use of open-source API or no API, would ideally allow for the application to achieve the concept of "what you speak is what you get". Future direction could include considering the work of [45] and evaluate its suitability to support the conceptual framework discussed in this paper. Once the application is developed to a reasonable level, the work of [67] might be useful from the perspective of quality testing. The framework of [68] might offer some interesting insight in the context of the reliability and could be further explored as the development progresses. Time constraints, along with the knowledge of development were limitations of this research that hindered the development process. However, it was shown that providing an application for "minority languages" like Tamil is more readily available than ever but not as conceptualized.

This project was challenging; a better understanding of the language would be advised in order to fully create a phonetic dictionary for the outcome to be achieved. Users would discover errors in their pronunciation organically with the use of such an application as words would not be "corrected" for incorrect pronunciation. The user acceptance of Tamil talk cannot be generalised as it lacks data. Since the concept of accuracy of pronunciation arises from the Vedic philosophy, testing it with non-native Tamil speakers in particular with people who do not follow Vedic philosophy might offer some interesting insights. Future work would include extending the application so it fulfils the conceptual framework and use the words tested in [3] as test cases by native Tamil speakers who satisfy the usability criteria.

References

[1] Rabiner L.A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE, vol. 77, no. 2, 1989, pp. 257–286.

[2] Mann D., Frederic K. Weston N., Ramachandran R. & Ogunshile E Tamil talk: What you speak is what you get! In Proc. of the 7th International Conference in Software Engineering Research and Innovation (CONISOFT), 2019, pp. 36-48.

[3] Ramachandran R. Predicting user acceptance of Tamil speech to text by native Tamil Brahmans. Doctoral dissertation. Sheffield Hallam University, 2018, 233 p.

[4] Keane E. Tamil. Journal of the International Phonetic Association, vol. 34, issue 1, 2004, pp. 111-116.

[5] Srinivasan A., Srinivasa K., Kannan K., & Narashimhan D. Speech Recognition of the letter "zha" in Tamil Language using HMM. International Journal of Engineering Science and Technology, vol.1, no. 2, 2009, pp 67-72.

[6] Shulman D. Tamil. Harvard University Press, 2016, 416 p.

[7] Srinivasan A. Real time speaker recognition of letter 'zha' in Tamil language. In Proc. of the Fourth International Conference on Computing, Communications and Networking Technologies, 2013, pp. 1-5.

[8] Crystal D. A little book of language (Little Histories). UNSW Press, 2010, 272 p.

[9] Boyd C. Speech Recognition Technology: The Past, Present, and Future. Available at https://medium.com/swlh/the-past-present-and-future-of-speech-recognition-technology-cf13c179aaf, accessed March 12, 2020.
[10] Fu T., Gao S., Wu X. Improving Minority Language Speech Recognition Based on Distinctive Features. Lecture Notes in Computer Science, vol 11266, 2018, pp. 411-420.
[11] Pandharipande R. Minority Matters: Issues in Minority Languages in India. International Journal on Mulitcutltural Socieitis (IJMS), vol 4, no. 2, 2002, pp 213-234.
[12] Steever S. Introduction to the Dravidian Languages. In Steever S. (ed.). The Dravidian Languages. London: Routledge, 1998, pp. 1–39/
[13] Zvelebil K. Companion studies to the history of Tamil literature. Leiden: Brill, 1992, 291 p.
[14] Lo L. Tamil. Ancient Scripts. Available at http://www.ancientscripts.com/tamil.html, accessed December 1, 2019.
[15] Argondizzo P. How Accurate is Google Translate in 2018? Available at https://www.argotrans.com/blog/accurate-google-translate-2018/, accessed March 10, 2020.
[16] Devarajan S. Relationship between Japanese and Dravidian (Tamil). Available at http://japanese-dravidian.blogspot.com/2009/01/relationship-between-japanese-and.html, accessed March 10, 2020.
[17] Deng Y., Li X., Kwan C. et al. Continuous Feature Adaptation for Non-Native Speech Recognition. International Journal of Computer, Information Science and Engineering, vol:1, no:6, 2007, pp. 1701-1708.
[18] Pornpanomchai C., Ngamwongsakoller P., Tangpitaksame P. & Wonvattanakij C. Thai-Speech-to-Text Transformation Using Dictionary-Based Technique. In Proc. of the International Conference on Networks and Information, 2012, pp. 65-69.
[19] Gales M., Young S. The Application of Hidden Markov Models in Speech Recognition. Foundation and Trends in Signal Processing, 2007, pp. 195-304.
[20] Blanken H., de Vries A., Blok H. & Feng L. (eds.) Multimedia Retrieval (Data-Centric Systems and Applications). Springer Science & Business Media, 2007, 390 p.
[21] Kreyssig F. Deep Learning for User Simulation in a Dialogue System. Master Thesis, University of Cambridge, 2018, 48 p.
[22] Stuttle M. A Gaussian Mixture Model Spectral Representation for Speech Recognition. PhD Thesis, Cambridge University Engineering Department, 2003, 163 p.
[23] Duran N. & Battle S. Probabilistic Word Association for Dialogue Act Classification with Recurrent Neural Networks. Communications in Computer and Information Science, vol. 893, 2018, pp. 229-239.
[24] Graves A., Fernandez S., Gomez F. & Schmidhuber J. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In Proc. of the 23rd International Conference on Machine learning, 2006, pp. 369–376.
[25] Yu D., Deng L. Deep Neural Network-Hidden Markov Model Hybrid Systems. In Automatic Speech Recognition. Signals and Communication Technology. Springer, London, 2015, pp. 99-116.
[26] Clarke S. Measuring API Usability. Dr. Dobb's Journal, vol. 29, 2004, pp. S6-S9.
[27] Samudravijaya K. & Barol M. Comparison of Public Domain Software Tools for Speech Recognition. In Proc. of the Workshop on Spoken Language Processing, 2003, pp. 125–131.
[28] Lange P. & Suendermann-Oeft D. Tuning Sphinx to Outperform Google's Speech Recognition API. In Proc. of the Conference on Electronic Speech Signal Processing (ESSV 2014), 2014, pp. 32-41.
[29] Matarneh R., Maksymova S., Lyashenko S., & Belova N. Speech Recognition Systems: A Comparative Review. IOSR Journal of Computer Engineering, vol. 19, issue 5, 2017, pp. 71-79.
[30] Atwood J. Don't Reinvent the Wheel, Unless you Plan on Learning More About Wheels. Available at https://blog.codinghorror.com/dont-reinvent-the-wheel-unless-you-plan-on-learning-more-about-wheels/, accessed March 10, 2020.
[31] Kepuska V. & Bohouta G. Comparing Speech Recognition Systems (Microsoft API, Google API and CMU Sphinx). International Journal of Engineering Research and Application, vol. 7, issue 3, Part-2, 2017, pp. 20-24.
[32] Lardinois F. Google 1launches an improved speech-to-text service for developers. Available at https://techcrunch.com/2018/04/09/google-launches-an-improved-speech-to-text-service-for-developers/, accessed March 10, 2020.
[33] Techseen Bureau. Google Cloud Speech API now supports 30 more languages. Available at https://techseen.com/2017/08/14/google-cloud-speech-api-update/, accessed March 10, 2020.
[34] Novet J. Google says its speech recognition technology now has only an 8% word error rate. Available at https://venturebeat.com/2015/05/28/google-says-its-speech-recognition-technology-now-has-only-an-8-

