ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с сентября 2010 г. DOI 10.17587/issn.2220-3397 ISSN 2220-3397

Журнал издается при поддержке Отделения математических наук РАН, Отделения нанотехнологий и информационных технологий РАН, МГУ имени М.В. Ломоносова, МГТУ имени Н.Э. Баумана

СОДЕРЖАНИЕ

Журнал зарегистрирован в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций. Свидетельство о регистрации ПИ № ФС77-38590 от 24 декабря 2009 г.

Журнал распространяется по подписке, которую можно оформить в любом почтовом отделении (индекс по Объединенному каталогу "Пресса России" — **22765**) или непосредственно в редакции.
Тел.: (499) 269-53-97. Факс: (499) 269-55-10.
Http://novtex.ru/prin/rus E-mail: prin@novtex.ru
Журнал включен в систему Российского индекса научного цитирования и базу данных RSCI на платформе Web of Science.
Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора и кандидата наук.

THEORETICAL AND APPLIED SCIENTIFIC AND TECHNICAL JOURNAL

SOFTWARE ENGINEERING

PROGRAMMNAYA INGENERIA

Vol. 10
N 1
2019

Published since September 2010 DOI 10.17587/issn.2220-3397 ISSN 2220-3397

CONTENTS

Information about the jornal is available online at:
http://novtex.ru/prin/eng e-mail:prin@novtex.ru

УДК 004.424.2 DOI: 10.17587/prin.10.3-13

С. Б. Митькин, консультант, e-mail: stipan.mitkin@gmail.com, ProsessPilotene AS, Норвегия

Автоматное программирование на языке ДРАКОН

Предложен новый метод визуального автоматного программирования на языке ДРАКОН, состоящий в совмещении конечных автоматов с деревьями принятия решений. Описана генерация программного кода из автоматных ДРАКОН-схем и способ передачи сигналов между автоматами. Рассмотрен опыт применения предлагаемого метода в реальном программном проекте.

Ключевые слова: *автоматное программирование, конечные автоматы, машины состояний, язык ДРАКОН, реактивные системы, визуальное программирование, деревья принятия решений*

Введение

Автоматное программирование облегчает моделирование сложного поведения. В работе [1] авторы рекомендуют использовать автоматный подход при создании любой программной системы, в которой есть сущности со сложным поведением. Поведение системы, как правило, прямо связано с управлением этой системой. В работе [2] авторы отмечают, что задачи управления присутствуют в любом программном обеспечении, как следствие, конечные автоматы могли бы использоваться во всех современных программах.

К настоящему времени разработаны методы построения программ с применением конечных автоматов, например, SWITCH-технология [3]. Тем не менее, в программном обеспечении автоматы встречаются редко. Причина недооценки автоматного подхода — высокая трудоемкость автоматного программирования.

Для работы с автоматами существуют библиотеки для популярных языков программирования, например, [4] и [5]. Библиотеки снижают трудозатраты, однако создаваемые с помощью библиотек автоматы сложны для чтения и анализа ввиду отсутствия визуализации.

Графические нотации, например, диаграммы состояний UML (*UML state machine*), показывают структуру автоматов в более понятном виде. Существуют инструментальные средства, например [6—8], которые генерируют программный код из диаграмм состояний. Графика облегчает анализ автоматов и снижает трудоемкость разработки по сравнению с чисто текстовым автоматным программированием. К сожалению, большинство таких средств, включая [6] и [7], требуют от разработчика писать часть автоматного кода отдельно от диаграммы, что по-прежнему создает трудности.

Диаграммы состояний UML имеют ряд недостатков. Во-первых, элементы на таких диаграммах не упорядочены, что заставляет читателя прилагать дополнительные усилия, чтобы понять структуру автомата. Во-вторых, на диаграмме состояний трудно выявить, какие именно входные сигналы приводят к тем или иным действиям автомата. Кроме того, неупорядоченность в программах со сложным поведением порождает ошибки. В работе [2] F. Wagner отмечает: "Поведение — самая сложная часть программы и поэтому источник самых запутанных ошибок".

Для предотвращения ошибок и повышения производительности труда при разработке автоматов имеет смысл обратиться к визуально упорядоченному языку ДРАКОН. В литературе по языку ДРАКОН [9] вопрос о преодолении сложностей автоматного программирования не ставится и не рассматривается. В настоящей работе впервые предложен метод визуального автоматного программирования, основанный на языке ДРАКОН. Этот метод был реализован в среде разработки DRAKON Editor [10].

В настоящей статье не рассматриваются математические и алгоритмические свойства конечных автоматов. Цель статьи — показать новый способ графического изображения автоматов.

Введение в автоматное программирование и визуальный язык ДРАКОН

В данном разделе приведены общие сведения о языке ДРАКОН и об автоматном программировании. Конечные автоматы — математическая абстракция, но здесь и далее автоматы рассмотрены с точки зрения практической разработки программ. Основы языка ДРАКОН в данном разделе изложены без связи с автоматным программированием.

Введение в автоматное программирование

Широко распространено ошибочное мнение, что автоматное программирование — узкоспециальная область, ограниченная разработкой электромеханических автоматов или других аппаратных средств. В действительности автоматное программирование представляет собой подход к разработке любого программного обеспечения вне зависимости от его назначения. Этот подход основан на применении особого рода объектов — автоматов — в качестве строительных блоков. Так же как объектно-ориентированная программа построена из классов, автоматная программа состоит из автоматов.

Классы и автоматы имеют общие черты, а именно: те и другие хранят внутри себя данные; те и другие принимают сигналы извне и реагируют на них.

Несмотря на сходство, автоматы имеют отличия от классов.

- Среди данных, хранимых в автомате, выделяют особое поле, называемое *управляющим состоянием*. Значение этого поля принадлежит некоторому конечному множеству.
- Реакция автомата на входящий сигнал задается не только типом сигнала, но и значением управляющего состояния.

У обычного класса каждый метод представляет собой процедуру. У автомата каждому методу соответствует несколько процедур. Какая именно процедура будет выполнена при вызове данного конкретного метода, зависит от текущего управляющего состояния. Иными словами, выбор процедуры определяется комбинацией типа сигнала и управляющего состояния.

Далее в статье под термином "состояние" в единственном числе понимается текущее значение, которое хранится в управляющем состоянии автомата. Термин "состояния" во множественном числе означает перечень допустимых значений, которые может принимать управляющее состояние автомата.

Возьмем элемент графического интерфейса, меняющий цвет с белого на голубой при наведении на него мыши. Можно построить автомат, который управляет цветом этого элемента. Такой автомат будет иметь два возможных состояния: "активное" и "обычное". Автомат будет принимать два вида сигналов: "движение мыши над элементом" и "мышь покинула элемент". Начальное состояние — "обычное", начальный цвет элемента — белый.

Если автомат получит сигнал "движение мыши над элементом", находясь в состоянии "обычное", он поменяет цвет элемента на голубой и переключится в состояние "активное". Если этот сигнал придет, когда автомат уже находится в состоянии "активное", автомат проигнорирует сигнал.

Если автомат получит сигнал "мышь покинула элемент", находясь в состоянии "активное", автомат поменяет цвет элемента на белый и перейдет в состояние "обычное". Если сигнал "мышь покинула элемент" поступит, когда автомат находится в состоянии "обычное", это может свидетельствовать об ошибке в других подсистемах программы. Процедура, запускаемая в состоянии "обычный", по сигналу "мышь покинула элемент" может либо проигнорировать сигнал, либо сообщить об ошибке.

В этом примере автомат, управляющий цветом, обрабатывает сигналы по-разному в зависимости от своего состояния, которое зависит от предыдущих сигналов. Можно сказать, что автомат — это класс, который переключается между несколькими возможными режимами работы.

Текущее состояние автомата — результат цепочки предыдущих событий, произошедших с автоматом, поэтому состояние представляет собой связь с прошлым автомата. Эта связь указывается в явном виде, причем прошлое получается сжатым в одну переменную. Именно такая компактность представления прошлого делает автоматы удобным инструментом моделирования поведения.

Автоматный подход хорошо масштабируется, когда сложность поведения систем возрастает. Масштабируемость достигается построением сетей, состоящих из многих простых автоматов.

Автомат — мощная и в то же время простая программная модель. Автоматы могут помочь в решении задач реализации поведения в самых разных видах программного обеспечения.

Основы языка ДРАКОН

Язык ДРАКОН — визуальный язык представления алгоритмов. ДРАКОН-схемы имеют сходство с блок-схемами, однако в языке ДРАКОН есть ряд упорядочивающих правил, направленных на облегчение понимания диаграмм. Вот некоторые из этих правил.

- Пересечения линий запрещены.
- Разрешены только прямые вертикальные и горизонтальные линии.
- Течение времени на диаграмме направлено сверху вниз.
- Ветвление происходит только вправо.
- Разрешен только один вход в цикл.

В языке ДРАКОН есть визуальные аналоги конструкций структурного программирования — `if-else`, `switch-case`, `foreach-break` и т. п. Можно сказать, что язык ДРАКОН имеет такие же преимущества перед традиционными блок-схемами, как структурное программирование перед программированием на основе оператора `goto`.

Из ДРАКОН-диаграмм можно генерировать исполняемый или программный код. Для этого в иконах помещают выражения на некотором языке программирования. В настоящей работе в качестве такого языка программирования выбран язык JavaScript. В выражениях внутри икон не должно быть ключевых слов `if-else`, `switch-case` и т. п. Роль таких ключевых слов играют конструкции языка ДРАКОН.

Для линейных участков алгоритма применяют иконы "действие" (рис. 1).

Для вопросов, на которые можно ответить "да" или "нет", применяют икону "вопрос" (аналог оператора `if-else`, рис. 2).

Для вопросов, на которые может быть несколько ответов, применяют макроикону "выбор" (аналог оператора `switch-case`), состоящую из одной иконы "выбор" и двух или более икон "вариант". Макроикона "выбор" на рис. 3 имеет три иконы "вариант". Пустая икона "вариант" справа означает "все остальные значения".

В языке ДРАКОН есть тип диаграмм, называемый *силуэт*. Силуэт разбивает большой алгоритм на несколько малых алгоритмов. Каждый из таких малых алгоритмов заключен в отдельной *ветке силуэта*. Вверху ветки силуэта находится икона "имя ветки".

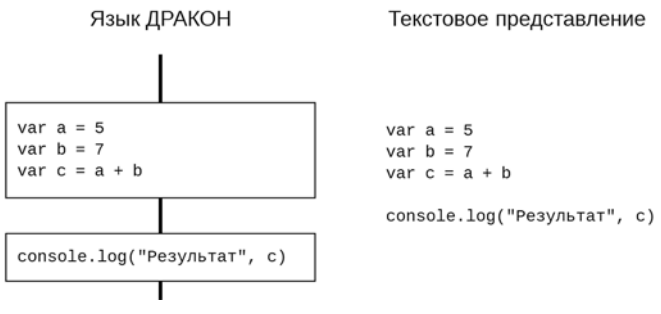


Рис. 1. Иконы "действие" на линейном участке алгоритма

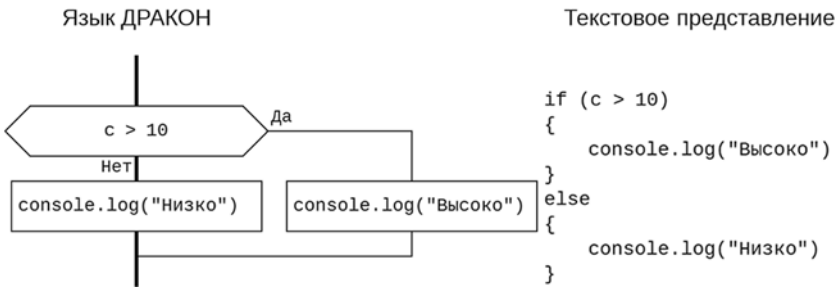
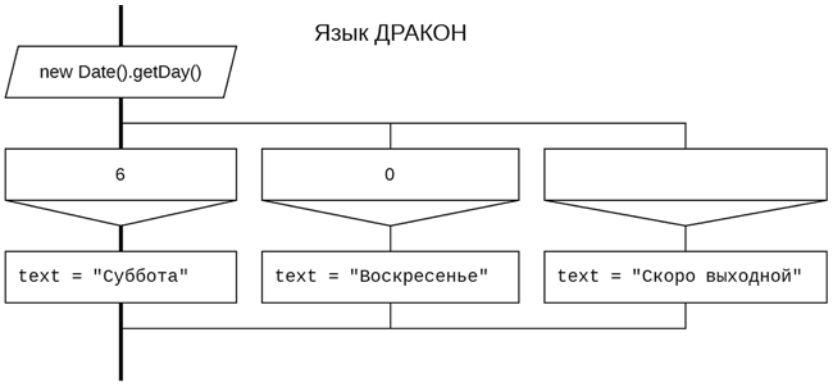


Рис. 2. Икона "вопрос" и ветвление алгоритма



Текстовое представление

```
switch (new Date().getDay()) {
    case 6:
        text = "Суббота"
        break
    case 0:
        text = "Воскресенье"
        break
    default:
        text = "Скоро выходной"
}
```

Рис. 3. Макроикона "выбор" и вопрос с несколькими ответами

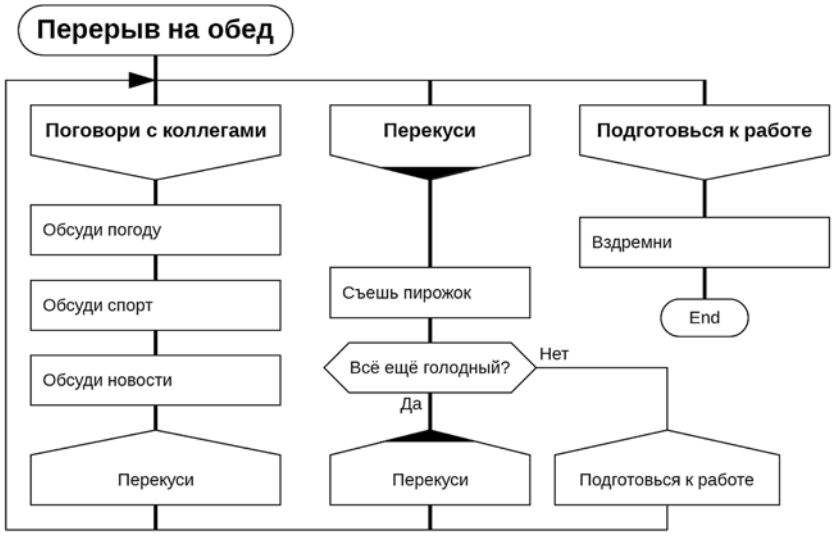


Рис. 4. Пример ДРАКОН-схемы "силуэт"

Ниже следует тело ветки, содержащее малый алгоритм — часть общего алгоритма. Внизу расположена одна или несколько икон "адрес", которые указывают на следующие ветки силуэта.

Самая первая ветка расположена слева. Порядок следования веток задается иконами "адрес", при этом существует соглашение упорядочивать ветки слева направо. Некоторые ветки могут выполняться несколько раз. Ветка с выходом из алгоритма, если выход есть, всегда находится справа. На рис. 4 представлена ДРАКОН-схема "силуэт", в которой есть три ветки: "Поговори с коллегами", "Перекуси", "Подготовься к работе".

Язык ДРАКОН представляет собой графический аналог структурного программирования, упорядоченный в целях повышения читаемости диаграмм. ДРАКОН-схемы пригодны для разработки программного обеспечения — из них можно получать работающие программы. Одной из особенностей языка ДРАКОН является тип диаграмм "силуэт", который разбивает задачу на логические части.

Изображение автоматов с помощью языка ДРАКОН

В данном разделе описана суть предлагаемого графического способа изображения автоматов. Вначале приведено формальное описание представления автоматов на языке ДРАКОН, затем дан пример автоматной ДРАКОН-схемы и программный код, сгенерированный из этой схемы. Для генерации кода применялась среда разработки DRAKON Editor, в которую автор добавил поддержку автоматных диаграмм.