word-error-rate/, accessed March 10, 2020.
[35] Juang B.H., & Rabiner L.R. Automatic speech recognition – A brief history of the technology development. In Elsevier Encyclopaedia of Language and Linguistics, 2005, pp. 806–819.
[36] Walker W., LamereP., Kwok, P. et al. Sphinx-4: A flexible open source framework for speech recognition. SMLI TR2004-0811, Sun Microsystems Laboratories, 2004. 9 p.
[37] Twiefel J., Baumann T., Heinrich S., & Wermte, S. Improving domain independent cloud-based speech recognition with domain-dependent phonetic post-processing. In Proc. of the 28th AAAI Conference on Artificial Intelligence, 2014, pp. 1-7.
[38] Ashwell T. & Elam J. How accurately can Google Web Speech API recognize and transcribe Japanese L2 English learners' oral production? The JALT CALL Journal, vol. 13, vo.1, 2017, pp 59-76.
[39] Беленко М.В., Балакшин П.В. Сравнительный анализ систем распознавания речи с открытым кодом. Международный научно-исследовательский журнал, вып, 4 (58) часть 4, 2017 г., стр. 13-18. / Belenko M.V. & Balakshin P.V. Comparative Analysis of Speech Recognition Systems with Open Code. International Research Journal, issue: № 04 (58) Part 4, 2017, pp. 13-18 (in Russian).
[40] Kamarudin M.R., Yusof M.A.F.M., & Jaya H.T. Low cost smart home automation via Microsoft speech recognition. International Journal of Engineering & Computer Science, vol. 13, no. 3, 2013, pp. 6-11.
[41] Lacoma T. How to set up speech-to-text in Windows 10. Available at https://www.digitaltrends.com/computing/how-to-set-up-speech-to-text-in-windows-10/, accessed March 10, 2020.
[42] Manaswi N. Deep Learning with Applications Using Python. Apress, 2018, 219 p.
[43] Tschacher T. From script to language: the three identities of 'Arabic-Tamil'. South Asian History and Culture, vol. 9, no. 1, 2018, pp. 16-37.
[44] Fuller C.J., & Narasimhan H. Tamil Brahmans: The making of a middle-class caste. University of Chicago Press, 2014, 288 p.
[45] Зиеп Н.Н., Жданов А.А. Нейроноподобный подход к распознаванию речи. Программирование, том 44, no. 3, 2018 г., стр. 49-62 / Diep N.N. & Zhdanov A.A. Neuron-Like Approach to Speech Recognition. Programming and Computer Software, vol. 44, no. 3, 2018, pp. 170-180.
[46] Saravanan V., Lakshmi S., & Caleon I.S. The debate over literary Tamil versus standard spoken Tamil: What do teachers say? Journal of Language, Identity, and Education, vol. 8, no. 4, 2009, pp. 221-235.
[47] Wee, C. L. The Indigenized West in Asian multicultures: literary-cultural production in Malaysia and Singapore. International Journal of Postcolonial Studies, vol. 10, no. 2, 2008, pp. 188-206.
[48] Ramazani J., & Ramanujan A.K. Metaphor and postcoloniality: the poetry of AK Ramanujan. Contemporary Literature, vol. 39, no. 1, 1998, pp. 27-53.
[49] Sefa Dei G.J., & Asgharzadeh A. Language, education and development: case studies from the Southern contexts. Language and Education, vol. 17, no. 6, 2003, pp. 421-449.
[50] Kailasapathy K. The Tamil purist movement: a re-evaluation. Social scientist, vol. 7, no. 10, 1979, pp. 23-51.
[51] López C.C. Language is the soul of the nation: Language, education, identity, and national unity in Malaysia. Journal of Language, Identity & Education, vol. 13, no. 3, 2014, pp. 217-223.
[52] Schiffman H.F. Standardization or restandardization: The case for "standard" spoken Tamil. Language in Society, vol. 27, no. 3, 1998, pp. 359-385.
[53] SlobodaT. & Waibel A. Dictionary Learning for Spontaneous Speech Recognition. In Proc. of the International Conference on Spoken Language Processing, 1996, pp. 2328-2331.
[54] Lionnet F. Créolité in the Indian Ocean: Two models of cultural diversity. Yale French Studies, no. 82, 1993, pp. 101-112.
[55] Rudisill K. Everyday Flamboyancy in Chennai's "Sabha" Theatre. Asian Theatre Journal, vol. 29, no. 1, 2012, pp. 276-290.
[56] Canagarajah A.S. Language shift and the family: Questions from the Sri Lankan Tamil diaspora. Journal of Sociolinguistics, vol. 12, no. 2, 2008, pp. 143-176.
[57] Ridge B. National language planning and language shifts in Malaysian minority communities: speaking in many tongues. Amsterdam University Press, 2012, 208 p.
[58] Das Neela S. Rewriting the past and reimagining the future: The social life of a Tamil heritage language industry. American ethnologist, vol. 38, no. 4, 2011, pp. 774-789.
[59] Das S.N. Between convergence and divergence: Reformatting language purism in the Montreal Tamil diasporas. Journal of Linguistic Anthropology, vol. 18, no. 1, 2008, pp. 1-23.
[60] Schiffman H.H.F. Malaysian Tamils and Tamil linguistic culture. Language & Communication, vol. 22, no. 2, 2002, pp. 159-169.

[61] Pillai A.D. Language Shift among Singaporean Malayalee Families. Language in India, vol. 9, 2009, pp. 1-25.
[62] Kadakara S. Status of Tamil language in Singapore: An analysis of family domain, Education Research and Perspectives, vol. 42, 2015, pp. 25-64.
[63] Muniandy M.K., Nair, G.K.S.N., Shanmugam et al. Sociolinguistic competence and Malaysian students' English language proficiency. English Language Teaching, vol. 3, no. 3, 2010, pp. 145-151.
[64] Barnes L. English as a global language: An African perspective. Studies in the Languages of Africa, vol. 36, issue 2, 2005, pp. 243-265.
[65] Fuller C.J., & Narasimhan H. Traditional vocations and modern professions among Tamil Brahmans in colonial and post-colonial south India. The Indian Economic & Social History Review, vol. 47, no. 4, 2010, pp. 473-496.
[66] Coperahewa S. Purifying the Sinhala Language: The Hela Movement of Munidasa Cumaratunga (1930s–1940s). Modern Asian Studies, vol. 46, issue 4, 2012 , pp. 857-891
[67] Липаев В.В. Методология верификации и тестирования крупномасштабных программных средств. Программирование, том 29, no. 6, 2003 г., стр. 7-24 / Lipaev V.V. A Methodology of Verification and Testing of Large Software Systems. Programming and Computer Software, vol. 29, no. 6, 2003, pp. 298-309.
[68] Фролов А.М. Гибридный подход к повышению надежности программных систем. Программирование, том 30, no. 1, 2004 г., стр. 25-36 / Frolov A.M. A Hybrid Approach to Enhancing the Reliability of Software. Programming and Computer Software, vol. 30, no. 1, 2004, pp. 18-24.

Information about authors / Информация об авторах

Raj RAMACHANDRAN SUBRAMANIAN, PhD, Lecturer in Computer Science at the Faculty of Environment and Technology. Research interests include feasibility studies, requirement engineering, user acceptance of language based technologies, and Human computer interaction.

EmmaРадж РАМАЧАНДРАН СУБРАМАНЬЯН, кандидат наук, преподаватель компьютерных наук на факультете окружающей среды и технологий. Научные интересы включают анализ реализуемости проектов, разработку требований, принятие пользователями языковых технологий и взаимодействие человека с компьютером.

Emmanuel Kayode Akinshola OGUNSHILE, Ph.D., Senior Lecturer in Computer Science. Research interests include Automated Specification, Verification and Testing of Software and Hardware, Model Based System Engineering, Formal Methods, Object-Oriented Languages and Type Theory.

Эммануэль Кайоде Акиншола ОГУНШИЛЕ, кандидат наук, старший преподаватель компьютерных наук. Область научных интересов: автоматическая спецификация, верификация и тестирование программного и аппаратного обеспечения, системная инженерия на основе моделей, формальные методы, объектно-ориентированные языки и теория типов.

DOI: 10.15514/ISPRAS–2021–33(1)–14

Поиск заимствований в армянских текстах путем внутреннего стилометрического анализа

Е.М. Ешилбашян, ORCID: 0000-0002-2200-7065 <yyeshilbashian@ispras.ru>
А.А. Асатрян, ORCID: 0000-0002-2529-0169 <arianasat@ispras.ru>
Ц.Г. Гукасян, ORCID: 0000-0003-2389-517X <tsggukasyan@ispras.ru>
Российско-Армянский университет,
ул. Овсепа Эмина 123, Ереван, 119991 РА

Аннотация. Работа посвящена применению внутренних стилометрических методов в задаче обнаружения текстовых заимствований для армянского языка. Мы исследуем два варианта постановки задачи: обнаружение изменения стиля в документе и обнаружение границ нарушений стиля. Для данных задач в рамках этой работы мы создаем синтетические примеры с заимствованиями для академического, художественного и новостного жанров текста, и на полученных примерах проверяем эффективность алгоритмов иерархической кластеризации и других моделей по обнаружению нарушений стиля из серии конференций PAN.

Ключевые слова: стилометрический анализ; обнаружение текстовых заимствований

Для цитирования: Ешилбашян Е.М., Асатрян А.А., Гукасян Ц.Г. Поиск заимствований в армянских текстах путем внутреннего стилометрического анализа. Труды ИСП РАН, том 33, вып. 1, 2021 г., стр. 209-224. DOI: 10.15514/ISPRAS–2021–33(1)–14

Благодарности. Авторы благодарят Недумова Я.Р., Скорнякова К.А. и Турдакова Д.Ю. за ценные отзывы и обсуждения.

Plagiarism Detection in Armenian Texts Using Intrinsic Stylometric Analysis

Ye.M. Yeshilbashian, ORCID: 0000-0002-2200-7065 <yyeshilbashian@ispras.ru>
A.A. Asatryan, ORCID: 0000-0002-2529-0169 <arianasat@ispras.ru>
Ts.G. Ghukasyan, ORCID: 0000-0003-2389-517X <tsggukasyan@ispras.ru>
Russian-Armenian University,
123 Hovsep Emin str., Yerevan, 0051 Armenia

Abstract. In this work we study the application of intrinsic stylometric methods to the task of plagiarism detection in Armenian texts. We use two task setups from PAN's series of conferences on text forensics and stylometry: style change detection and style breach detection. Style change detection aims to determine whether the text is written by more than one author, while style breach detection detects the boundaries of stylistically distinct text fragments. For these tasks, we generate synthetic test sets for three genres of text: academic, literature, and news, and then use them to evaluate the effectiveness of hierarchical clustering and other relevant models from PAN conferences. We employ a standard set of character-level, lexical and readability features, and additionally perform morphological and dependency parsing of text fragments to extract syntactic features encoding author style information. The evaluation results show that the clustering-based approach fails to correctly detect style change detection in longer texts and is only marginally better for shorter texts. For style breach detection, hierarchical clustering-based approach performs better than a random baseline classifier, but the difference is not sufficient to warrant its practical use. In a complementary experiment, we show that

reducing the number of features and multicollinearity in them via PCA helps to increase the precision of style breach detection methods for certain text categories.

Keywords: stylometric analysis; plagiarism detection

For citation: Yeshilbashian Ye.M., Asatryan A.A., Ghukasyan Ts.G. Plagiarism Detection in Armenian Texts Using Intrinsic Stylometric Analysis. Trudy ISP RAN/Proc. ISP RAS, vol. 33, issue 1, 2021, pp. 209-224 (in Russian). DOI: 10.15514/ISPRAS–2021–33(1)–14

Acknowledgements. The authors thank Ya.R. Nedumov, K.A. Skorniakov and D.Yu. Turdakov for insightful feedback and discussions.

1. Введение

Методы поиска заимствований разбиваются на внешние и внутренние. Внешние методы сравнивают текст проверяемого документа с проверочным набором потенциальных источников. При отсутствии проверочной базы применяются внутренние методы поиска заимствований, которые основываются исключительно на имеющемся документе для нахождения подозрительных участков в нем. Так как для армянского языка наличие такой базы ограничено, и многие заимствования, встречаемые на практике, являются переводом текстов на других языках, то становится актуальным исследование возможности применения внутренних методов поиска заимствований.

Для установления наличия заимствований в тексте, методы внутреннего анализа подвергают его стилометрическому анализу. В этом контексте стиль текста определяется закономерностью использования словоформ, знаков препинания, стиля речи, а также уровнем читабельности. Предполагается, что каждый автор обладает собственным уникальным стилем написания текста, и фрагменты, где наблюдается отхождение от этого стиля, предположительно являются заимствованиями из работ других авторов.

В рамках данной работы мы проверяем на армяноязычных текстах применимость существующих моделей обнаружения нарушений стиля в документах. Эффективность моделей оценивается на автоматически сгенерированных наборах примеров из трех жанров текстов – академических, художественных, и новостных. Мы представляем результаты для двух конфигураций постановки задачи: бинарная классификация, отвечающая на вопрос, содержит ли текст нарушения стиля, и определение границ стилистически однородных участков в тексте. Наборы данных, исходные код и другие ресурсы, необходимые для тестирования моделей, доступны на GitHub[1].

2. Обзор литературы

Исследованию и разработке стилометрических методов были посвящены несколько конференций PAN. В них встречаются разные постановки задач внутреннего анализа, из которых основными являются обнаружение изменения стиля в документе (style change detection) [1-2], обнаружение границ нарушений стиля (style breach detection) [3], кластеризация по авторству (author clustering) [3-4].

2.1 Обнаружение изменения стиля

Обнаружение изменения стиля формулируется как задача классификации, где нужно определить содержит ли текст отклонения от преобладающего в нем стиля. Для обзора существующих методов по обнаружению изменения стиля текста были изучены работы участников конференции PAN 2018 и 2019 годов [1-2]. На тестовых выборках наилучшие результаты показали модели Натха (Sukanya Nath) [5], Златковой (Dimitrina Zlatkova) и др.

[1] https://github.com/ivannikov-lab/style-change-analysis

[6], Хосснйнии (Marjan Hosseinia) и Мукерджи (Arjun Mukherjee) [7], а также Сафина (Kamil Safin) и Огальцова (Aleksandr Ogaltsov) [8].