Формальное описание изображения автоматов на языке ДРАКОН

В предлагаемом методе изображения автоматов применяется подмножество языка ДРАКОН:

- используются только схемы типа "силуэт", схемы "примитив" не используются;
- в теле ветки схемы "силуэт" допускаются следующие иконы языка ДРАКОН: "действие", "вопрос" и макроикона "выбор".

Графический синтаксис языка ДРАКОН остается без изменений. Способ расположения элементов на схеме не меняется. Меняется семантика схемы "силуэт".

Ветки на обычных, не автоматных схемах "силуэт" обозначают последовательно исполняемые участки алгоритма. Начало ветки (икона "имя ветки") не подразумевает остановки алгоритма.

Ветки на автоматных схемах "силуэт" означают возможные состояния автомата. Каждая ветка соответствует одному возможному состоянию автомата. Переход в начало ветки означает переключение в соответствующее состояние, приостановку алгоритма работы автомата и ожидание входящего сигнала в этом состоянии.

Пусть автомат M задается пятеркой следующих элементов:

$$M = \{V, Q, q_0, F, \delta\},$$

где V — входной алфавит; Q — множество возможных состояний; q_0 — начальное состояние; F — множество конечных состояний ($F \subset Q$); δ — функция переходов, δ: $V \times Q \to Q$.

Каждому состоянию $q_i \subset Q$ соответствует одна ветка силуэта. Самая левая ветка — начальное состояние q_0. Если силуэт не имеет ветки с иконой "конец", множество конечных состояний пусто. Если ветка с иконой "конец" есть, то такая ветка расположена справа на схеме. Данная конечная ветка означает единственное конечное состояние $f \subset Q$.

В начале каждой ветки, кроме ветки с иконой "конец", находится макроикона "выбор" с ключевым словом "receive". Эта макроикона ставится непосредственно под иконой "имя ветки". В ветке может быть только одна макроикона "выбор" с ключевым словом "receive".

Иконы "вариант" сразу под иконой "выбор" с ключевым словом "receive" задают типы входящих сигналов $v \subset V$, которые автомат принимает в данном состоянии q_i. Такие иконы "вариант", расположенные на разных ветках, могут задавать различные подмножества принимаемых сигналов. Непустые иконы "вариант" на всех ветках вместе задают входной алфавит, или множество входных символов (сигналов) V автомата. Пустая икона "вариант" означает "любой из сигналов, упомянутый на других ветках".

Тела веток силуэта задают функцию переходов δ: $V \times Q \to Q$, $\delta(v_j, q_i) \to q_k$, где входной символ v_j обозначается иконой "выбор", входное состояние q_i обозначается иконой "имя ветки", а выходное состояние q_k — иконой "адрес".

Участок диаграммы под каждой иконой "вариант" представляет собой обработчик одного типа сигнала в состоянии, соответствующем данной ветке. Обработчики сигналов задают действия автомата при получении сигналов. В обработчиках сигналов могут быть иконы "действие", а также средства ветвления — иконы "вопрос" и макроиконы "выбор" без ключевого слова "receive". Благодаря ветвлению, из одной иконы "вариант" можно попасть в несколько икон "адрес", поэтому обработчики сигналов представляют собой деревья принятия решений.

Таким образом, при данном способе визуализации наиболее важные компоненты автомата присутствуют на одном чертеже: перечень состояний автомата задается именами веток силуэта, входной алфавит определяется иконами "выбор", а функция переходов содержится в телах веток.

Пример автоматной ДРАКОН-схемы и сгенерированного из нее кода

Среда разработки DRAKON Editor генерирует исходный код из ДРАКОН-диаграмм. По состоянию на октябрь 2018 г. поддерживается 13 языков программирования.

В обычном режиме из одной ДРАКОН-схемы получается одна функция. Для некоторых языков, включая JavaScript, Python, C# и Lua, DRAKON Editor генерирует не только обычные функции, но и конечные автоматы. Сгенерированные автоматы построены в виде классов на выбранном языке программирования. Из одной автоматной ДРАКОН-схемы получается один класс — один автомат.

На рис. 5 представлена автоматная ДРАКОН-схема на гибридном языке ДРАКОН-JavaScript. Чтобы указать, что ДРАКОН-схема является автоматной, в первой строке иконы "параметры" помещают ключевую фразу "state machine" (см. прямоугольник в левой верхней части рис. 5). DRAKON Editor строит автоматы только из ДРАКОН-схем "силуэт". Для схем "примитив" генерация автоматного кода не предусмотрена.

Имя класса берется из названия автоматной ДРАКОН-схемы. Названия состояний берутся из названий веток. В названии схемы и названиях веток не должно быть пробелов или символов пунктуации, так как эти названия войдут в состав идентификаторов в сгенерированной программе.

В состоянии `Foo` автомат обрабатывает сигналы `yellow` и `red`, а в состоянии `Bar` — `yellow` и `black`. Таким образом, автомат `Abc` принимает три вида сигналов: `yellow`, `red` и `black`. В состоянии `Exit` автомат выключен и не принимает сигналов.

DRAKON Editor сгенерирует из этой диаграммы класс `Abc` с тремя методами: `yellow`, `red` и `black`. Все эти методы имеют одинаковую сигнатуру — они принимают один аргумент `msg`, который задан в иконе "Параметры". Клиентский код может использовать класс `Abc` так:

```
var machine = new Abc()
machine.yellow(msg1)
machine.black(msg2)
machine.red(msg3)
```

Рассмотрим состояние `Foo` (первое слева на рис. 5). В состоянии `Foo` автомат принимает сигналы `yellow` и `red`. Если вызвать метод `black` в этом состоянии, ничего не случится. Вызов будет проигнорирован без генерации исключения. Важно иметь возможность игнорировать ненужные сигналы. Если в автомат приходит сигнал, который не обрабатывается в текущем состоянии, это не ошибка. Игнорирование сигналов — полезный элемент поведения автоматов. Если же приход некоторого сигнала считается ошибкой, следует создать для этого сигнала обработчик и предпринять там соответствующие явные действия.

Обработчик сигнала `yellow` в состоянии `Foo` вызывает функцию `goLeft(self, msg)` и переключает автомат в состояние `Bar`, если значение `msg.x` больше 100. Если значение `msg.x` меньше или равно числу

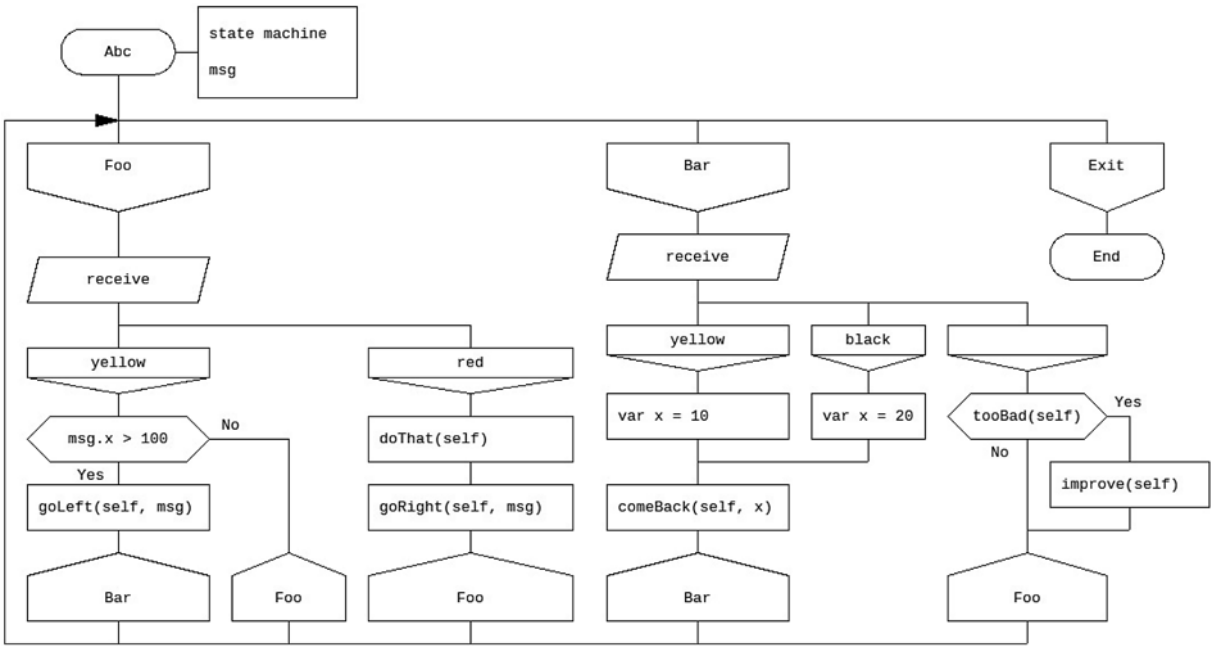


Рис. 5. Схема конечного автомата `Abc`, который имеет состояния `Foo`, `Bar` и `Exit`

100, обработчик сигнала `yellow` не совершает никаких действий. Обработчик сигнала `red` вызывает функции `doThat(self)` и `goRight(self, msg)` и не переключает состояние автомата. `goLeft`, `doThat`, `goRight` и другие функции должны быть определены в другом месте, вне автомата `Abc`.

Переменная `self` указывает на объект автомата, она доступна во всех обработчиках.

Перейдем к состоянию `Bar` (средняя ветка на рис. 5). В состоянии `Bar` автомат обрабатывает сигналы `yellow` и `black`. Хотя пути обработчиков сигналов `yellow` и `black` соединяются внизу диаграммы, DRAKON Editor построит разные функции для этих сигналов, причем совпадающий участок алгоритмов будет скопирован. Поэтому локальная переменная `x` должна быть объявлена два раза — по одному разу в каждом обработчике.

Пустая икона "вариант" означает "все остальные сигналы". Для состояния `Bar` "все остальные" — это сигнал `red`. При вызове метода `red` в состоянии `Bar` будет запущен обработчик, который начинается с пустой иконы "вариант".

Пустую икону "вариант" можно также использовать, когда в некотором состоянии автомат принимает только один вид сигналов. Макроикона "выбор" с одной иконой "вариант" запрещена, но если к единственной иконе "вариант" добавить пустую икону "вариант", генератор кода посчитает такую макроикону "выбор" корректной.

Генератор кода анализирует ДРАКОН-схему и составляет список состояний и список типов принимаемых сигналов (входной алфавит автомата).

Список состояний для автомата `Abc` на рис. 5: `Foo`, `Bar`, `Exit`.

Список типов сигналов для автомата `Abc`: `black`, `red`, `yellow`.

Затем генератор создает тело класса, в который для каждого типа сигналов добавляет метод. В теле метода происходит выбор конкретного обработчика исходя из текущего состояния. В итоге обработчик определяется исходя из двух входных данных: типа сигнала и текущего состояния (*double dispatch*). Все обработчики принимают одни и те же аргументы. Это аргумент `self` и аргументы, перечисленные в иконе "параметры" ниже строки с ключевой фразой "state machine".

Генератор добавляет в класс поле `state`, которое хранит название текущего состояния автомата в виде строки. При создании автомата в поле `state` записывается начальное состояние автомата.

Строковое поле `state` удобно для тестирования и отладки. Если поле `state` содержит `null`, автомат считается выключенным и не реагирует на сигналы. Автомат можно выключить в клиентском коде принудительно, если записать `null` в поле `state`:

```
machine.state = null
```

Возможны и другие, более эффективные, способы хранения текущего состояния и выбора конкретного обработчика. Данный способ — один из наиболее простых в языках JavaScript и Lua.

Если на ДРАКОН-схеме в каком-то состоянии есть пустая икона "вариант", то для тех типов сигналов, которые не указаны явно в иконах "вариант" для данного состояния, будет сгенерирован и вызван обработчик по умолчанию. Например, в состоянии `Bar` класса `Abc` явно указаны события `yellow` и `black`, а событие `red` не указано, но есть пустая икона "вариант". Поэтому в методе `red` для состояния `Bar` вызывается обработчик `Bar` по умолчанию: `Abc _ Bar _ default`.

Тело класса для автомата Abc:

```
function Abc() {
   var _self = this;
   _self.type_name = "Abc";
   _self.state = "Foo";
   _self.black = function(msg) {
      var _state_ = _self.state
      if (_state_ == "Bar") {
         Abc_Bar_black(_self, msg)
      }
   }
   _self.red = function(msg) {
      var _state_ = _self.state
      if (_state_ == "Foo") {
         Abc_Foo_red(_self, msg)
   }
      else if (_state_ =="Bar") {
         Abc_Bar_default(_self, msg)
      }
   }
   _self.yellow = function(msg) {
      var _state_ = _self.state
      if (_state_ == "Foo") {
         Abc_Foo_yellow(_self, msg)
   }
      else if (_state_ == "Bar") {
         Abc_Bar_yellow(_self, msg)
      }
   }
}
```

Для каждого из сочетаний состояние — сигналы, которые присутствуют на диаграмме, генератор создает функцию-обработчик. В конце обработчика генератор добавляет код, который переключает автомат в следующее состояние. Начиная с версии генератора 1.32, обработчики могут возвращать значение с помощью ключевого слова return.

Пример: обработчик сигнала yellow для состояния Foo в классе Abc:

```
function Abc_Foo_yellow(self, msg) {
   if (msg.x > 100) {
      goLeft(self, msg)
      self.state = "Bar"
   } else {
      self.state = "Foo"
   }
}
```

Как видно из примера, из ДРАКОН-схемы автоматически генерируется не только базовая структура автомата, но и код обработчиков сигналов, включая алгоритм выбора следующего состояния. Генератор кода формирует весь исходный код автомата, так что нет необходимости дописывать код вручную.

Особенности применения автоматного программирования

Автор использовал средства визуального автоматного программирования среды DRAKON Editor при создании коммерческого веб-приложения DrakonHub [11]. К сожалению, создать приложение полностью в автоматном стиле не получилось, так как эти средства появились, когда часть кода данного веб-приложения уже была написана. Тем не менее, автору удалось опробовать предлагаемый метод на практике и получить обратную связь от применения этого метода в реальном программном проекте. Данный раздел содержит описание опыта, полученного в ходе выполнения этого проекта.

Решаемые задачи

Исходный код веб-приложения DrakonHub насчитывает 51 конечный автомат на языке ДРАКОН. Автоматное программирование на языке ДРАКОН показало свою пригодность для решения перечисленных далее задач.

- Асинхронный ввод/вывод, например, вызов веб-сервисов.
- Обработка действий пользователя, например, событий мыши, сенсорного экрана и клавиатуры.
- Мастера (wizards), т. е. графические диалоги, состоящие из последовательности экранов. Пример: диалог создания диаграммы, который состоит из двух шагов: выбора типа диаграммы и ввода названия диаграммы.
- Работа с таймером и управление длительными процессами. Пример: поиск по содержимому во многих диаграммах.
- Управление событиями в графическом интерфейсе пользователя, включая события нескольких связанных виджетов со сложным поведением. Пример: координация виджета редактора диаграмм и событий на сервере при одновременном редактировании диаграмм несколькими пользователями.
- Обход деревьев и графов. Пример: обход элементов диаграммы для поиска возможных мест, куда можно пересадить лиану (ребро ДРАКОН-схемы) без пересечения линий.
- Лексический анализ (разбор строк на лексемы).

Код приложения DrakonHub является закрытым. Для верификации практической применимости предлагаемого подхода можно обратиться к демонстрационному приложению Лифт [12].

Автоматное программирование на языке ДРАКОН хорошо себя зарекомендовало в событийно-ориентированной среде браузера и сервера приложений tarantool. Автор предполагает, что автоматы на основе ДРАКОН-схем будут также успешно работать и в других окружениях, например, в сервере приложений NodeJs.