В [6] документ делится на несколько сегментов, для каждого из которых выполняется извлечение лексических, синтаксических признаков, и далее на каждом из этих групп признаков применяются 4 классификатора – SVM, случайный лес, AdaBoost и многослойный персептрон. Каждому классификатору присваивается вес на основе достоверности предсказания, и на основе этого далее каждой группе признаков сопоставляется вектор с вероятностями предсказания класса. Эти векторы вместе с результатами градиентного бустинга на основе TF-IDF подаются на вход логистической регрессии, результатом которой является ответ, присутствует ли в тексте изменение стиля или нет.

Авторы [5] использует метод, основанный на алгоритме пороговой кластеризации. Для извлечения признаков документ делится на окна, и каждое окно преобразуется в вектор признаков на основе нормализованной частоты выбранных токенов. Для определения расстояния между векторами авторы выбрали меру расстояния Матусита. Идея алгоритма пороговой кластеризации состоит в том, чтобы построить список расстояний между окнами и итеративно выбирать окна с наименьшим расстоянием так, чтобы при формировании кластера ближайшие элементы включались первыми. После этого включаются окна, которые находятся дальше. Результатом алгоритма является число кластеров, которое обозначает количество авторов в проверяемом тексте.

Метод [7] основан на идее определения изменения стиля в тексте на основе синтаксического анализа. Для каждого предложения в документе строится синтаксического дерево разбора, далее последовательность этих деревьев передается на вход двух рекуррентных нейронных сетей, первая из которых обрабатывает последовательность в прямом порядке, как в документе, а вторая – в обратном. Далее, при помощи функции схожести оценивается разность между результатами сетей, на основе которой определяется вероятность изменения стиля текста. Так как данная модель сильно зависима от качества синтаксического анализа текстов, а в случае армянского языка точность таких анализаторов сравнительно низкая, она не рассматривалась в рамках данной работы.

В работе [8], как и в модели [6], используется ансамбль классификаторов, каждый из которых по некоторому признаку определяет, встречается ли в документе изменение стиля текста или нет. Первый классификатор использует 19 статистических признаков (число предложений, частота уникальных слов, длина текста и т.д.), для второго классификатора каждый текст представляется в виде 3000-мерного вектора, который содержит информацию о частоте символьных n-грамм в тексте. Третий классификатор применяется на векторном представлении текста, который содержит векторные представления n-грамм (n=1, …, 6), и размерность которого составляет более 3 миллионов. В конце вычисляется взвешенная сумма результатов классификаторов, которая используется как оценка вероятности присутствия в тексте изменения стиля.

2.2 Обнаружение границ нарушений стиля

Задача выявления границ нарушений стиля заключается в определении моно- или мульти-авторства исследуемого документа и разбиении документа на стилистически однородные фрагменты в случае мульти-авторства с указанием границ смен стиля. Данная задача была предложена участникам PAN в 2017 году [3]. Лучший результат показал метод Карася (Daniel Karaś) и др. [9], в котором документ разбивается по параграфам, а затем каждый параграф представляется в виде вектора признаков, полученного путем конкатенации tf-idf значений по униграммам слов, по 3-граммам, по стоп-словам, по размеченным частям речи и пунктуационным символам. Для выявления отклонений от стиля используется статистический подход – тест Вилкоксона над попарными векторами-строками. Наименьшие

30% значений тестов считаются аномальными и ставится граница в начало крайнего параграфа соответствующего теста.

Другие методы из этой конференции, Хана (Jamal Ahmad Khan) и др. [10], а также Сафина и др. [11] были менее эффективны в данной задаче, и поэтому не рассматривались в нашей работе для адаптации и применения к армянскому языку. В этих методах сравниваются представления рядом стоящих предложений. В [10] документ разделяется по предложениям, и рассматриваются скользящие окна из трех предложений, имеющие одно общее предложение. Они используют статистические признаки на основе синтаксических, лексических характеристик текста. Алгоритм [11] применяет модель Skip-Thought, основанную на предсказании предыдущего и следующего предложения, с архитектурой кодировщик-декодировщик и с использованием GRU сетей. Предложение считается отклонением, если среднее расстояние его векторного представления от остальных векторов больше заданного допустимого. Помимо того, что данная модель работала значительно медленно по сравнению с остальными, ее адаптация к армянскому языку была бы проблематичной также потому, что для предобучения и эффективной работы используемой нейронной сети понадобилось бы большое количество текстовых данных.

2.3 Кластеризация по авторству

Задача авторской кластеризации заключается в группировании небольших моно-авторских документов по группам авторов. В рамках данной работы мы рассматривали модели, предназначенные для решения той вариации этой задачи, в которой нужно каждый документ необходимо присвоить кластеру одного автора. Данная задача авторской кластеризации рассматривалась в рамках открытых соревнований PAN 2016 и 2017 [3-4].

В условиях неизвестности точного количества авторов лучший результат показали алгоритмы Гомес-Адорно (Helena Gómez-Adorno) и др. [12], Гарсии-Мондеха (Yasmany García-Mondeja) и др. [13], а также Кохера (Mirco Kocher) и Савоя (Jacques Savoy) [14], использующие иерархическую кластеризацию. В модели [12] для представления документа используются n-граммы слов и символов, над которыми далее применяется иерархическая агломеративная кластеризация, где количество кластеров определяется путем максимизации индекса Калинского-Харабаша (Calinski-Harabaz score, отношение среднего значения дисперсии между кластерами и дисперсия внутри кластера).

В [13] для векторного представления документов используется модель мешка слов. Схожести между векторами определяется по трем функциям расстояния – косинусное, Дайса и Жаккара. Они использовали β-compact графовую кластеризацию – метод построения ориентированного графа, где вершины i и j соединяются в один кластер, если они являются β -схожими и вершина j наиболее схожая из всех остальных вершин. Порог схожести β определяется на стадии тренировки, при котором максимизируется метрика FBcubed.

В своем решении авторы [14] используют иерархическую кластеризацию Single-link, где для слияния текущих кластеров сравниваются их наиболее близкие вектора. Для представления документа вектором алгоритм использует частоты наиболее частых токенов (слов и пунктуации) и наиболее частых символьных 6-грамм. Для обозначения схожести используется Канберровское расстояние, и для сравнения полученных значений вводится понятие «достаточно маленькое расстояние».

Все указанные алгоритмы используют некую парадигму «снизу-вверх», оценивая схожесть всех пар документов и используя эти значения для последовательного формирования кластеров. Применение методов кластеризации для группировки небольших текстов по стилю также было исследовано в работе [15], где применяется метод k-средних на коллекции анонимных электронных писем. С учетом того, что документы, на которых проверялись эти модели кластеризации по авторству, имели размер, сравнимый с параграфами текста, мы

посчитали целесообразным применение этих моделей в задаче определения нарушений границ стиля.

3. Методы

Для определения изменения стиля в данной работе мы рассматриваем алгоритм [5], а для задачи нахождения границ нарушений стиля – алгоритм [9] и иерархическую модель кластеризации. В рассматриваемых моделях мы в качестве единиц стилистически однородных участков текста используем параграфы, считая маловероятной нарушения стиля внутри одного параграфа [16]. Соответственно, в алгоритме [9] и иерархической кластеризации документ представляется как список параграфов.