Сеть автоматов

Одиночный автомат полезен только для ограниченного круга задач, например, для некоторых случаев обхода графов и простого лексического анализа. Чаще всего задачу решают с помощью нескольких взаимодействующих автоматов. Разбиение программы на множество автоматов позволяет работать с простыми и понятными автоматами, каждый из которых легко разработать и протестировать.

Существуют графические нотации, изображающие несколько автоматов на одной диаграмме, например, диаграммы состояний UML с иерархическими автоматами [13]. Проблемный вопрос с такими нотациями состоит в том, что сложность моделируемой системы ограничена размерами диаграммы.

Предлагаемый способ автоматного программирования не изображает дочерние автоматы на одной диаграмме с родительским. Граф автоматов строится отдельно, вне автоматных диаграмм, и как следствие, может быть довольно большим. Благодаря этому на размер моделируемой системы не налагается искусственных ограничений.

Способ организации взаимодействия автоматов имеет существенное значение. В процессе разработки приложения DrakonHub стало ясно, что сеть автоматов без ограничений на топологию трудно отлаживать. Практически действенным оказалось решение, описанное в работе [2]: упорядочить автоматы в виде дерева. Объединять все автоматы в программе в одно дерево не обязательно. Можно иметь набор деревьев или лес автоматов.

Требование выстраивать автоматы в виде дерева в некоторых случаях представляется слишком жестким. Сама необходимость упорядочивать автоматы возникает вследствие того, что во время выполнения некоторого метода автомата `A` другой автомат `B` может вызвать этот же или другой метод автомата `A`. Это может привести к ошибкам в силу разрушения инкапсуляции. Такая нежелательная ситуация возникает как с автоматами, так и с классами. Чтобы избежать этой ситуации, достаточно того, что автоматы образуют ориентированный граф без циклов. Те автоматы, которые находятся "выше" в графе, могут напрямую вызывать методы своих непосредственных соседей "ниже по течению" графа. Автоматы, которые находятся "ниже" в графе, посылают сигналы своим "верхним" соседям только опосредованно, через цикл событий (*event loop*).

Повышение сложности системы должно выражаться не в увеличении размера автоматов, а в росте их числа. Автоматы объединяются в сеть, причем важно, что сеть автоматов упорядочена в виде ориентированного графа без циклов. Каждый автомат изображается на отдельной диаграмме, а автоматная сеть строится вне автоматных диаграмм. Такой подход хорошо масштабируется и позволяет создавать системы с высоким уровнем сложности.

Размер диаграмм

Автоматные диаграммы в реальных приложениях могут быть большими, более 4 тыс. пикселей по горизонтали. Чтобы увидеть весь автомат, приходится прокручивать ДРАКОН-схему. Высота же автоматных диаграмм "силуэт" остается небольшой, не больше высоты экрана. Кроме того, в большинстве случаев можно добиться того, что одно состояние автомата полностью помещается на экране. Таким образом, значительный размер диаграмм не является препятствием.

Привлекательность автоматного программирования состоит в том числе и в том, что объект со сложным поведением разбивается на небольшие, относительно несложные части. Причем в один момент времени можно рассматривать только одну часть. Автоматные ДРАКОН-схемы способствуют именно такому подходу, так как элементы, связанные с одним состоянием, расположены в одной области диаграммы.

Анализ автоматов, изображаемых на языке ДРАКОН

Существуют различные виды автоматов и различные способы организации их работы. В данном разделе подчеркнуто отличие предлагаемого метода от других способов визуализации конечных автоматов, указано место ДРАКОН-автоматов в классификации автоматов, приведен перечень автоматных событий, или действий, которые можно отобразить на ДРАКОН-схеме, а также рассмотрена работа ДРАКОН-автоматов в средах, основанных на событиях, и в средах, проводящих регулярный опрос своих компонентов.

Деревья принятия решения

Отличительная особенность автоматных диаграмм на языке ДРАКОН — сочетание деревьев принятия решений и автоматной логики.

На автоматной ДРАКОН-схеме выбор реакции автомата делается в два шага. Первый шаг — тот же, что и в диаграммах состояний UML. Исходя из текущего состояния и входного сигнала выбирается обработчик сигнала. Второй, дополнительный шаг выбора осуществляется деревом принятия решений, которое находится в обработчике. Например, при поступлении в автомат на рис. 5 сигнала `yellow` в состоянии `Foo` сначала выбирается обработчик в левой нижней части диаграммы. Затем обработчик в зависимости от значения `msg.x` либо вызывает процедуру `goLeft` и переключает автомат в состояние `Bar`, либо оставляет автомат в состоянии `Foo`.

В литературе автоматы часто упоминаются вместе с деревьями принятия решений: деревья противопоставляются автоматам; деревья служат для иллюстрации автоматов; автоматы применяются для обхода деревьев и т. п. Новизна данного метода состоит в совместном использовании автоматов и деревьев принятия решений на одной диаграмме.

Автоматы Мили

Конечные автоматы, изображаемые на языке ДРАКОН, можно отнести к автоматам Мили. Выходная последовательность автомата Мили зависит от состояния автомата и входных сигналов. Именно так и работают автоматы, построенные в среде DRAKON Editor. Выбор обработчика происходит на основании двух факторов: типа входящего сигнала и состояния автомата.

Действия автомата, которые можно изобразить на ДРАКОН-схеме

Некоторые авторы [2] выделяют следующие автоматные события — действия, которые автомат выполняет за один цикл работы.

- **Входное действие** — процедура, которая выполняется, когда автомат входит в состояние. Для каждого состояния может быть одно входное действие.
- **Выходное действие**, которое выполняется, когда автомат покидает состояние. Для каждого состояния может быть одно выходное действие.
- **Действия перехода**, которые уникальны для каждой направленной пары состояний.
- **Действие обработки ввода**, которое автомат проводит при получении сигнала. Действия обработки ввода разные для разных комбинаций состояния и типов входного сигнала.

Программа DRAKON Editor версии 1.31 не поддерживает входные действия на автоматных ДРАКОН-схемах. Тем не менее принципиальная возможность добавить эти действия существует. Входные действия можно расположить над макроиконой "выбор" с ключевым словом "receive".

Выходные действия автоматные ДРАКОН-схемы не поддерживают совсем.

Действия перехода поддерживаются, но лишь частично. Часть алгоритма прямо над иконой "адрес" будет выполнена перед переключением из текущего состояния в следующее. Однако из одного состояния в другое может вести несколько икон "адрес". Это не соответствует требованию к действиям перехода быть уникальными для каждой пары состояний.

Автоматные диаграммы на языке ДРАКОН поддерживают действия обработки ввода в полной мере. Участки ДРАКОН-схемы, которые начинаются от икон "выбор" сразу под словом "receive", задают отдельные самостоятельные алгоритмы. Эти алгоритмы запускаются входящими сигналами. Комбинация состояния и типа сигнала определяет, который из таких алгоритмов следует выполнить.

На автоматной ДРАКОН-схеме можно изобразить только два из четырех автоматных действий. Это является ограничением предлагаемого метода. Вместе с тем самое важное для многих автоматов действие — действие обработки ввода — с помощью языка ДРАКОН отображается.

Событийно-ориентированное программирование и регулярный опрос

Автоматы могут работать в различных режимах, включая следующие.

- Событийно-ориентированное программирование ("реактивные" системы). В среде, которая реагирует на события, обработчик сигнала запускается, только когда приходит сигнал.
- Регулярный опрос (*polling*). Среда регулярно опрашивает автомат, вызывая процедуру обновления. Процедура обновления автомата проверяет наличие входных условий и, если условия выполняются, проводит соответствующие действия.

Автоматы в среде DRAKON Editor изначально создавались для событийно-ориентированного программирования, где при возникновении событий среда посылает автоматам сигналы. Чтобы послать сигнал автомату, в обработчике события нужно вызвать метод автоматного объекта для данного типа сигналов.

Работа в режиме регулярного опроса также возможна. Для этого автоматные ДРАКОН-программы должны принимать один сигнал `update`, обработчик которого и будет процедурой обновления автомата.

Таким образом, с помощью языка ДРАКОН можно создавать автоматы как для сред, основанных на событиях, так и для сред, работающих в режиме регулярного опроса. Однако наиболее удобно применять предлагаемый метод в рамках событийно-ориентированного программирования.

Преимущества автоматного программирования на языке ДРАКОН

Предлагаемый метод визуализации автоматов был создан в целях облегчения разработки программ в автоматном стиле. В данном разделе приведено детальное описание выгод, которые приносит представление автоматов на языке ДРАКОН.

Преимущества деревьев принятия решений

Деревья принятия решений позволяют получить следующие полезные эффекты.

- Увеличение выразительной силы диаграмм. ДРАКОН-схема позволяет чередовать проверку условий и действия в дереве принятия решений (рис. 6). Такое чередование нельзя изобразить на диаграмме состояний UML.
- Возможность визуально проследить путь через дерево принятия решений. Эта возможность увеличивает наглядность автоматной диаграммы, облегчает работу с ней и снижает риск возникновения ошибок.
- Помощь в устранении противоречивых условий. Условия перехода между состояниями на диаграмме состояний UML записывают в виде текстовых логических формул. При этом можно указать неполные или противоречащие условия. Особенно легко допустить такую ошибку, когда условия состоят из сложных логических выражений (рис. 7). Деревья принятия решений ввиду своей визуальной природы сводят возможность таких ошибок к минимуму.
- Привлекательность для широкого круга разработчиков. Многие разработчики не принимают автоматное программирование в силу того, что оно слишком сильно отличается от привычных методов создания программ. Деревья принятия решений близки к традиционному структурному программированию, и поэтому автоматное программирование на языке ДРАКОН имеет шансы на более широкое распространение, чем диаграммы состояний UML.

Эти преимущества усиливаются тем, что деревья принятия решений в автоматных ДРАКОН-схемах изображаются на одной визуальной сцене с другими частями автомата. Не требуется переключаться между разными документами и типами диаграмм, чтобы полностью охватить автомат.

Наличие деревьев принятия решений повышает выразительность автоматных диаграмм. Кроме того, деревья принятия решений делают автоматы более легкими для понимания.

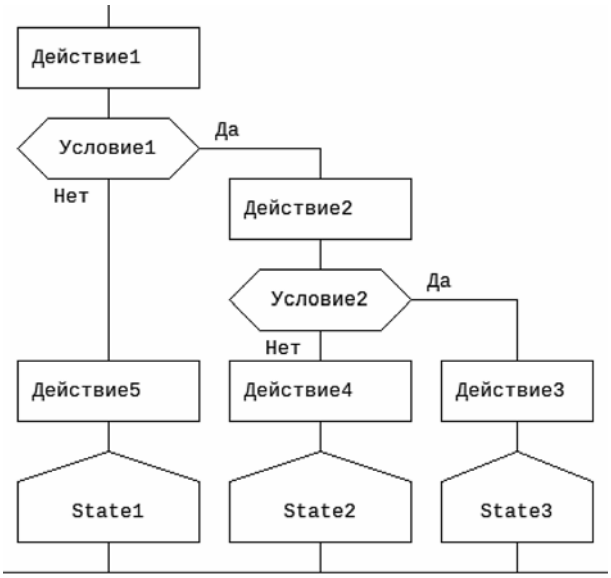


Рис. 6. Дерево принятия решения, в котором чередуются условия и действия

Избавление от "технических" состояний, не несущих смысла для читателя диаграммы

Возможность чередовать проверку условий и действия в деревьях принятия решений дает дополнительный выигрыш: алгоритм обработчика может сделать несколько последовательных вызовов методов дочерних автоматов. В результате появляется возможность избавиться от многих "технических" состояний, которые образуются при ожидании ответа от дочерних автоматов.

Например, автомат `Scheduler` (планировщик) из работы [12] имеет два рабочих состояния: `Up` и `Down`. Эти состояния соответствуют двум направлениям движения кабины лифта: вверх и вниз. Нет необходимости добавлять дополнительные состояния для ожидания ответа от дочернего автомата `Buttons`.

Благодаря этому каждое состояние автомата является конкретным наблюдаемым режимом работы. Такие конкретные режимы работы легко выявлять на этапе разработки автоматов.

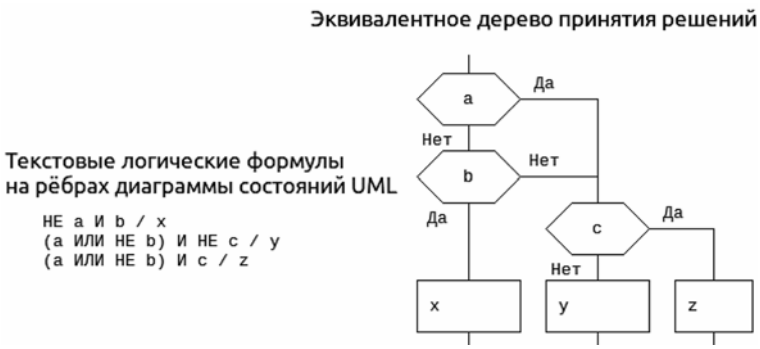


Рис. 7. Текстовые логические формулы на ребрах диаграммы состояний UML и эквивалентное им дерево принятия решений

Наглядность автоматов

Правила языка ДРАКОН эргономически выверены, и поэтому делают автоматные диаграммы легкими для прочтения. Например, чтобы определить, из каких веток состоит силуэт, не нужно изучать всю диаграмму. Для этого достаточно просмотреть самый верх схемы, так как заголовки веток выровнены по горизонтали и расположены вверху. В результате одного взгляда на верх автоматной ДРАКОН-схемы достаточно, чтобы увидеть состояния автомата.

Для сравнения: чтобы увидеть состояния на диаграмме состояний UML, схему требуется просмотреть полностью.

Иконы "вариант", задающие типы сигналов, всегда находятся вверху ветки, и это тоже облегчает чтение автоматной диаграммы. Во-первых, набор принимаемых сигналов оформлен в виде списка, который выстроен по горизонтали. Во-вторых, читатель заранее знает, где искать этот список — наверху ветки.

Для сравнения: чтобы увидеть входные сигналы на диаграмме состояний UML, необходимо отследить все ребра, исходящие из иконы выбранного состояния, причем ребра могут идти в произвольных направлениях. Затем нужно найти подпись на каждом ребре и выделить из нее условие.

Иконы "адрес" выровнены по горизонтали и расположены внизу диаграммы. Благодаря этому можно узнать, в какие состояния автомат может перейти из любого выбранного состояния, без изучения всей ДРАКОН-схемы.

Обработчики сигналов обязательно заканчиваются иконами "адрес". Данная нотация исключает возможность забыть указать следующее состояние в алгоритме обработчика.

Так как начальное состояние на автоматной ДРАКОН-схеме всегда расположено слева, алгоритм инициализации, если он есть, тоже расположен слева. Это — еще один фактор, который способствует единообразию автоматных диаграмм.

Язык ДРАКОН налагает на диаграммы строгие правила. Одно из таких правил — запрет на пересечение линий. Отсутствие пересечений устраняет целый класс запутанных диаграмм и связанных с ними ошибок.

Другое правило состоит в том, что время в ветке силуэта течет сверху вниз, а ветвление идет только вправо. Это правило дает деревьям принятия решений предсказуемость. Читатель находит следующий шаг алгоритма там, где ожидает.

Кроме того, упорядоченность ветки силуэта в направлении сверху вниз вводит стандартную последовательность прочтения состояния в автомате.

- На самом верху — название состояния.
- Чуть ниже — сигналы, которые автомат принимает в этом состоянии.
- Еще ниже — обработчики сигналов.
- В самом низу — следующие состояния, в которые можно перейти из данного состояния.