Для кластеризации применяется агломеративная модель, в которой новые кластеры создаются рекурсивным объединением схожих мелких кластеров. Схожесть между кластерами определяется по методу Варда как прирост суммы квадратов расстояний объектов до центра кластера в случае их объединения. На каждом шаге алгоритма объединяются те два кластера, которые приводят к минимальному увеличению дисперсии. В результате работы алгоритма каждому параграфу присваивается номер кластера. Для обнаружения нарушения стиля между параграфами последовательно сравниваются номера их кластеров. В случае если два соседних параграфа принадлежат разным кластерам, между ними ставится граница смены стиля.

2.1 Признаки

В задаче определения изменения стиля модель [5] используется в своем оригинальном варианте, без изменений набора признаков. В задаче нахождения границ нарушений стиля в модели [9] и кластеризации, помимо общепринятных распространенных признаков из решений PAN, мы дополнительно извлекаем собственный набор признаков для описания синтаксических особенностей параграфа. Эффективность синтаксической информации в стилометрических задачах была установлена в работах [17-19]. В целом, используются различные символьные, лексические, морфологические и синтаксические признаки (полный список приводится в Приложении А).

1. Символьные признаки: мы извлекаем две группы символьных признаков: первая группа описывает наличие и частоту используемых суффиксов и префиксов, вторая - описывает наличие и частоту знаков пунктуации, а также стиль их комбинирования с пробелами (например, наличие пробела перед знаком пунктуации).
2. Лексические признаки:
   - общие характеристики, такие как наличие неформальных слов, терминов на латинском/кириллице, наличие и частота использования длинных и коротких слов;
   - использование сокращений, использование сокращений вместо полных форм, предпочтительные формы сокращений и стиль их склонения, а также выбор заглавных или строчных букв при написании;
   - формы написания количественных и порядковых числительных, стиль написания дат;
   - наличие и частота нетипичных для других документов n-грамм слов (с низким значением idf, вычисленном на коллекции текстов [20]).
3. Синтаксические признаки:
   - общие закономерности на основе длин предложений (отдельно рассматривается и посимвольная длина, и количество слов в предложении), как, например, минимальная, средняя, максимальная длина предложений, наличие и частота длинных и коротких предложений, количество предложений в параграфе;
   - предпочтения к использованию определенных частей речи, наличие и частота использования конкретных 2-грамм и 3-грамм частей речи;

- наличие и частота конкретных синтаксических зависимостей в предложении, использование простых или сложных предложений;

4. Читабельность: индексы читабельности текста, адаптированные к армянскому языку: Flesch reading ease, Flesch-Kincaid, SMOG, Coleman-Liau, automated readability index, Dale-Chall, difficult words, Linsear write formula и Gunning fog.

Для токенизации текста на слова и предложения использовалась библиотека UDPipe 2.0[2] Для морфологического и синтаксического анализа использовалась библиотека Stanza[3].

3. Эксперименты

Мы провели эксперименты, чтобы оценить качество работы алгоритмов определения изменения стиля и границ нарушений стиля в документе. Для этих экспериментов мы создали автоматически сгенерированные синтетические тестовые примеры для трех жанров текстов: академического, художественного и новостного. Также, чтобы оценить эффективность этих алгоритмов, их результаты на этих данных были сравнены со случайными и тривиальными базовыми моделями.

3.1 Наборы данных

Для построения набора данных были использованы документы с одинаковой тематикой, написанные разными авторами. Построение каждого примера происходило автоматически, путем объединения параграфов исходных документов. Один из документов выбирался в качестве основного, и далее некоторые из его параграфов случайным образом заменялись на параграфы других документов одинакового содержания. В процессе генерации примеров вероятность замены параграфа на заимствование из другого документа контролировалась с целью, чтобы получить примеры с разным содержанием заимствований. Также многие сгенерированные примеры в итоге были отфильтрованы, чтобы избежать присутствия очень близких примеров в полученном наборе.

В качестве исходных данных выбирались следующие наборы данных.

1. Защищенные кандидатские диссертации РА.

   Работы, представленные в открытом доступе на официальном сайте[4], доступны только в формате PDF. Мы обработали извлеченные тексты (удалили вводные главы, списки литературы, сокращений, нумерацию, сноски, номера страниц) и далее сгруппировали на основе направления работы. При выборе данного корпуса мы основывались на предположении, что все работы написаны одним автором.

2. Энциклопедические статьи.

   В качестве источников для генерации примеров мы использовали статьи об известных личностях из Википедии и онлайн-версии Армянской энциклопедии[5].

3. Учебники.

   В качестве другого источника примеров академического жанра, были выбраны материалы из учебники старших школ и университетские пособия. Были выбраны учебники по истории армянского народа 7-9 классов на армянском языке и пособие «История Армении» для вузов[6].

4. Художественные тексты

   Для генерации примеров с текстом из литературного жанра мы использовали

[2] http://ufal.mff.cuni.cz/udpipe/2
[3] https://stanfordnlp.github.io/stanza/
[4] http://etd.asj-oa.am/
[5] http://www.encyclopedia.am/
[6] https://lib.armedu.am/

произведения, которые имеют одинаковую сюжетную линию, но написаны разными авторами: произведения «Фазан» Акселя Бакунца и сценарий к фильму «Этот зеленый, красный мир», написанный Грантом Матевосяном, по мотивам данного произведения;

5. Новостные статьи

   Чтобы найти новостные тексты, описывающие одно и то же событие, но из разных источников, мы использовали агрегатор новостей NewsHub[7].

В результате было сгенерировано 144 примера из учебников, 84 новостных, 50 художественных, 400 из диссертаций и 44 из энциклопедий (табл. 1).

Табл. 1. Количество примеров в сгенерированных тестовых наборах
Table 1. The number of examples in the generated test cases

Жанр		Количество примеров								
		Все	Количество параграфов				Процент заимствований			
			5	10	25	50	0	0-20	20-40	40-50
Академический	Диссертации	400	100	100	100	100	70	82	173	75
	Учебники	144	14	33	46	51	10	41	65	22
Художественный		50	11	22	17		10	14	19	6

3.2 Результаты

Обнаружение изменения стиля: в данной конфигурации мы протестировали алгоритмы [5], [6] и, дополнительно, алгоритмы обнаружения границ нарушений стиля – [9] и кластеризацию. Последние три модели почти на всех примерах давали положительный ответ, то есть фактически работали как тривиальный классификатор. По этой причине в данном разделе более подробно изучаются только результаты модели [5] (Таблица 2). Для каждого тестового набора результаты приводятся отдельно. Учитывая преобладание положительных примеров в наборах, для получения полной картины поведения алгоритмы также изучается доля ложно положительных предсказаний, помимо точности и полноты.

Табл. 2. Качество обнаружения изменения стиля модели [5]
Tab. 2. The quality of the model [5] in detection of changes in the style

Жанр текстов		Точность	Полнота	F1	Специфичность	Доля ложно положительных (FPR)
Академический	Диссертации	0.9002	0.8105	0.853	0.3432	0.6567
	Учебники	0.9217	0.8217	0.8688	0.4	0.6
	Энциклопедии	0.4074	0.55	0.468	0.3333	0.6666
Художественный		0.8437	0.675	0.75	0.5	0.5
Новости		0.0417	0.1428	0.0645	0.7012	0.2987

Среди академических текстов алгоритм работает с примерно одинаковым уровнем доли ложно положительных, и дает правильные предсказания лишь на около трети полностью оригинальных примеров. В этом плане заметно лучше результаты для новостных текстов, доля ложно положительных ниже 30%. Однако для этих примеров присутствует другая проблема: алгоритм показывает склонность к классификации большинства новостных примеров как полностью оригинальных. Для энциклопедических примеров также присутствует данная проблема.