Несмотря на то, что автоматные ДРАКОН-схемы могут достигать значительных размеров по ширине, каждое конкретное состояние обычно помещается на экран. Все элементы автомата, связанные с одним состоянием, расположены недалеко друг от друга на диаграмме. Это облегчает концентрацию внимания на одном состоянии в один момент времени, что позволяет применять автоматное программирование максимально эффективно.

Наглядность — понятие во многом субъективное, однако описанные выше меры повышения читаемости автоматных диаграмм действуют вне зависимости от индивидуальных предпочтений разработчиков. Упорядоченное расположение элементов на схеме воспринимается лучше, чем хаотичное, а графическая группировка частей автомата по принадлежности к состоянию помогает чтению диаграммы больше, чем случайный разброс этих частей. Все вместе эти упорядочивающие меры повышают комфорт разработчика и сокращают вероятность ошибок.

Отсутствие необходимости дописывать автоматный код вручную

Чем больше рабочего программного кода можно получить из диаграммы, тем лучше. И наоборот, чем больше кода приходится дописывать вручную, тем менее очевидна выгода от визуального программирования.

Многие инструменты визуального автоматного программирования, например [6] и [7], генерируют лишь часть кода, составляющего автомат. Эти инструменты производят только код, который составляет структуру автомата. Содержательную часть автомата, например, обработчики сигналов, программист должен написать в текстовом виде в отдельном файле.

Порядок работы с такими инструментами таков.

- Разработчик рисует автоматную диаграмму.
- Инструмент генерирует код структуры автомата.
- Разработчик пишет код реакций автомата в текстовых файлах. Связывание написанного кода со структурой автомата происходит либо неявно, на основе соглашений по именованию функций или классов, либо явно, с помощью связующего кода.

Для автоматных ДРАКОН-программ порядок работы иной.

- Разработчик рисует автоматную диаграмму.
- Инструмент генерирует весь код автомата. Нет необходимости писать и связывать дополнительный код — автомат готов к работе.

Таким образом, автоматное программирование на языке ДРАКОН дает очевидную выгоду: разработчик более не вынужден редактировать дополнительный файл с исходным кодом при создании и доработке автомата. Такая выгода повышает удобство и производительность труда разработчика.

Визуальная подсказка в тестировании

Одна из сильных сторон автоматного программирования — ясная стратегия тестирования. Большая программа со сложным поведением строится как система автоматов. Каждый автомат тестируется отдельно. В автомате последовательно перебирают состояния и проверяют реакции автомата в каждом из состояний.

ДРАКОН-схема предоставляет визуальную помощь в следовании этой стратегии. Чтобы перебрать состояния автомата, достаточно прочитать заголовки веток силуэта. Это просто, так как заголовки веток выровнены, как горизонтальный список. Чтобы перебрать типы входящих сигналов для выбранного состояний, следует пройтись по иконам "вариант" вверху соответствующей ветки. Иконы "вариант" — это тоже горизонтальный список. Одна икона "вариант" — один обработчик входящих сигналов.

Далее нужно составить тестовые сценарии для каждого пути через обработчик. Обработчик на ДРАКОН-схеме представляет собой дерево принятия решений. Чтобы построить тест, нужно визуально проследить каждый путь через дерево и записать шаги, из которых состоит этот путь. Графическое представление дерева делает такое прослеживание тривиальным.

Эти визуальные подсказки помогают при написании тестов и облегчают тестирование. Легкость тестирования способствует более полному покрытию программы тестами, что повышает надежность программы.

Заключение

Предложен новый метод визуального автоматного программирования на языке ДРАКОН. Данный способ повышает производительность труда разработчика и имеет ряд преимуществ по сравнению с другими методами построения автоматов.

ДРАКОН-схема "силуэт" изображает на одной визуальной сцене все части автомата: состояния, типы принимаемых сигналов, обработчики сигналов и переходы. При этом относящиеся к одному состоянию элементы находятся в одной области на диаграмме. Это облегчает анализ и редактирование автоматов, а также сокращает число ошибок.

Автоматная ДРАКОН-схема содержит всю необходимую информацию для генерации программного кода. Из ДРАКОН-схемы можно сгенерировать не только код, который задает структуру автомата, но и содержательный код обработчиков. Не требуется вручную дописывать исходный код автомата в отдельном файле.

Достоинство автоматных ДРАКОН-схем — предсказуемость и единообразие, которые улучшают читаемость диаграмм. Язык ДРАКОН предоставляет соглашение, которому должны следовать диаграммы. Это соглашение упорядочивает автоматы и дает автоматам единую структуру.

Принципиальное отличие автоматных ДРАКОН-схем от других графических нотаций для автоматного программирования состоит в совмещении автоматного выбора действий с деревьями принятия решений. Деревья принятия решений делают автоматы более гибкими и понятными для широкого круга разработчиков.

Благодаря деревьям принятия решений сокращается число "технических" состояний автомата, не несущих смысла для человека, что облегчает проектирование программ.

Автоматные ДРАКОН-программы просто тестировать, так как ДРАКОН-схема дает визуальную подсказку при составлении тестовых сценариев.

Предлагаемый метод автоматного программирования, основанный на языке ДРАКОН, снижает

трудоемкость создания автоматов и сокращает число ошибок. Благодаря языку ДРАКОН работа с автоматами облегчается настолько, что разработчик может применять автоматы везде, где к этому располагает задача. Автоматное программирование становится более доступным и эффективным.

Список литературы

1. **Поликарпова Н. И., Шалыто А. А.** Автоматное программирование. СПб.: Питер, 2011. 176 с.
2. **Wagner F., Schmuki R., Wagner T., Wolstenholme P.** Modeling Software with Finite State Machines. New York: Auerbach Publications, 2006. 392 p.
3. **Шалыто А. А.** SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998. URL: http://is.ifmo.ru/books/switch/1
4. **Machina.js.** URL: http://machina-js.org/
5. **Boost** Meta State Machine. URL: https://www.boost.org/doc/libs/1_67_0/libs/msm/doc/HTML/index.html
6. **UniMod.** URL: http://unimod.sourceforge.net/
7. **Rosmaro**. URL: https://rosmaro.js.org/
8. **YAKINDU** Statechart Tools. URL: https://www.itemis.com/en/yakindu/state-machine/
9. **Пароңджанов В. Д.** Учись писать, читать и понимать алгоритмы. М.: ДМК Пресс, 2012. 520 с.
10. **DRAKON** Editor. URL: http://drakon-editor.sourceforge.net/
11. **DrakonHub.** URL: https://drakonhub.com/
12. **Митькин С. Б.** An interactive state machine demo. URL: https://drakonhub.com/files/lift.html
13. **Miro Samek.** Introduction to Hierarchical State Machines, 2016, URL: https://barrgroup.com/Embedded-Systems/How-To/Introduction-Hierarchical-State-Machines

Automata-Based Programming in DRAKON Language

S. Mitkin, Consultant, stipan.mitkin@gmail.com, ProsessPilotene AS, Asker, 1383, Norway

Corresponding author:

Mitkin Stepan, Consultant, ProsessPilotene AS, Asker, 1383, Norway,
E-mail: stipan.mitkin@gmail.com

Received on September 09, 2018
Accepted on October 23, 2018

A new method for visualization of finite automata (state machines) is proposed. This method combines automata with decision trees using the DRAKON algorithmic language.

DRAKON has visual branching and looping constructs that correspond to the building blocks of structured programming: if/then/else, switch/case, while/for. DRAKON's branching constructs follow strict ergonomic guidelines that help one build consistent and readable decision trees. Another feature of DRAKON is "silhouette" diagrams. A silhouette splits a large algorithm into smaller parts which are called "branches."

In the proposed method, the branches of a silhouette designate the states of an automaton. Each branch has all the information on the corresponding state including its name, the state-specific input symbols, the names of the next states. The branch also contains decision trees that represent the input symbol processing algorithms and state-switching logic. This kind of state machine diagrams is suitable for programming—the author has built a software tool that generates working programs from automata diagrams based on DRAKON.

The presence of decision trees on state machine diagrams produces several positive effects, such as a higher expression power and better readability of automata. Besides, the DRAKON language brings a standardized layout that makes diagrams predictable and easy to comprehend. These benefits together make automata more comfortable to work with and reduce the number of errors. As a result, automata-based programming becomes more practical and productive.

Keywords: *automata-based programming, finite automaton, state machine, DRAKON language, event-driven system, visual programming, decision tree, software system behavior, state diagram*

For citation:

Mitkin S. Automata-Based Programming in DRAKON Language*, Programmnaya Ingeneria,* 2019, vol. 10, no. 1, pp. 3—13

DOI: 10.17587/prin.10.3-13

References

1. **Polikarpova N. I., Shalyto A. A.** *Avtomatnoe programmirovanie* (Automata-based programming), Saint-Petersburg, Piter, 2011, 176 p. (in Russian).
2. **Wagner F., Schmuki R., Wagner T., Wolstenholme P.** Modeling Software with Finite State Machines, New York, Auerbach Publications, 2006. 392 p.
3. **Shalyto A. A.** *SWITCH-tehnologija. Algoritmizacija i programmirovanie zadach logicheskogo upravlenija* (Switch-technology. Algorithmization and programming of logic control problems), Saint-Petersburg, Nauka, 1998, available at: http://is.ifmo.ru/books/switch/1
4. **Machina.js,** available at: http://machina-js.org/
5. **Boost** Meta State Machine, available at: https://www.boost.org/doc/libs/1_67_0/libs/msm/doc/HTML/index.html
6. **UniMod**, available at: http://unimod.sourceforge.net/
7. **Rosmaro**, available at. https://rosmaro.js.org/
8. **YAKINDU** Statechart Tools, available at: https://www.itemis.com/en/yakindu/state-machine/
9. **Parondzhanov V. D.** *Uchis' pisat', chitat' i ponimat' algoritmy* (Learn to write, read and understand algorithms). Moscow: DMK Press, 2012, 520 p. (in Russian).
10. **DRAKON** Editor, available at: http://drakon-editor.sourceforge.net/
11. **DrakonHub,** available at: https://drakonhub.com/
12. **Mitkin S.** An interactive state machine demo, available at: https://drakonhub.com/files/lift.html
13. **Miro Samek,** Introduction to Hierarchical State Machines, 2016, available at: https://barrgroup.com/Embedded-Systems/How-To/Introduction-Hierarchical-State-Machines

УДК 004.056.53 DOI: 10.17587/prin.10.14-19

П. В. Закалкин, канд. техн. наук, сотр., e-mail: ansmed82@mail.ru,
П. В. Мельников, канд. техн. наук, сотр., e-mail: ansmed82@mail.ru,
М. Н. Горюнов, канд. техн. наук, сотр., e-mail: brviktorov@mail.ru,
Р. В. Борзов, сотр., e-mail: brviktorov@mail.ru, Академия Федеральной службы охраны Российской Федерации, г. Орел

Подход к разработке анализатора исходных текстов программ на основе использования LLVM

Описан подход к разработке анализатора исходных текстов программ, в основе которого лежит механизм символьного исполнения кода (инструментарий LLVM/Clang посредством Python binding). Предлагаемое решение обеспечивает корректный разбор файлов исходных текстов на отдельные составляющие с учетом их структуры и взаимосвязей. Оно позволяет избежать ошибок вставки датчиков и предоставляет эффективный механизм реализации алгоритмов поиска уязвимостей.

Ключевые слова: *исходный код, дефект кода, поиск уязвимостей, статический анализ кода, динамический анализ кода, символьное исполнение кода, LLVM, Clang, Python binding, недекларированные возможности, анализатор исходных текстов*

Обоснование актуальности

Обеспечение безопасности информации, которая хранится и обрабатывается в различных информационных системах, подразумевает использование совокупности программно-технических и организационных средств и мер защиты. Важная роль при этом отводится средствам поиска дефектов безопасности программного кода (далее по тексту — дефектов кода), не выявленных на этапе его тестирования [1].

Наличие дефектов кода и уязвимостей программного кода может негативно сказываться на ресурсоемкости и скорости выполнения программы, а также создает угрозу безопасности. Использование ряда дефектов и уязвимостей, наличие которых не было своевременно обнаружено, потенциально способно предоставить злоумышленникам возможности по нарушению конфиденциальности, целостности и доступности информации [2]. Указанные обстоятельства обусловливают необходимость разработки инструментария для поиска и нейтрализации дефектов программного кода.

Наиболее эффективными средствами обнаружения дефектов и уязвимостей являются инструментальные средства, реализующие функции статического и динамического анализа [3, 4], ядром которых является лексический анализатор исходного кода. При этом, как правило, лексические анализаторы разрабатываются в виде решений, реализованных в отрыве от инфраструктуры компилятора или командного интерпретатора. В условиях проведения тематических испытаний, когда все этапы получения исполняемого кода из исходного являются открытыми, такой подход является неэффективным ввиду значительного числа ошибок при разборе кода на совокупность лексем. Кроме того, несмотря на многообразие инструментальных средств анализа исходного кода, отдельные утилиты не позволяют удовлетворить в полной мере требованиям к составу и содержанию проверок тематических исследований [5]. Это обстоятельство вынуждает экспертов испытательных лабораторий использовать набор различных инструментальных механизмов анализа и заниматься разработкой собственного инструментария для автоматизации проверок на отсутствие недекларированных возможностей [6, 7].

Подходы к разработке анализатора исходных текстов

Простейший подход к выявлению дефектов и уязвимостей кода основан на использовании сигнатурного анализа, который не требует предварительного преобразования исходных текстов программного обеспечения. Сигнатурный анализ позволяет найти отдельные уязвимости (потенциально опасные функции) в исходных текстах путем сопоставления их с базой данных шаблонов. Основным недостатком сигнатурного анализа является значительное число ложных срабатываний, что чрезмерно загружает отчеты, создавая определенные неудобства при их анализе. Частично вопрос загруженности отчетов решается использованием регулярных выражений, однако их использование не позволяет формировать условия проверок с учетом семантики исходного кода. Од-

нако такие проверки необходимы, например, при выявлении дефекта кода CWE*—120 "Классическое переполнение буфера".

Более совершенный подход к поиску дефектов основан на использовании результатов работы лексического и синтаксического анализаторов исходных текстов программного обеспечения. Лексический анализатор или "токенизатор" — это программа, выполняющая лексический анализ исходных текстов в две стадии: сканирование и оценка [8]. На стадии сканирования лексический анализатор реализуется в виде конечного автомата, определяемого регулярными выражениями. В нем кодируется информация о возможных последовательностях символов, которые могут встречаться в токенах (лексемах). Синтаксический анализ представляет собой процесс сопоставления линейной последовательности лексем (слов, токенов) естественного или формального языка с его формальной грамматикой. В ходе синтаксического анализа исходный текст преобразуется в структуру данных, которая, как правило, представляет собой дерево. Оно отражает синтаксическую структуру входной последовательности и хорошо подходит для дальнейшей обработки.

Данные, полученные по итогам лексического и синтаксического анализа, могут быть использованы для дальнейшего анализа исходного кода с помощью механизма символьного исполнения. Суть этого механизма заключается в том, что автоматический анализатор моделирует поведение программы шаг за шагом вдоль всех возможных путей в графе потока управления (дерева). При этом для неизвестных величин вводятся символьные обозначения ("символы") и возможные комбинации конкретных значений символов объявляются эквивалентными, если они ведут программу по одному и тому же пути исполнения. Всякое ветвление в программе, таким образом, разбивает множество "состояний программы" (возможных комбинаций значений символов) на классы эквивалентных состояний. При последующем обходе эти классы анализируются независимо. Символьное исполнение применяется при статическом анализе текста программы в упрощенном промежуточном представлении, таком как Java-байткод или LLVM. Результаты подобного анализа являются наиболее полезными, поскольку позволяют получить наглядное и полноценное описание найденных дефектов в терминах исходного языка и исходного кода [9]. Более подробная информация по механизмам символьного исполнения представлена в работе [10].

Разработка алгоритмов поиска уязвимостей на основе использования механизма символьного выполнения кода является более ресурсоемкой. Однако такой подход видится более перспективным ввиду больших возможностей по реализации проверок параметров исследуемых функциональных объектов.