[7] https://newshub.am/

Чтобы понять насколько действительно эффективен алгоритм, мы сравниваем его со случайным классикатором (рис. 1). Для сравнения качестве метрики используется специфичность, которая показывает какой процент действительно оригинальных документов был предсказан таковым алгоритмом. Данный анализ показал, что для длинных документов (с количеством параграфов 25 и более), независимо от жанра текста результаты алгоритма работает значительно хуже базового метода. Это связано с тем, что с ростом размера текста алгоритм показывает тенденцию находить присутствие заимствований.

Дополнительно мы также изучили влияние количества заимствований в тексте документа на качество предсказаний (рис. 2). Для всех категорий примеров наблюдается четкая корреляция роста качества предсказания с ростом процента заимствований.

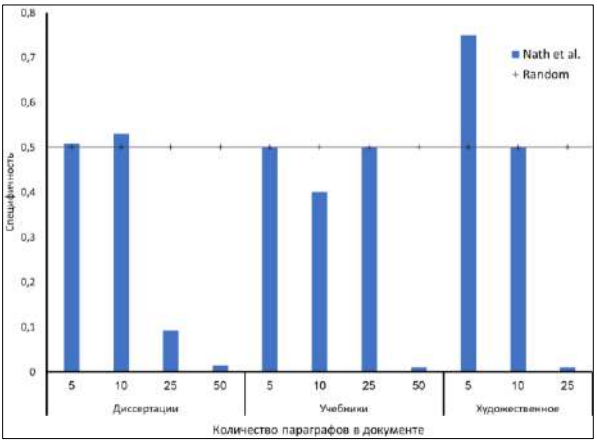


Рис. 1. Уровень специфичности обнаружения изменения стиля в тексте в зависимости от его жанра и длины

Fig. 1. The level of specificity of detecting changes in style in the text depending on its genre and length

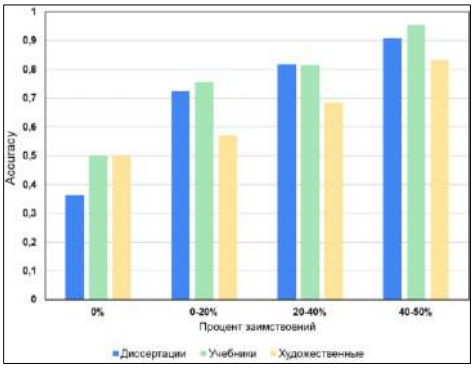


Рис. 2. Зависимость accuracy от процента заимствований для модели [5]

Figure: 2. Dependence of accuracy on the percentage of borrowing for the model [5]

Обнаружение границ нарушений стиля: в данной конфигурации мы протестировали алгоритмы [9] и агломеративную кластеризацию (AC). Аналогично [2], мы тоже в качестве базового метода используем случайный классификатор, однако с немного другой конфигурацией: в наших экспериментах мы используем базовый метод, который для каждой границы параграфов с вероятностью 0.25 предсказывает присутствие нарушения стиля.

Для оценки качества этих моделей используются точность (WinP) и полнота (WinR), заданные по формулам:

$$WinP = \frac{TP}{TP + FP} \quad (1)$$

$$WinR = \frac{TP}{TP + FN} \quad (2)$$

Точность определяется как соотношение правильно поставленных моделью границ от всех границ (TP), поставленных моделью (TP+FP), а полнота – соотношение правильно поставленных моделью границ от всех реальных границ (TP+FN).

Результаты алгоритмов для каждого тестового набора иллюстрированы на рис. 3. Чтобы определить, насколько уверенно можно утверждать превосходство [9] и AC над случайным классификатором, мы вычислили 90% доверительный интервал. В целом, почти на всех текстах, кроме художественных, кластеризация обошла [9]. Также, для художественных текстов преимущество данного алгоритма над базовым методом несущественное, а для энциклопедий и новостных текстов можно говорить, что его качество заметно хуже. Кроме диссертаций и художественных текстов, алгоритм [9] показал результаты хуже базового метода. В случае учебников и энциклопедий его точность оказалась близкой к нулю. Для этих категорий текстов данный алгоритм редко находил границы нарушений стиля.

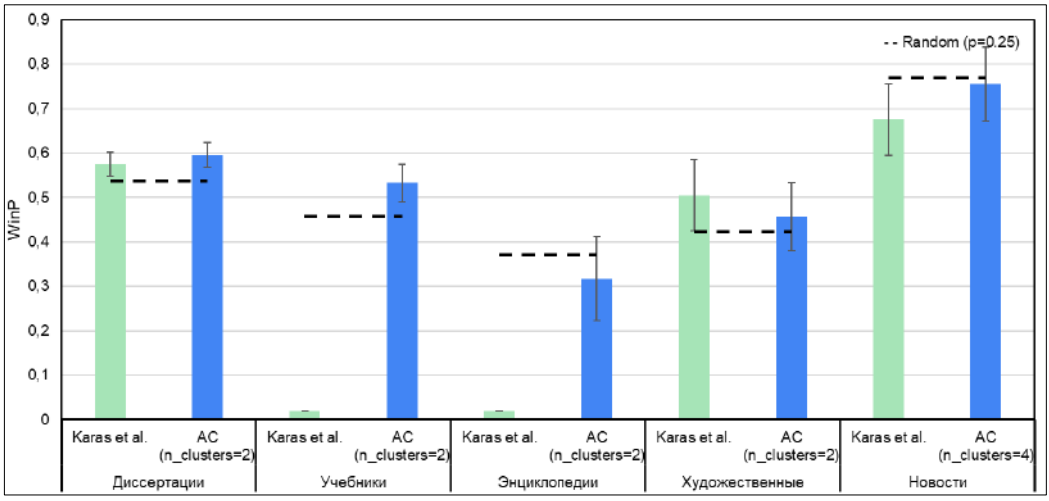


Рис. 3. Сравнение наиболее точных (90% доверительный интервал) моделей обнаружения границ нарушений стиля и случайного классификатора для каждого жанра
Fig. 3. Comparison of the most accurate (90% confidence interval) models for detecting the boundaries of style violations and a random classifier for each genre

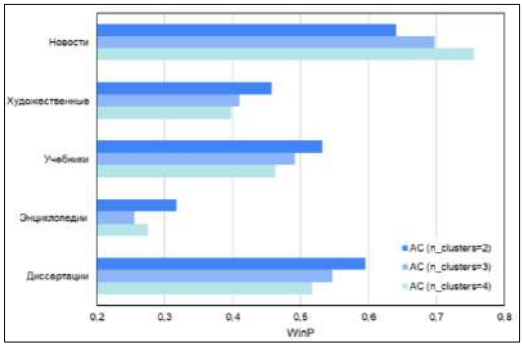


Рис. 4. Зависимость точности модели AC от количества кластеров
Fig. 4. Dependence of the accuracy of the AC model on the number of clusters

Для алгоритма кластеризации мы проверяли качество для количества кластеров 2, 3, и 4 (рис. 4). За исключением новостных текстов, в среднем самая высокая точность была получена при

ограничении количества кластеров на 2. С учетом наличия многих тесно связанных признаков в описаниях параграфов, а также ограниченного количества параграфов в отдельном документе, для повышения точности работы мы также протестировали сокращение размерности признакового описания с помощью PCA, следуя примеру [21], и для художественных текстов, учебников и диссертаций получили заметные улучшения в точности (рис. 5).