Предлагаемое решение

В качестве основы для анализатора исходных текстов (C/C++) авторами настоящей работы предлагается использовать универсальную систему анализа, трансформации и оптимизации программ LLVM с фронтендом Clang [11], предоставляющие набор инструментов, позволяющих полностью воспроизвести функциональные возможности компилятора и провести анализ исходных текстов на уровне абстрактного синтаксического дерева (АСД).

Важной особенностью LLVM является наличие механизмов работы с файлом исходного текста на АСД с использованием API-функций посредством Python binding. Использование Python binding (см. таблицу) позволяет получить доступ к внутренним элементам абстрактного синтаксического дерева (внутреннее представление исходного кода в LLVM); провести анализ его структуры и взаимосвязей функциональных и информационных объектов; осуществить поиск дефектов кода, минимизировав при этом ошибки первого и второго рода за счет использования возможностей механизма символьного выполнения.

Структурная схема анализатора исходных текстов программ на основе LLVM [12, 13] приведена на рис. 1.

Сценарий работы анализатора включает ряд этапов.

1. Компиляция исследуемого ПО с параллельным анализом аргументов командной строки (передаваемых компилятору опций) и формированием безизбыточного (на уровне файлов) перечня исходных текстов.

2. Выполнение анализа файлов исходных текстов встроенным анализатором Clang Static Analyzer на предмет наличия ошибок кода.

3. Построение метрик в соответствии с требованиями руководящего документа [5].

4. Проверка наличия потенциально опасных конструкций и CWE-уязвимостей.

5. Формирование отчетов по результатам проверок (использование SQLite-базы данных).

Этапы в сценарии работы анализатора разработаны на основании требований регулятора в данной области — Федеральной службы по техническому и экспортному контролю (ФСТЭК России).

При работе анализатора используется база данных "Объекты исходных текстов программного обеспечения", которая инкапсулирует информацию, извлекаемую из файлов исходных текстов программного обеспечения [14]. База данных предназначена для автоматизации технологических процедур, выполняемых экспертом испытательной лаборатории при определении соответствия исследуемого программного обеспечения требованиям соответствующего

* CWE (Common Weakness Enumeration) — общий перечень дефектов безопасности ПО. Представляет собой реестр (словарь) общих дефектов безопасности, которые могут проявиться в архитектуре, проектировании, коде или реализации ПО, и могут быть использованы злоумышленниками для получения несанкционированного доступа к системе.

Основные функции Clang Python binding, используемые при разработке анализатора исходных текстов

Функция	Описание
import clang.cindex from clang.cindex import	Подключение библиотеки libclang
tu = idx.parse(f_name, args = incl_list, options = 8)	Выполнение парсинга файла исходного текста (f_name) на узлы абстрактного синтаксического дерева
for c in tu.cursor.walk_preorder(): ...	Осуществление прохода по узлам абстрактного синтаксического дерева
if c.kind == CursorKind.FUNCTION_DECL or c.kind == CursorKind.CXX_METHOD: ...	Поиск и обработка узлов, содержащих определяемые функциональные объекты
if c.kind == CursorKind.CALL_EXPR: ...	Поиск и обработка узлов, содержащих вызываемые функциональные объекты
if c.kind == CursorKind.VAR_DECL or c.kind == CursorKind.OBJC_IVAR_DECL: ...	Поиск и обработка узлов, содержащих информационные объекты
if childr.kind == CursorKind.IF_STMT: ... if childr.kind == CursorKind.SWITCH_STMT: ... if childr.kind == CursorKind.CASE_STMT: ...	Поиск и обработка узлов, содержащих ветвления функциональных объектов

уровня контроля отсутствия недекларируемых возможностей.

Данные в базе сгруппированы в семь таблиц "Файл анализируемого проекта", "Определяемый функциональный объект", "Вызываемый функциональный объект", "Ветвь функционального объекта", "Информационный объект", "Ошибки ПО (CWE)" и "Потенциально опасные функции". Структура базы данных (рис. 2) обеспечивает компактное хранение и быстрый доступ к информации о функциональных и информационных объектах в составе исходных текстов анализируемого программного обеспечения, а также о связях между ними.

Данные в таблице позволяют сформировать совокупность отчетов для поддержки принятия решения о результатах тематических исследований на их соответствие требованиям определенного уровня контроля отсутствия недекларированных возможностей (рис. 3).

Работа модуля сигнатурного анализа (рис. 4) на уровне структур внутреннего представления исходного кода в LLVM позволяет снизить процент ложных срабатываний за счет корректной обработки параметров функциональных объектов. Таким образом, для уязвимости CWE-120 (переполнение буфера) процент ложных срабатываний удалось понизить до уровня около 70 % (модуль сигнатурного анализа находится в стадии активной разработки и значение продолжает снижаться). При использовании классического сигнатурного подхода этот показатель близок к 95 %.

Уменьшение числа срабатываний "разгружает" отчеты сигнатурного анализа и позволяет экспертам провести их более тщательный анализ.

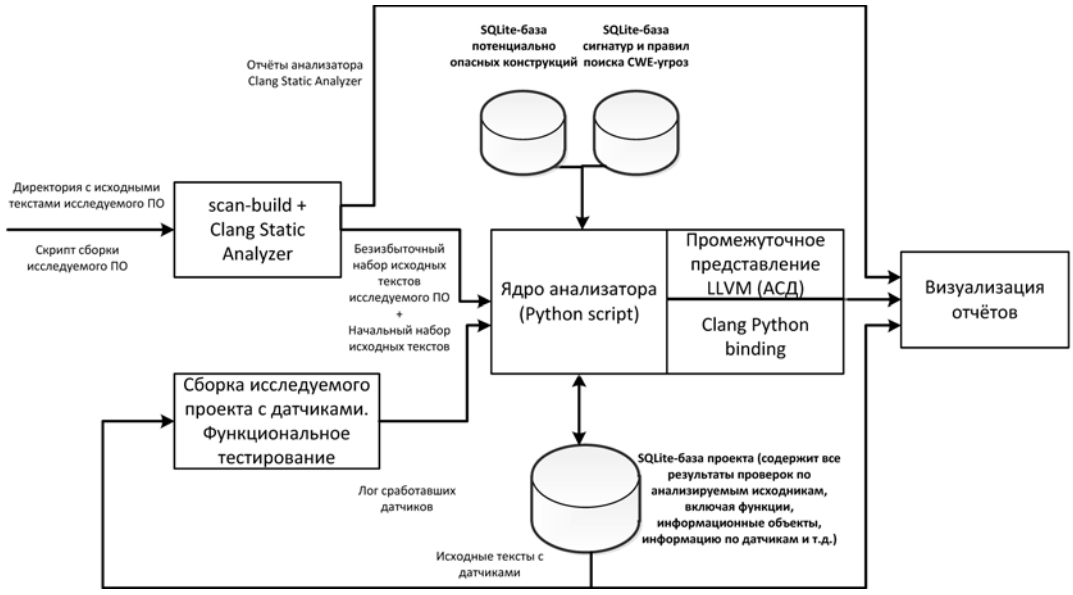


Рис. 1. Структурная схема анализатора исходных текстов программ на основе LLVM

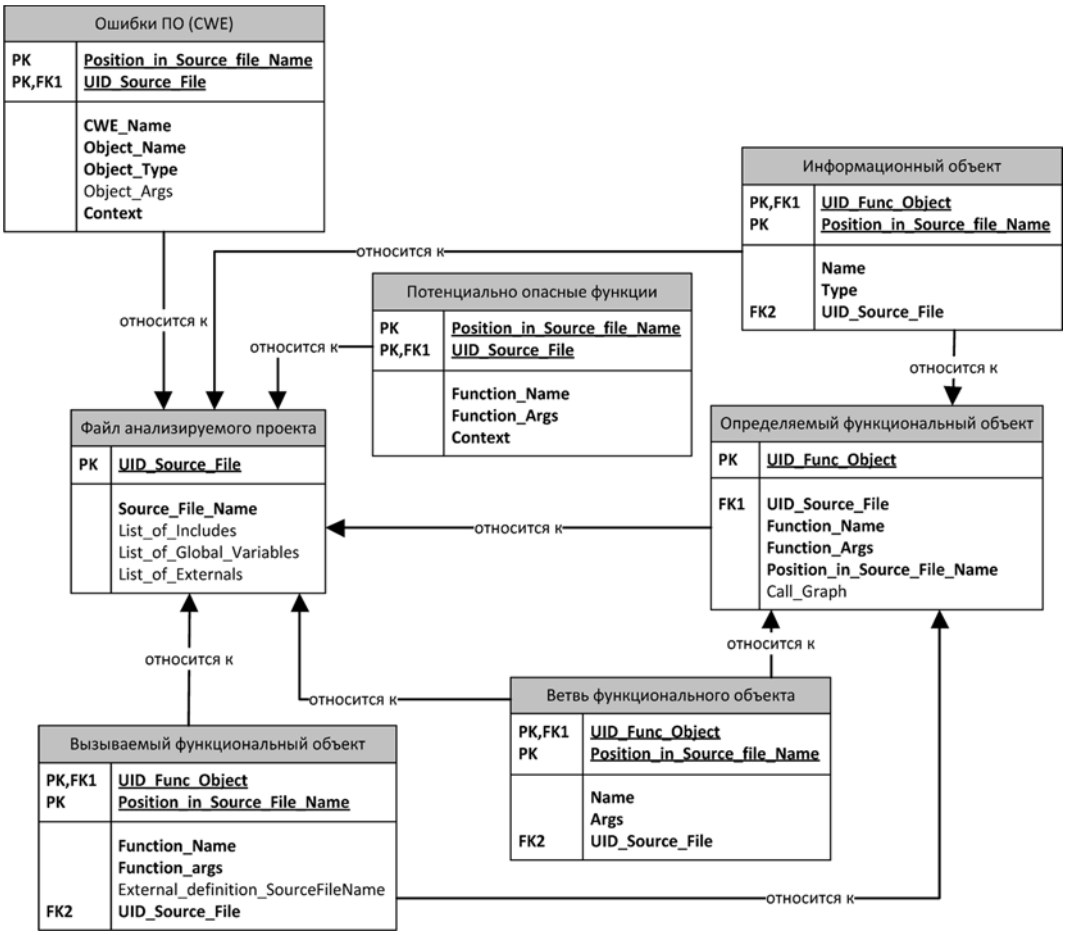


Рис. 2. Структура таблиц базы данных "Объекты исходных текстов программного обеспечения"

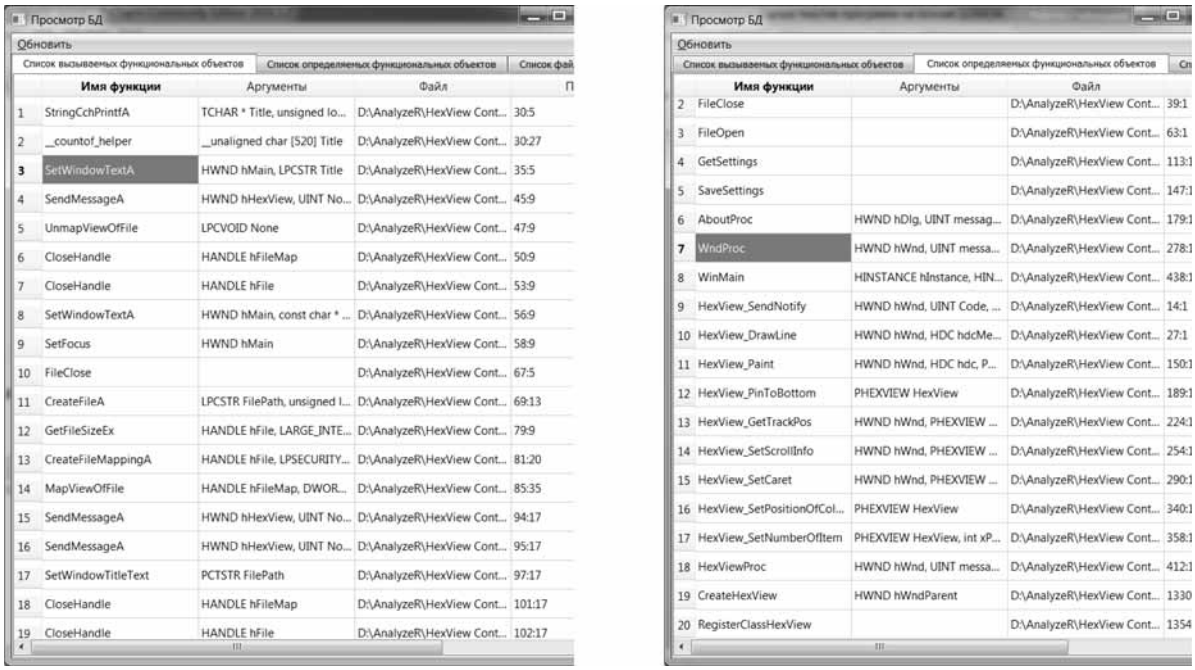
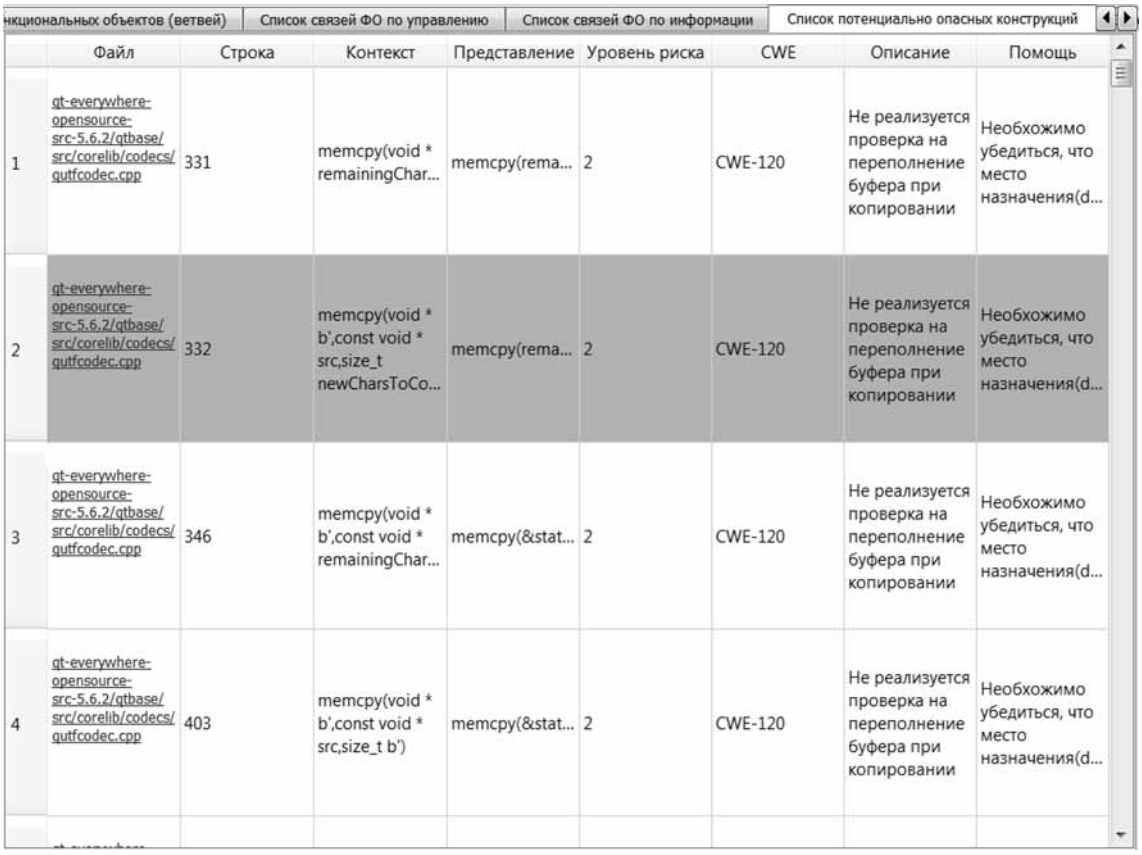


Рис. 3. Пример отчетов по результатам статического анализа с использованием подхода на основе LLVM



нкциональных объектов (ветвей) | Список связей ФО по управлению | Список связей ФО по информации | Список потенциально опасных конструкций

	Файл	Строка	Контекст	Представление	Уровень риска	CWE	Описание	Помощь
1	qt-everywhere-opensource-src-5.6.2/qtbase/src/corelib/codecs/qutfcodec.cpp	331	memcpy(void * remainingChar...	memcpy(rema...	2	CWE-120	Не реализуется проверка на переполнение буфера при копировании	Необхожимо убедиться, что место назначения(d...
2	qt-everywhere-opensource-src-5.6.2/qtbase/src/corelib/codecs/qutfcodec.cpp	332	memcpy(void * b',const void * src,size_t newCharsToCo...	memcpy(rema...	2	CWE-120	Не реализуется проверка на переполнение буфера при копировании	Необхожимо убедиться, что место назначения(d...
3	qt-everywhere-opensource-src-5.6.2/qtbase/src/corelib/codecs/qutfcodec.cpp	346	memcpy(void * b',const void * remainingChar...	memcpy(&stat...	2	CWE-120	Не реализуется проверка на переполнение буфера при копировании	Необхожимо убедиться, что место назначения(d...
4	qt-everywhere-opensource-src-5.6.2/qtbase/src/corelib/codecs/qutfcodec.cpp	403	memcpy(void * b',const void * src,size_t b')	memcpy(&stat...	2	CWE-120	Не реализуется проверка на переполнение буфера при копировании	Необхожимо убедиться, что место назначения(d...