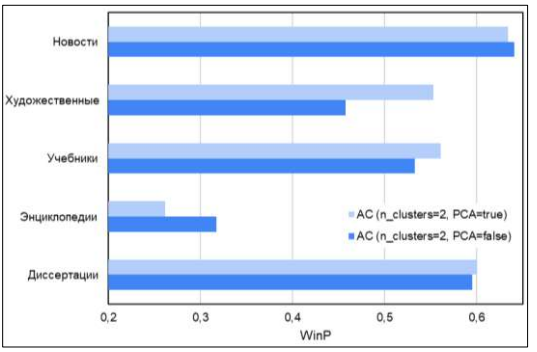


Рис. 5. Влияние PCA на точность модели AC
Figure: 5. Influence of PCA on the accuracy of the AC model

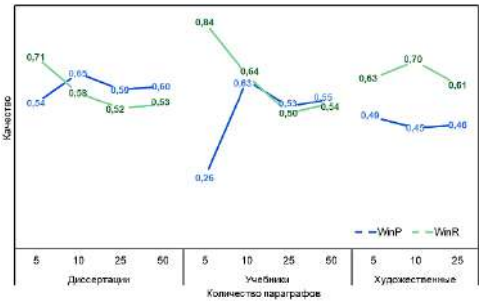


Рис. 6. Зависимость качества обнаружения границ изменений стиля от длины документов для модели AC (n_clusters=2)
Fig. 6. Dependence of the quality of detection of boundaries of style changes on the length of documents for the AC model (n_clusters = 2)

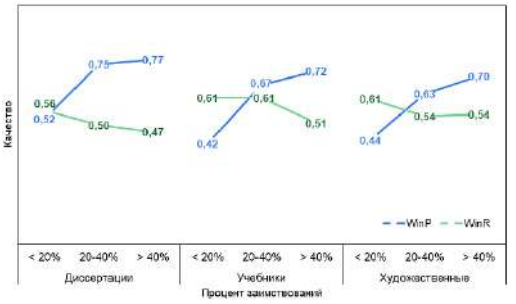


Рис. 7. Зависимость качества обнаружения границ нарушений стиля от процента заимствований в документах для модели AC (n_clusters=2)
Fig. 7. Dependence of the quality of detection of style violation boundaries on the percentage of borrowings in documents for the AC model (n_clusters = 2)

Дополнительно было изучено влияние длины текста и количества заимствований на качество предсказаний алгоритма кластеризации (рис. 6 и рис. 7). С точки зрения полноты результатов, наблюдается закономерность по его уменьшению параллельно с ростом длины документа и процента заимствований. В случае точности нет явной корреляции между значениями по этой

метрике и количеством параграфов в тексте. Только в случае текстов учебников наблюдается аномально низкая точность для маленьких документов с 5 параграфами. При этом чем меньше заимствованных участков текста в документе, тем менее точно работает алгоритм.

4. Заключение

В данной работе была исследована эффективность стилометрических методов внутреннего анализа для нахождения заимствований в текстах на армянском языке. Были изучены для трех категорий текстов (академических, художественных, новостных) и протестированы алгоритмы, основанные на лучших моделях с соревнований PAN 2017-2019. Было установлено, что для длинных документов алгоритмы нахождения изменения стиля на основе кластеризации показывают низкую специфичность и поэтому неэффективны. В задаче определения границ нарушений стиля для конкретных категорий текстов алгоритмы достигают качества выше случайного классификатора. Кроме того, была установлена эффективность применения PCA на входных признаках для сокращения их размерности.

В качестве направления дальнейших исследований рассматривается поиск более эффективных и точных наборов признаков. Дополнительным обоснованием необходимости в таком исследовании служит тот факт, что текущие синтаксические анализаторы армянского языка все еще сравнительно отстают от state-of-the-art для аналогичных языков.

Список литературы / References

[1]. Mike Kestemont, Michael Tschuggnall, Efstathios Stamatatos, Walter Daelemans, Günther Specht, Benno Stein, and Martin Potthast. Overview of the Author Identification Task at PAN-2018: Cross-domain Authorship Attribution and Style Change Detection. Working Notes of CLEF 2018 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 2125, 2018.

[2]. Eva Zangerle, Michael Tschuggnall, Günther Specht, Martin Potthast, and Benno Stein. Overview of the Style Change Detection Task at PAN 2019. Working Notes of CLEF 2019 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 2380, 2019.

[3]. Michael Tschuggnall, Efstathios Stamatatos, Ben Verhoeven, Walter Daelemans, Günther Specht, Benno Stein, and Martin Potthast. Overview of the Author Identification Task at PAN 2017: Style Breach Detection and Author Clustering. Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 1866, 2017.

[4]. Paolo Rosso, Francisco Rangel, Martin Potthast, Efstathios Stamatatos, Michael Tschuggnall, and Benno Stein. Overview of PAN 2016 – New Challenges for Authorship Analysis: Cross-genre Profiling, Clustering, Diarization, and Obfuscation. Lecture Notes in Computer Science, vol. 9822, 2016, pp. 332-350.

[5]. Sukanya Nath. Style Change Detection by Threshold Based and Window Merge Clustering Methods. Working Notes of CLEF 2019 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 2380, 2019.

[6]. Dimitrina Zlatkova, Daniel Kopev, Kristiyan Mitov, Atanas Atanasov, Momchil Hardalov, Ivan Koychev, and Preslav Nakov. An Ensemble-Rich Multi-Aspect Approach for Robust Style Change Detection – Notebook for PAN at CLEF 2018. Working Notes of CLEF 2018 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 2125, 2018.

[7]. Marjan Hosseinia and Arjun Mukherjee. A Parallel Hierarchical Attention Network for Style Change Detection – Notebook for PAN at CLEF 2018. Working Notes of CLEF 2018 – Conference and Labs of the Evaluation Forum, vol. 2125, 2018.

[8]. Kamil Safin and Aleksandr Ogaltsov. Detecting a Change of Style Using Text Statistics – Notebook for PAN at CLEF 2018. Working Notes of CLEF 2018 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 2125, 2018.

[9]. Daniel Karaś, Martyna Śpiewak, and Piotr Sobecki. OPI-JSA at CLEF 2017: Author Clustering and Style Breach Detection – Notebook for PAN at CLEF 2017. Working Notes of CLEF 2017 – Conference and Labs of the Evaluation Forum, vol. 1866, 2017.

[10]. Jamal Ahmad Khan. Style Breach Detection: An Unsupervised Detection Model – Notebook for PAN at CLEF 2017. Working Notes of CLEF 2017 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 1866, 2017.