Рис. 4. Пример отчета сигнатурного анализа

Заключение

Представленный в настоящей работе подход к разработке анализатора исходных текстов программ обеспечивает корректный разбор файлов исходных текстов на отдельные составляющие (с учетом их структуры и взаимосвязей). Он позволяет избежать ошибок вставки датчиков и предоставляет эффективный механизм реализации алгоритмов поиска дефектов и уязвимостей.

По сравнению с существующими решениями предложенный подход позволяет:

— повысить оперативность тематических исследований посредством автоматизации ряда рутинных процедур;

— повысить достоверность тематических исследований с использованием совокупности отчетов поддержки принятия решения о соответствии исходных текстов программного обеспечения требованиям определенного уровня контроля отсутствия недекларированных возможностей;

— снизить число ложных срабатываний сигнатурного анализа за счет обработки параметров функциональных объектов на уровне внутреннего представления исходного кода в LLVM.

Список литературы

1. **Контроль** уязвимостей в программных приложениях. URL: https://habrahabr.ru/company/jetinfosystems/blog/241353/.

2. **Марков А. С., Фадин А. А., Цирлов В. Л.** Систематика дефектов и уязвимостей программного обеспечения //Безопасные информационные технологии. Сборник трудов II НТК. М.: МГТУ им. Н. Э. Баумана, 2011. С. 83—87.

3. **Аветисян А., Бородин А., Несов В.** Использование статического анализа для поиска уязвимостей и критических ошибок в исходном коде программ // Труды ИСП РАН. 2011. Том. 21. С. 23—28.

4. **Аветисян А. И., Белеванцев А. А., Чукляев И. И.** Технологии статического и динамического анализа уязвимостей программного обеспечения // Вопросы кибербезопасности. 2014. № 3 (4). С. 20—28.

5. **Руководящий** документ. Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровням контроля отсутствия недекларированных возможностей. М.: Гостехкомиссия России, 1999.

6. **Анисимов Д. В., Мельников П. В.** Проведение сертификационных исследований программного обеспечения с использованием технологии LLVM // Информационные системы и технологии. 2016. № 2. С. 99—104.

7. **Мельников П. В., Горюнов М. Н., Анисимов Д. В.** Подход к проведению динамического анализа исходных текстов программ // Вопросы кибербезопасности. 2016. № 3 (16). С. 33—39.

8. **Ахо А. В., Лам М. С., Сети Р., Ульман Д. Д.** Компиляторы: принципы, технологии и инструментарий = Compilers: Principles, Techniques, and Tools. 2-е изд. М.: Вильямс, 2008. 1184 с.

9. **Дергачев А. В., Сидорин А. В.** Основанный на резюме метод реализации произвольных контекстно-чувствительных проверок при анализе исходного кода посредством символьного выполнения // Труды ИСП РАН. 2016. Том 28, Вып. 1. С. 41—62.

10. **Ермаков М. К.** Методы повышения эффективности итеративного динамического анализа программ: дисс. ... канд. техн. наук. Институт системного программирования им. В. П. Иванникова РАН, Москва, 2016.

11. **Clang:** a C language family frontend for LLVM. URL: http://clang.llvm.org.

12. **Закалкин П. В., Мельников П. В.** Система анализа программного обеспечения на предмет отсутствия недекларированных возможностей // Программная инженерия. 2018. Том 9, № 2. С. 69—75.

13. **Патент** 2622622 Российская Федерация, МПК G06F 21/00, G06F 21/50, G06F 11/30, G06F 12/16. Система анализа программного обеспечения на отсутствие недекларированных возможностей / Горюнов М. Н., Мельников П. В., Закалкин П. В., Воробьев С. А., Анисимов Д. В., Пе-

тров К. Е.; заявитель и патентообладатель Федеральное государственное казенное военное образовательное учреждение высшего образования "Академия Федеральной службы охраны Российской Федерации". 2016110533; заявл. 22.03.2016; опубл. 16.06.2017. 14 с.

14. **Государственная** регистрация базы данных 2017620777 Российская Федерация, Объекты исходных текстов программного обеспечения / Мельников П. В., Закалкин П. В., Воробьев С. А.; заявитель и патентообладатель Федеральное государственное казенное военное образовательное учреждение высшего образования "Академия Федеральной службы охраны Российской Федерации". — 2017620471; заявл. 22.05.201; опубл. 19.07.2017.

Approach to Development of the Analyzer of Source Texts of Programs on the Basis of LLVM

P. V. Zakalkin, ansmed82@mail.ru, **P. V. Mel'nikov,** ansmed82@mail.ru, **M. N. Gorunov,** brviktorov@mail.ru, **R. V. Borzov,** brviktorov@mail.ru, Academy of the Federal Guard Service of the Russian Federation, Orel, 302034, Russian Federation

Corresponding author:

Zakalkin Pavel V., Employer, Academy of the Federal Guard Service of the Russian Federation, Orel, 302034, Russian Federation,
E-mail: ansmed82@mail.ru

Received on October 24, 2018
Accepted on November 06, 2018

Despite the variety of decisions in the field of source code analysis, separate means do not allow to fully meet requirements to structure and content of checks of case studies. This circumstance forces experts of testing laboratories to use a set of various tools for the analysis and to be engaged in development of own automation equipment of technological process. The important part at the same time is assigned to the means of search of defects (shortcomings) and vulnerabilities of a program code bearing a threat to security of the processed information assets.

In this work an approach to development of the analyzer of source texts of programs using the LLVM/Clang tools by means of Python binding is described. The proposed solution provides correct analysis of source text files on separate components (taking their structure and interrelations into account), allows one to avoid errors of sensors inserting and provides an effective implementation of algorithms for vulnerabilities search from the list of actual CWEs.

Keywords: *source texts of programs, defects of a code, static analysis, dynamic analysis, symbolical execution of a code, LLVM, Clang, Python binding, search of vulnerabilities, undeclared opportunities, the analyzer of source texts*

For citation:

Zakalkin P. V., Mel'nikov P. V., Gorunov M. N., Borzov R. V. Approach to Development of the Analyzer of Source Texts of Programs on the Basis of LLVM, *Programmnaya Ingeneria,* 2019, vol. 10, no. 1, pp. 14—19.

DOI: 10.17587/prin.10.14-19

References

1. **Control** of vulnerabilities in software application (Kontrol' ujazvimostej v programmnyh prilozhenijah), available at: https://habrahabr.ru/company/jetinfosystems/blog/241353/ (in Russian)

2. **Markov A. S., Fadin A. A., Cirlov V. L.** Systematization of defects and vulnerabilities software (Sistematika defektov i ujazvimostej programmnogo obespechenija), *Bezopasnye informacionnye tehnologii. Sbornik trudov II NTK*, Moscow, MGTU im. N. Je. Baumana, 2011, pp. 83—87 (in Russian).

3. **Avetisjan A., Belevancev A., Borodin A., Nesov V.** Use of the static analysis for search of vulnerabilities and critical mistakes in a source code programs (Ispol'zovanie staticheskogo analiza dlja poiska ujazvimostej i kriticheskih oshibok v ishodnom kode program), *Trudy ISP RAN,* 2011, vol. 21, pp. 23—28 (in Russian).

4. **Avetisjan A., Belevancev A., Chukljaev I.** Technologies of the static and dynamic analysis of vulnerabilities Software (Tehnologii staticheskogo i dinamicheskogo analiza ujazvimostej programmnogo obespechenija), *Voprosy kiberbezopasnosti,* 2014, no. 3 (4), pp. 20—28 (in Russian).

5. **The leading document** "Protection against unauthorized access to information. P.1. Software of means of information protection. Classification by the level of control of lack of not declared opportunities" (Zashhita ot nesankcionirovannogo dostupa k informacii. Ch. 1. Programmnoe obespechenie sredstv zashhity informacii. Klassifikacija po urovnju kontrolja otsutstvija nedeklarirovannyh vozmozhnostej). It is approved as the decision of the chairman of the State technical commission at the President of the Russian Federation 04.06.1999 (in Russian).

6. **Anisimov D. V., Mel'nikov P. V.** Carrying out certified researches of the software with use of the LLVM technology (Provedenie sertifikacionnyh issledovanij programmnogo obespechenija s ispol'zovaniem tehnologii LLVM), *Informacionnye sistemy i tehnologii*, 2016, no. 2, pp. 99—104 (in Russian).

7. **Mel'nikov P. V., Anisimov D. V., Gorjunov M. N.** Approach to carrying out the dynamic analysis of source texts programs (Podhod k provedeniju dinamicheskogo analiza ishodnyh tekstov programm), *Voprosy kiberbezopasnosti,* 2016, no. 3 (16), pp. 33—39 (in Russian).

8. **Akho A. V., Lahm M. S., Seti R., Ullman J. D.** *Compilers: principles, technologies and tools*, 2nd ed., Addison Wesley, 2007, 1038 p.

9. **Dergachyov A. V., Sidorin A. V.** Based on the summary a method of realization of any context-sensitive checks in the analysis of a source code by means of symbolical performance (Osnovannyj na rezjume metod realizacii proizvol'nyh kontekstno-chuvstvitel'nyh proverok pri analize ishodnogo koda posredstvom simvol'nogo vypolnenija*), Trudy ISP RAN*, 2016, vol. 28, issue 1, pp 41—62 (in Russian).

10. **Ermakov M. K.** Metody povysheniya ehffektivnosti iterativnogo dinamicheskogo analiza program (Methods to improve the efficiency of iterative dynamic program analysis): diss. kand. tekh. nauk. Institut sistemnogo programmirovaniya im. V. P. Ivannikova RAN, Moscow, 2016 (in Russian).

11. **Clang:** a C language family frontend for LLVM, available at: http://clang.llvm.org.

12. **Zakalkin P. V., Mel'nikov P. V.** System of the analysis of the software regarding lack of not declared opportunities **(**Sistema analiza programmnogo obespecheniya na predmet otsutstviya nedeklarirovannyh vozmozhnostej) *Programmnaya inzheneria,* 2018, vol. 9, no. 2, pp. 69—75 (in Russian).

13. **Gorjunov M. N., Mel'nikov P. V., Zakalkin P. V., Vorob'ev S. A., Anisimov D. V., Petrov K. E.** Pat. 2622622 Russian Federation, MPK G06F 21/00, G06F 21/50, G06F 11/30, G06F 12/16. System of the analysis of the software on absence not declared opportunities (Sistema analiza programmnogo obespechenija na otsutstvie nedeklarirovannyh vozmozhnostej). Applicant and patent holder Federal public state military educational institution of the higher education "Academy of Federal Guard Service of the Russian Federation". — 2016110533; 22.03.2016 is declared; 16.06.2017 is published. 14 p. (in Russian).

14. **Zakalkin P. V., Mel'nikov P. V., Vorob'ev S. A.** State registration of the database 2017620777 Russian Federation, Objects of source texts of software (Ob'ekty ishodnyh tekstov programmnogo obespechenija). Applicant and patent holder Federal public state military educational institution of the higher education "Academy of Federal Guard Service of the Russian Federation". — 2017620471; 22.05.201 is declared; 19.07.2017 is published (in Russian).

УДК 004.056 DOI: 10.17587/prin.10.20-29

Д. К. Левоневский, науч. сотр., e-mail: DLewonewski.8781@gmail.com,
И. В. Ватаманюк, мл. науч. сотр., e-mail: vatamaniuk@iias.spb.su,
Д. А. Малов, аспирант, e-mail: malovdmitrij@gmail.com, Санкт-Петербургский институт информатики и автоматизации Российской академии наук (СПИИРАН)

Обеспечение доступности сервисов корпоративного интеллектуального пространства посредством управления потоком входных данных

Рассмотрены вопросы обеспечения доступности сервисов корпоративного интеллектуального пространства. В качестве угроз доступности для такой системы рассмотрены намеренные и ненамеренные воздействия на интеллектуальное пространство. К их числу относятся конфликтующие запросы, исчерпание ресурсов, сетевые атаки. Для противодействия этим угрозам исследованы известные методы организации запросов, предложена методика оценки доступности сервисов, построена модель системы массового обслуживания и проведены эксперименты с различными стратегиями управления запросами и методами их фильтрации. Эксперименты показывают, что оценка эффективности такого управления имеет максимумы, зависящие от условий функционирования прикладной системы и параметров системы управления запросами. Таким образом, задача определения этих параметров может быть представлена как оптимизационная задача в многомерном пространстве. Для учета изменяющихся условий функционирования прикладной системы может использоваться адаптивный подход. Проведенные исследования указывают, что улучшить качество обслуживания пользователей в интеллектуальном корпоративном пространстве можно путем повышения доступности его сервисов.

Ключевые слова: *интеллектуальное пространство, многомодальные интерфейсы, корпоративные информационные системы, доступность, система массового обслуживания, компьютерная безопасность*

Введение

Интеллектуальные пространства обеспечивают наиболее естественный способ взаимодействия пользователей с информационной системой, которая формирует среду организации [1]. Эффективность реализации такого интеллектуального пространства поддерживается механизмами повсеместных вычислений, повсеместных коммуникаций и многомодальных интерфейсов [2]. Простота и скорость доступа к информации, а также удобство предоставляемых пользовательских интерфейсов являются отличительной чертой систем подобного класса. Вместе с тем при разработке и развертывании таких многомодальных повсеместных систем человеко-машинного взаимодействия возникает целый ряд сложностей. К ним относятся такие аспекты, как обеспечение безопасности, установление надежных связей между узлами, управление разнородными устройствами, приоритетность выполнения задач, оценка качества их работы и т. д. Особо следует отметить обработку множественных запросов, поступающих одновременно от разных пользователей, и обеспечение надлежащего качества восприятия при работе со многими пользователями, которое выражается в обеспечении доступности сервисов интеллектуального пространства. Рассмотрим подробнее решения, которые предлагаются в контексте разрешения отмеченных сложностей.

1. Состояние проблемы

Рассмотрим типовые примеры угроз доступности информации в сервисах интеллектуального пространства.