[11]. Kamil Safin and Rita Kuznetsova. Style Breach Detection with Neural Sentence Embeddings – Notebook for PAN at CLEF 2017. Working Notes of CLEF 2017 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 1866, 2017.
[12]. Helena Gómez-Adorno, Yuridiana Alemán, Darnes Vilariño Ayala, Miguel A. Sanchez-Perez, David Pinto, and Grigori Sidorov. Author Clustering using Hierarchical Clustering Analysis – Notebook for PAN at CLEF 2017, Working Notes of CLEF 2017 м Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 1866, 2017.
[13]. Yasmany García-Mondeja, Daniel Castro-Castro, Vania Lavielle-Castro, and Rafael Muñoz. Discovering Author Groups using a B-compact graph-based Clustering – Notebook for PAN at CLEF 2017. Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 1866, 2017.
[14]. Mirco Kocher and Jacques Savoy. UniNE at CLEF 2017: Author Clustering – Notebook for PAN at CLEF 2017. Working Notes of CLEF 2017 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, vol. 1866, 2017.
[15]. Iqbal Farkhund, Hamad Binsalleeh, Benjamin C.M. Fung, and Mourad Debbabi. Mining writeprints from anonymous e-mails for forensic investigation. Digital Investigation, vol. 7, issue 1-2, 2010, pp. 56-64.
[16]. Zuo Chaoyuan, Yu Zhao, and Ritwik Banerjee. Style Change Detection with Feed-forward Neural Networks. Working Notes of CLEF 2019 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR Workshop Proceedings, vol. 2125, 2019.
[17]. Hirst Graeme, and Ol'ga Feiguina. Bigrams of syntactic labels for authorship discrimination of short texts. Literary and Linguistic Computing, vol. 22, no. 4, 2007, pp. 405-417.
[18]. Rupesh Kumar Dewang and A. K. Singh. 2015. Identification of Fake Reviews Using New Set of Lexical and Syntactic Features. In Proc. of the Sixth International Conference on Computer and Communication Technology (ICCCT '15), 2015, pp. 115–119.
[19]. C. Zhao, W. Song, L. Liu, C. Du and X. Zhao. Research on Author Identification Based on Deep Syntactic Features. In Proc. of the 2017 10th International Symposium on Computational Intelligence and Design (ISCID), 2017, pp. 276-279.
[20]. K. Avetisyan and T. Ghukasyan. Word embeddings for the armenian language: intrinsic and extrinsic evaluation. Bulletin of the Russian-Armenian University: Physico-Mathematical and Natural Sciences, no. 1, 2019, pp. 59-72.
[21]. Gishamer Flurin. Using Hashtags and POS-Tags for Author Profiling. Working Notes of CLEF 2019 – Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR Workshop Proceedings, vol. 2125, 2019.

Информация об авторах / Information about authors

Ева Максимовна ЕШИЛБАШЯН – студентка магистратуры по направлению машинного обучения факультета прикладной математики и информатики. Занимается обработкой естественного языка.

Yeva Maksimovna YESHILBASHIAN is a student of Machine Learning master's degree programme. Deals with natural language processing.

Ариана Арменовна АСАТРЯН – магистрант кафедры математической кибернетики. Научные интересы – интеллектуальный анализ текстов и машинное обучение.

Ariana Armenovna ASATRYAN is a master student at the Department of Mathematical Cybernetics. Her research interests include intelligent text analysis and machine learning.

Цолак Гукасович ГУКАСЯН является аспирантом кафедры системного программирования. Его научные интересы включают обработку естественного языка, машинное обучение.

Tsolak Gukasovitch GHUKASYAN is a postgraduate student of the Department of System Programming. His research interests include natural language processing, machine learning.

Приложение А. Список использованных признаков.

Уровень	Группа	Признаки
символы	пунктуация	• знаки пунктуации, их частота, например: ○ наличие знака пунктуации ○ #[знаки пунктуации] / #[слова] для всех знаков вместе, а также для каждого отдельно • комбинации пунктуации и пробельных символов, например: ○ наличие пробела перед знаком пунктуации ○ наличие пробела после знака пунктуации ○ #[наличие пробела перед знаком пунктуации] / #[знаки пунктуации] ○ #[наличие пробела после знака пунктуации] / #[знаки пунктуации] ○ вышеперечисленное для каждого знака пунктуации отдельно
	общие	• суффиксы: ○ наличие конкретного суффикса ○ #[слова с конкретным суффиксом] / #[слова] • префиксы: ○ наличие конкретного префикса ○ #[слова с конкретным префиксом] / #[слова]
слова	общие	○ частота слов: ○ наличие длинных слов ○ #[длинные слова] / #[слова] ○ наличие редких слов ○ #[редкие слова] / #[слова] ○ средняя частота используемых слов ○ наличие терминов на латинском/кириллице ○ наличие жаргона/неформальных выражений ○ написание именованных сущностей [1]
	сокращения	• использование сокращения вместо полной формы (например, թ. вместо թվական, թ-ի вместо թվականի) • стиль склонения сокращений (например, թ. vs թ.-ի vs թ-ի) • сокращения пишутся с маленькой буквы (например, բուհ, բnh) • сокращения пишутся с большой буквы (например, ԲՈՒՀ, ԲՈՀ) • наличие разделителя в сокращениях (դդ. vs դ.դ. или տտ. vs տ.տ.)
	числительные	• написание количественных (например, տասը հազար vs 10000) • написание порядковых (например, երկրորդ vs ii vs II vs 2-րդ vs 2րդ) • формат написания дат (например, 12\18\10 vs 12/18/10 vs 12-18-10 vs 12.18.10 vs 12\18\2010 vs 12/18/2010 vs 12-18-2010 vs 12.18.2010 vs 18 դեկտեմբերի, 2010)

	н-граммы	• биграммы: ○ наличие биграммы с низкой IDF ○ #[биграммы с низкой IDF] / (#[слов] - 1) • триграммы: ○ наличие траграммы с низкой IDF ○ #[триграммы с низкой IDF] / (#[слов] - 2)
Предло-жения	общие	• количество предложений • средняя длина предложений: ○ среднее число слов в предложении ○ среднее число символов в предложении • максимальная длина предложений: ○ максимальное число слов в предложении ○ максимальное число символов в предложении • минимальная длина предложений: ○ минимальное число слов в предложении ○ минимальное число символов в предложении • длинные предложения: ○ наличие длинных предложений ○ #[длинные предложения (с точки зрения числа слов)] / #[предложения] ○ #[длинные предложения (с точки зрения числа символов)] / #[предложения] • короткие предложения: ○ наличие коротких предложений ○ #[короткие предложения (с точки зрения числа слов)] / #[предложения] ○ #[короткие предложения (с точки зрения числа символов)] / #[предложения]
	морфологические	• части речи: ○ #[слова с этой частью речи] / #[слова] ○ наличие последовательности 2-х конкретных частей речи ○ #[последовательность 2-х конкретных частей речи] / (#[слова] - 1) ○ наличие последовательности 3-х конкретных частей речи ○ #[последовательность 3-х конкретных частей речи] / (#[слова] - 2)
	грамматические	• синтаксические связи, их частота ○ наличие синтаксической связи в параграфе (за исключением частых связей, которые повсеместны) ○ #[предложения, где присутствует синтаксическая связь] / #[предложения] • частота простых предложений: ○ наличие простых предложений ○ #[простые предложения] / #[предложения] • частота сложных предложений : ○ наличие сложных предложений ○ #[сложные предложения] / #[предложения]

Параграфы	читабельность	• Flesch reading ease • SMOG grade • Flesch-Kincaid grade • Coleman-Liau index • automated readability index • Dale-Chall readability score • difficult words • Linsear write formula • Gunning fog