1. Целенаправленные деструктивные воздействия на сервисы с помощью недостоверной или некорректной информации. В этом случае источником угрозы является пользователь, который может быть как легитимным, так и нелегитимным. Типовым примером реализации такой угрозы является "отказ в обслуживании" (DoS).

2. Нецеленаправленные воздействия на сервис в условиях поступления от множества пользователей заявок, которые не могут быть выполнены сервисами в сроки, при которых актуальность заявок сохраняется. В этом случае источником угрозы является легитимный пользователь. Пример реализации угрозы — исчерпание пропускной способности канала передачи данных [3].

3. Ошибочное восприятие сервисами поступающих заявок. Эта угроза вероятна при использовании многомодальных средств человеко-машинного взаимодействия — при взаимодействии с сервисами с помощью речи, жестов, при распознавании образов на видео. Угроза может также осуществляться вследствие ошибок в программном обеспечении. Источник угрозы — программное обеспечение. Обработка ошибочно воспринятых данных негативно сказывается на доступности сервисов для легитимных пользователей.

Необходимо отметить, что взаимодействие пользователя с корпоративной информационной системой — это комплексный процесс, объективное и субъективное качество которого обусловлено множеством различных факторов. Основными его показателями являются так называемые качество восприятия (*Quality of Experience*, QoE) и качество обслуживания (*Quality of Service*, QoS). Первое понятие, как правило, рассматривается в более широком смысле и включает в себя второе в качестве одного из показателей (рис. 1).

В литературе [4—6] предложены следующие объективные и субъективные определения QoE. Объективное QoE определяет качество восприятия, предоставляемое пользователю информационной системы с точки зрения измеримых показателей производительности услуг, сети и приложений. Субъективное QoE определяет качество, воспринимаемое пользователем с позиции получаемых им эмоций, биллинга услуг и соответствия опыта его взаимодействия с системой.

Доступность сервисов в различных условиях может оцениваться с помощью известных методов оценки качества обслуживания. Например, среднее время выполнения заявки может рассматриваться как параметр QoS. Это время определяется как математическое ожидание интервала времени между отправкой запроса о выделении ресурса и его фактическим выделением. Эту величину можно оценить, как среднее арифметическое значение измерений, составляющих представительную выборку.

Основные аспекты обеспечения надлежащего качества многопользовательского обслуживания многомодальных распределенных систем рассмотрены в работе [7]. Авторы предлагают решение для обработки нескольких пользовательских запросов для многоэкранной системы цифровых вывесок (*digital signage*, DS), основанной на технологии программно-конфигурируемого Интернета вещей (*software-defined IoT*). Предлагаемый метод использует процесс динамического выделения каналов, который реализуется на основе программно-конфигурируемой сетевой архитектуры (*software-defined networking*, SDN). В такой системе функциональные возможности физического уровня отделены от уровня управления. Таким образом, SDN-контроллер системы DS оценивает состояние сети и число пользовательских запросов, он автоматически обновляет правила обработки запросов и перераспределяет системные ресурсы. Кроме перечисленных выше возможностей авторы предлагают оригинальную реализацию так называемой "невидимой связи", которая предоставляет пользователю инструментарий для эффективного доступа к сетевым службам. Такая интерактивность между стационарными терминалами и мобильными устройствами пользователя существенно расширяет возможности пользователей.

Как правило, показатель качества восприятия наиболее существенен для передачи потокового видео. Его оценка и прогнозирование на основе расширенной SSIM-метрики обсуждены в работе [8]. Подходы, основанные на методах машинного обучения, SDN и подобных им, рассмотрены в работах [9—11].

Качество обслуживания визуальных сервисов, предоставляемых системой цифровых вывесок, обсуждено в работе [12]. Метрика, которая рассматривается в этой публикации, определяется на основе уровня воспри-

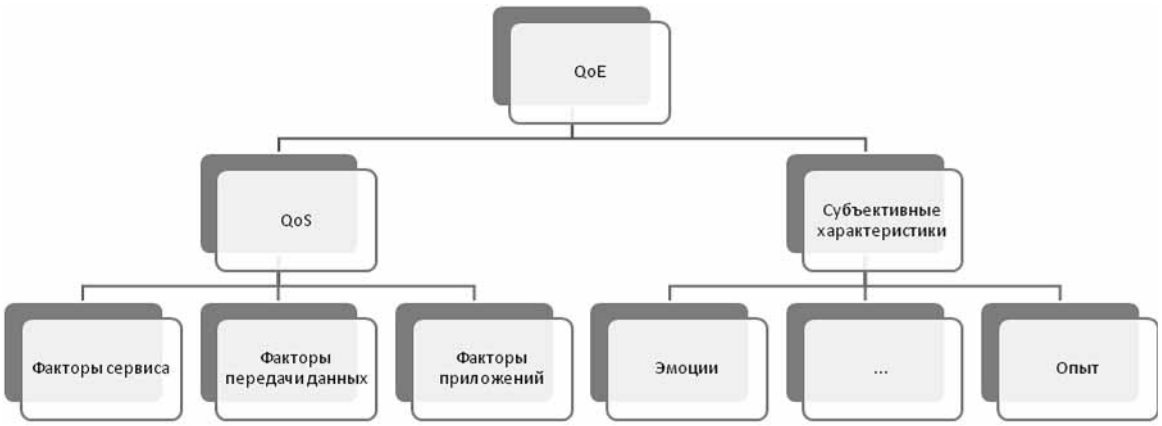


Рис. 1. Структура QoE

нимаемой чувствительности двумерного изображения и способности выполнять слияние двумерных изображений в стереоизображение. Такое изображение будет представляться трехмерным для пользователей, которые находятся на разных расстояниях от экрана. Использование данного подхода позволяет автоматически контролировать визуальную информацию для нескольких пользователей без организации дополнительных механизмов взаимодействия.

Системы цифровых вывесок находят свое применение в розничных услугах, размещении рекламы сторонних организаций, в сфере развлечений, в образовании и на транспорте. В работе [13] представлена архитектура системы предоставления телеуслуг с поддержкой мобильных клиентов и с обеспечением надлежащего качества обслуживания в транспортных сетях. Она позволяет управлять регистрацией пользователей, рекламными объявлениями, сетевой информацией и качеством обслуживания благодаря адаптивному выбору параметров QoS и специализированному механизму принятия решений для обеспечения надлежащего качества изображения и улучшения восприятия пользователями мультимедийных материалов.

Многоэкранное окружение (*multi-screen environment*) позволяет применять технологии цифровых вывесок в качестве эффективного средства рекламы и информации [14], что делает его неотъемлемой частью киберфизического интеллектуального пространства, незаменимой в плане предоставления информационных сервисов. Для обеспечения синхронизации нескольких экранов, а также для обеспечения корректности вывода мультимедийного контента могут быть использованы методы программно-реализованной синхронизации многоэкранного окружения. Такой подход обеспечивает лучшую производительность и расширяет возможности системы цифровых вывесок, позволяя использовать экраны нестандартных размеров и форм независимо от их расположения в интеллектуальном пространстве.

В работе [15] описан многоадресный многоэкранный сервис (*N-Screen Multicast Service*), обеспечивающий надлежащее качество обслуживания в зависимости от доступных параметров качества восприятия элементов системы. В такой многоэкранной системе пользователи динамически объединяются в многоадресные группы в зависимости от емкости их устройств и интернет-трафика. Воспринимаемое качество мультимедийного контента, предоставляемого каждой группе, динамически корректируется для лучшего удовлетворения требований качества восприятия в соответствии с доступными ресурсами пользовательского устройства. Подобная схема помогает оптимизировать использование сетевых ресурсов в зависимости от требований, предъявляемых к качеству контента.

В контексте настоящей статьи авторы используют допущение о том, что программное обеспечение многомодальной системы может проверять валидность запросов и отбрасывать некорректные. Методы валидации известны и основаны на формальных грамматиках, регулярных выражениях и на использовании ряда других подходов. Сложность представляют ситуации, когда система получает формально валидные запросы, которые могут оказать негативное влияние на ее доступность и которые сложно отличить от безопасных запросов с помощью предлагаемых методов. Тем не менее существуют способы противодействия такого сорта неопределенностям, основанные на математической статистике (пороговый анализ [16], анализ функций распределения [17]), на вычислении энтропии [18] и т. д. В статистических подходах анализируются выборочные статистические моменты и форма функции распределения для измеряемых величин. Например, низкие значения моментов и энтропии, а также экспоненциальная функция распределения могут быть характерны для трафика, генерируемого автоматически.

2. Разработка модели

Типичной областью приложения распределенных многомодальных систем являются корпоративные интеллектуальные пространства, обеспечивающие человеко-машинное взаимодействие между пользователями и компонентами этого пространства. В качестве примера можно рассмотреть процесс обращения пользователей к информационному сервису МИНОС [19]. Информационно-навигационные сервисы МИНОС взаимодействуют с гостями с помощью комплекса веб-камер и экранов, помогающих пользователям ориентироваться в корпоративном пространстве СПИИРАН. Веб-камеры используются в процессах регистрации, аутентификации и распознавания пользователей и позволяют автоматизировать работу пропускного пункта. Экраны, в том числе сенсорные, позволяют пользователям легко получать требуемую информацию.

Пользователи могут отдавать команды сервису, используя различные модальности, например, речь и жесты, они также могут управлять сервисом с мобильного устройства или с персональных компьютеров (для администраторов). Каждая команда требует обслуживания (выполнения заявки), которое предполагает загрузку определенных аппаратных и программных ресурсов (дисплей, звук, пользовательская сессия) в течение некоторого времени. В случае, когда с сервисом работает один пользователь единовременно, при поступлении новой команды можно предполагать завершение выполнения предыдущей команды, и конфликтов не возникает. Однако при наличии нескольких пользователей в конкретный момент времени сервис может столкнуться с отсутствием возможности выделить требуемый ресурс всем заявкам. В этом случае необходимо найти оптимальную стратегию управления заявками, которая обеспечивала бы максимальное качество обслуживания. Оптимальность той или иной стратегии при этом зависит от ряда параметров, к числу которых относятся интенсивность поступления заявок, их ресурсоемкость, число пользователей.

Каждая заявка характеризуется параметрами, определяющими пользователя, сессию, модальность и запрашиваемое действие.

Рассмотрим классификацию заявок в МИНОС, которые различаются:

1) по распределенности: локальные (требуют обработки только на одном канале обслуживания); распределенные (требуют обработки на нескольких каналах); глобальные (требуют обработки на всех каналах);

2) по приоритету — критические (с немедленным вытеснением всех других заявок); регулярные (имеющие относительный приоритет в дисциплине обслуживания); фоновые (выполняются только при отсутствии других заявок, немедленно вытесняются любыми нефоновыми заявками);

3) по процедуре выполнения — непрерывные и вытесняемые;

4) по длительности — с фиксированным и изменяемым временем их выполнения;

5) по связи с другими заявками — одиночные, сессионные.

Более детально параметры заявки можно описать в формате JSON:

```
request: {
    user: {
          id,
          priority
},
    session: {
          id,
          start _ timestamp
    },
    modality,
    command: {
          id,
          options,
          priority,
          resumable,
          resources: [{
                id,
                requiredAmount,
                requiredTime
          }]
    }
}
```

От этих параметров зависит приоритет заявки. Таким образом, для вычисления QoS предоставляемых пользователю услуг нужно определить:

- исходные данные;
- методику определения приоритетов;
- стратегию управления заявками.

Рассмотрим каждый из перечисленных пунктов подробнее. К исходным данным относится информация о характере процесса поступления заявок каждого вида. Условия возникновения заявок строго не определены и могут меняться в зависимости от контекста. Однако при использовании центральной предельной теоремы теории вероятностей для потоков событий процесс поступления одиночных заявок можно описать с применением распределения Пуассона и использовать параметр распределения λ для характеристики интенсивности поступления заявок.

При построении методики определения приоритетов необходимо использовать следующие принципы. Во-первых, необходимо принимать во внимание вероятность точного восприятия и распознавания команд. Так, наиболее точным будет восприятие команд, отданных с мобильных устройств, так как для этого пользователю необходимо целенаправленно открыть приложение сервиса, установить сессию, просканировав QR-код устройства и выполнить команду в приложении. Вероятность случайного выполнения этих действий крайне мала. В случае, когда используется модальность речи, вероятность ошибки выше, а при управлении жестами она еще выше [20].

Во-вторых, необходимо учитывать достоверность и легитимность воспринимаемых сигналов. Отсутствие достоверности может проявляться в том, что поступающий от пользователя ряд заявок не выглядит как сгенерированный человеком (например, очень высокая частота поступления заявок, противоречивость заявок от одного пользователя). Это может быть связано с ошибками восприятия, но может быть и вредоносной активностью (например, когда к интерфейсу сервиса подключается бот). В этом случае речь идет о нелегитимности сигналов. В том и другом случае ряд заявок отклоняется или переводится на более низкий приоритет. Для этого могут использоваться методы фильтрации, описанные в разд. 1.

В-третьих, при расстановке приоритетов заявок следует учитывать возможность пользователя воспринимать ответ сервиса. Так, если пользователь оставил заявку, предполагающую вывод информации на дисплей, но не находится в зоне видимости дисплея, выполнение заявки может быть отложено до возвращения пользователя или отменено по истечении тайм-аута.

В-четвертых, необходимо учитывать свойства самой задачи, в частности, срочность, которая определяется типом задачи автоматически (на основе приоритета пользователя или процесса) или администратором вручную, способность задачи к вытеснению с последующим возобновлением.

Множество стратегий управления заявками достаточно изучено в теории массового обслуживания [21]. Дисциплины буферизации и обслуживания различаются по признакам наличия приоритета, по правилам выбора и вытеснения заявок, по длине очереди и т. п. При поступлении заявки возможны следующие способы ее обработки: немедленное исполнение (с возможным вытеснением активной заявки); отклонение заявки; постановка заявки в очередь.

В последнем случае могут быть использованы дисциплины обслуживания FIFO или LIFO. Для симуляции процесса обслуживания заявок была построена модель, описанная в следующем разделе.

3. Эксперименты

Экспериментальные исследования предложенной модели состояли в симуляции потока входящих заявок и процесса их обработки. Результат симуляции при таком подходе отражает эффективность управляющей стратегии. При моделировании были использованы некоторые допущения. В рамках рассматриваемой модели не учитываются фоновые задачи, так как их выполнение никак не влияет на задачи более высокого приоритета. Критические задачи также не рассматриваются в силу их нерегулярности и малой вероятности появления. Кроме того, в рамках используемого подхода для упрощения в ходе экспериментов моделируются только локальные заявки. В общем виде процесс обработки новой заявки сводится к следующим шагам:

1) если в настоящий момент времени для обработки заявки имеется достаточно свободного ресурса, то заявка обрабатывается немедленно;

2) если в настоящий момент времени ресурс занят заявкой того же пользователя, то ее обработка отменяется, а новая заявка обрабатывается немедленно;

3) в иных случаях используется одна из стратегий, рассмотренных далее.

К стратегиям без приоритета относятся:

- система с отказами, согласно которой обработка заявки отменяется, если ресурс занят;
- система с ограниченным временем ожидания, когда новая заявка обрабатывается после предыдущих, если не прошел некоторый заранее заданный период времени.

К стратегиям с приоритетом относятся:

- сдвиг задач с более низким приоритетом;
- вытеснение задач с более низким приоритетом;
- отмена задач с более низким приоритетом.

Для оценки качества процесса обработки заявок можно измерять следующие величины:

- относительная доля выполненных заявок (относительная пропускная способность);
- доля выполненных заявок, взвешенных по их приоритету;
- среднее время ожидания при выполнении заявок;
- время загрузки канала.

Для экспериментов использовалась модель системы, написанная на языке Python (рис. 2).

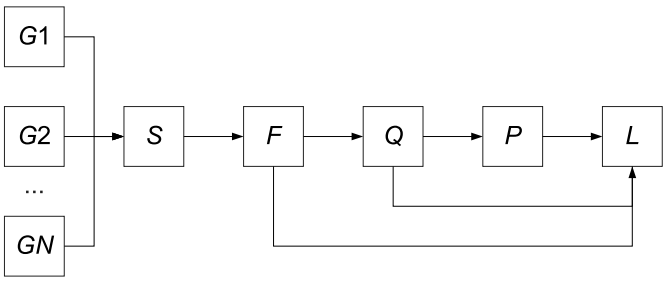


Рис. 2. Модель системы массового обслуживания

Представленная на рис. 2 модель состоит из следующих блоков: генераторы потоков *G*1...*GN*; диспетчер сессий *S*; фильтр *F*; контроллер очереди *Q*; обработчик запросов *P*; логгер *L*. Генераторы потоков создают объекты класса "Поток", которые моделируют пользовательские сессии. Они характеризуются паттернами трафика и его распределениями. Кроме того, для каждого потока задается его длительность. Диспетчер сессий сохраняет объекты потоков и генерирует запросы для каждого активного потока в соответствии с его параметрами. Фильтр реализует меры защиты, которые зависят от конфигурации системы защиты. Контроллер очереди передает запросы обработчику, используя данные об очередности, приоритете запросов и о стратегиях управления. Логгер принимает сведения обо всех запросах и их статусе по окончании их обработки и сохраняет эти сведения в базу данных.

Рассмотрим для начала модель, в которой не используется фильтрация (т. е. блок *F* на рис. 2 передает все заявки в блок *Q*). В модели использованы следующие исходные данные: *N* типов одиночных задач от разных пользователей, появляющихся с интенсивностями λ_i и собственными приоритетами P_i, определенными исходя из используемых модальностей. В качестве задач рассматриваются запросы пользователей на выдачу информации. Выполнено моделирование для стратегий, описанных выше. При моделировании учитывались три приоритета, связанные с используемыми модальностями. При поступлении новой заявки в зависимости от приоритетов конфликтующей заявки выполняется одно из действий A_H (приоритет новой заявки выше), A_E (приоритеты равны), A_L (приоритет ниже). Если $A_H = A_L = A_E = A$, применяемая стратегия не учитывает приоритет.

При моделировании результат выполнения каждой задачи характеризуется степенью успешности ее решения и задержкой (временем ожидания). Стратегии, при которых преобладает ожидание, порождают более высокую задержку и одновременно большую степень успешности ее реализации. Для стратегий с преобладанием отказов результаты противоположные. Для комплексной оценки стратегий можно составить показатель эффективности выполнения задачи *i*, зависящий от обеих величин. Так, в случае отказа можно принять значение задержки $D_i = \infty$, а для перехода к конечным величинам использовать оценку

$$E_i = \frac{D_0}{D_0 + D_i},$$

где D_0 — значение задержки, при котором показатель эффективности уменьшается вдвое. При рассмотрении стратегий, при которых задача может быть прервана, необходимо также учитывать фактическое время ее выполнения. В этом случае показатель E_i можно обобщить в виде

$$E_i = \frac{T_{F_i} D_0}{T_{P_i}\left(D_0 + D_i\right)},$$

где T_{Fi} — фактическое время выполнения; T_{Pi} — планируемое. Для получения интегрированной оценки той или иной стратегии с учетом приоритетов задач можно использовать показатель

$$E = \frac{\sum_{i=1}^{N} P_i E_i}{\sum_{i=1}^{N} P_i},$$

где P_i — приоритет i-й задачи; N — число задач.

Чтобы учесть легитимность запросов, можно ввести логическую оценку $C_i \in [0; 1]$, где 0 означает отсутствие легитимности, а 1 характеризует доверенный запрос. Тогда представленная выше формула может быть обобщена следующим образом:

$$E = \frac{\sum_{i=1}^{N} P_i C_i E_i}{\sum_{i=1}^{N} P_i}.$$

В более общем случае может применяться нечеткая оценка легитимности $C_i \in [0; 1]$.

При низкой интенсивности поступления заявки не конфликтуют, значение эффекта E близко к 1 и почти не зависит от стратегии обработки заявок. При увеличении интенсивности стратегии показывают разный эффект. На рис. 3 показаны зависимости эффекта от интенсивности при стратегиях без приоритета.

Рис. 3 показывает, что без учета приоритетов стратегии с отказами более эффективны, чем стратегии с очередями в случае, если приоритет заявок не учитывается. Это, в частности, объясняется чувствительностью системы к задержкам. При экспериментах использовалось значение $D_0 = T_P$, что с точки зрения человеко-машинного взаимодействия означает довольно низкую чувствительность. Причина в том, что время реакции интерфейса, при котором система воспринимается пользователем как управляемая, не превышает нескольких секунд [22]. Очевидно, что более жесткие условия при задании величины D_0 только уменьшат значение эффекта для этих стратегий.

Заметно также, что в стратегиях с очередями ограничение длины очереди положительно влияет на значение величины эффекта. Подобное ограничение приводит к отмене задач с большим значением задержки, которые занимают ресурс, но не вносят существенного вклада в значение величины эффекта. Тем не менее такие стратегии все равно уступают стратегиям с отказами, которые являются их частным случаем при нулевой длине очереди.

Теперь рассмотрим стратегии с учетом приоритета и сравним их с бесприоритетными стратегиями с отказами. Рассматривалось множество стратегий, у которых среди действий A_H, A_L, A_E присутствуют хотя бы два различных, и при этом заявки с более высоким приоритетом не могут получать менее приоритетное обслуживание.

Графики для выявленных наиболее эффективных стратегий приведены на рис. 4.

На рис. 4 обозначены следующие эффекты для стратегий: *1* — отклонение новой заявки (A — отказ); *2* — отклонение конкурирующих заявок (A — вытеснение); *3* — отклонение конкурирующих заявок при более высоком или равном приоритете новой заявки (A_L — отказ, A_E, A_H — вытеснение); *4* — отклонение конкурирующих заявок только при более высоком приоритете новой заявки (A_L, A_E — отказ,

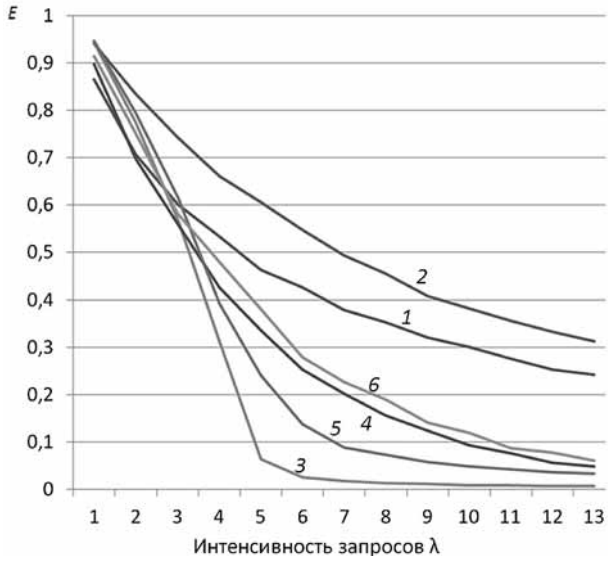


Рис. 3. Зависимость эффекта от интенсивности при различных стратегиях обработки заявок:

1 — отклонение новой заявки; *2* — отклонение конкурирующих заявок; *3* — FIFO, *4* — LIFO; *5* — FIFO (максимальная задержка 1); *6* — LIFO (максимальная задержка 1)

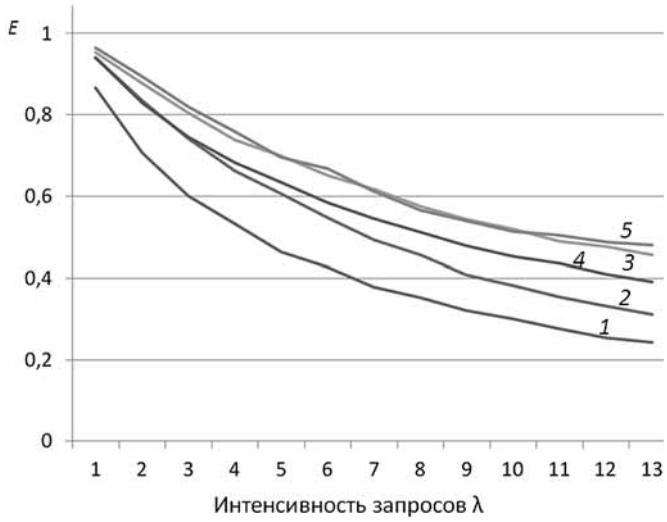


Рис. 4. Сравнение бесприоритетных (*1*, *2*) и приоритетных (*3*, *4*, *5*) стратегий

A_H — вытеснение); *5* — очередь FIFO только при более низком приоритете новой заявки, иначе отклонение конкурирующих заявок (A_L — FIFO, A_E, A_H — вытеснение).

Для большинства стратегий моделирование показывает, что постановка заявки в очередь приводит к меньшему значению эффекта. Однако использование очереди FIFO при обработке задач с низким приоритетом приводит к небольшому повышению эффекта. Этот факт можно объяснить тем, что использование стратегий с отказами в этом случае приведет к отмене низкоприоритетных заданий и их эффекту $E_i = 0$, а при использовании очереди эти задания имеют шанс на выполнение с задержкой при условии отсутствия заданий с более высоким приоритетом ($E_i \geqslant 0$).

Эксперименты с другими приоритетными стратегиями показывают более низкую эффективность, что отражено на рис. 5.

На рис. 5 обозначены следующие стратегии:

6: A_L — отказ; A_E — вытеснение; A_H — LIFO;

7: A_L, A_E — вытеснение; A_H — LIFO;

8: A_L — FIFO; A_E — вытеснение; A_H — LIFO;

9: A_L — FIFO; A_E, A_H — LIFO;

10: A_L — FIFO; A_E — LIFO; A_H — вытеснение.

Таким образом, в рассматриваемых условиях можно считать оптимальной стратегию *5*, предусматривающую использование дисциплины FIFO, если новая заявка имеет более низкий приоритет, и отклонение конкурирующих заявок в остальных случаях.

Теперь рассмотрим ситуацию, когда в системе обработки заявок в совокупности с определенной ранее стратегией *5* используется модуль фильтрации. При возрастании интенсивности запросов при наличии угрозы и отсутствии фильтрации часть запросов будет отклоняться при реализации управляющей стратегии. В частности, при наличии угроз доступности типа DDoS-атак легитимные запросы могут быть отклонены, а нелегитимные — выполнены, причем ввиду большого числа нелегитимных запросов при DDoS-атаке последние будут чаще передаваться на исполнение. Следовательно, оценка эффекта будет уменьшаться. На рис. 6 представлена зависимость значения эффекта от средней интенсивности возникновения запросов λ_R и потоков λ_S при условии, что потоки легитимны.

Значение эффекта также значительно снижается при появлении паттернов нелегитимных запросов с интенсивностями λ_M (рис. 7).

Фильтрация позволяет частично устранить нелегитимные запросы и увеличить значения эффекта. Было проведено моделирование для оценки значений эффекта с использованием фильтрации по разным пороговым значениям статистических моментов и энтропии. Рис. 8 (см. третью сторону обложки) иллюстрирует изменение эффекта при изменении

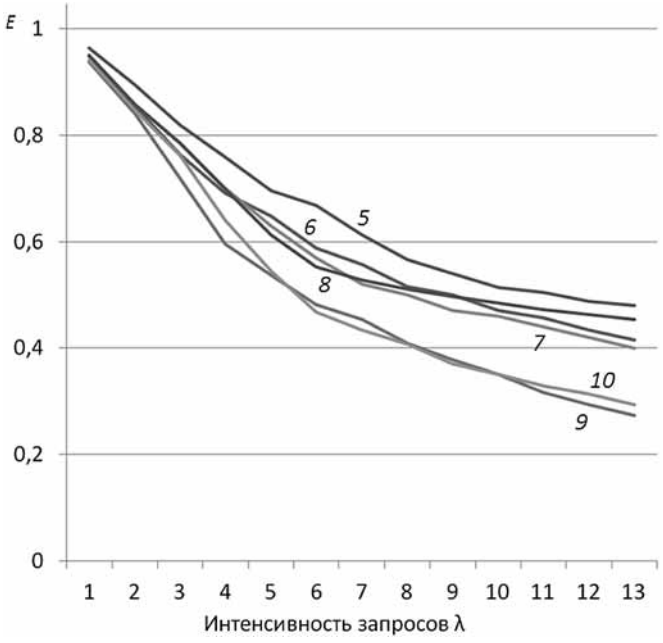


Рис. 5. Эффективность приоритетных стратегий

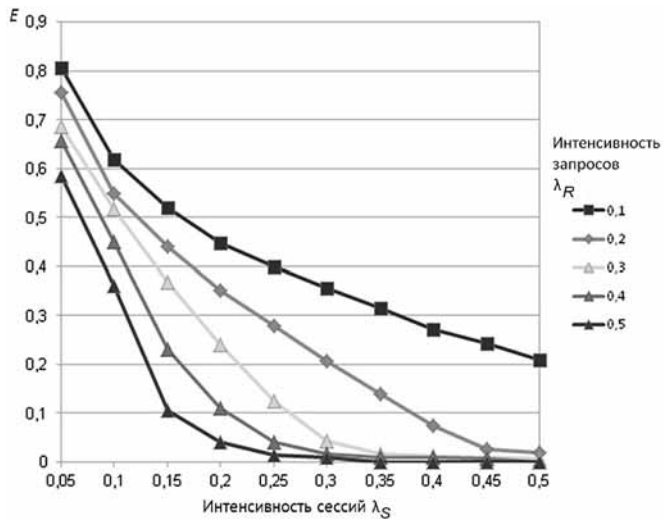


Рис. 6. Зависимость эффекта от λ_R и λ_S

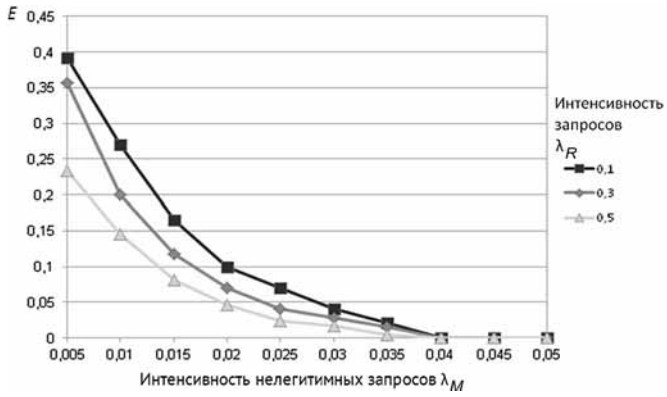


Рис. 7. Зависимость эффекта от λ_R и λ_M (λ_S = 0,1)

порога оценки математического ожидания для частоты появления запросов. Рис. 9 (см. третью сторону обложки) характеризует ту же величину в зависимости от порога среднеквадратического отклонения. Рис. 10 (см. третью сторону обложки) отражает зависимость эффекта от допустимого нижнего порога энтропии.

Эксперимент показывает, что значение эффекта имеет максимумы, которые зависят как от выбранных пороговых значений, так и от условий функционирования (интенсивностей запросов). Слишком низкие значения порогов приводят к множеству ошибок первого рода, высокие — к ложным срабатываниям. Таким образом, задача определения параметров для методов фильтрации может быть представлена как оптимизационная задача в многомерном пространстве, где измерения соответствуют параметрам применяемых методов фильтрации. Изменение условий функционирования системы может быть учтено с помощью адаптивного подхода к управлению методами фильтрации.

Заключение

Рассмотренные исследования позволяют сделать человеко-машинное взаимодействие более дружественным, повысить качество обслуживания пользователей и их комфорт при их взаимодействии с интеллектуальным пространством.

К дальнейшим направлениям исследований относятся перечисленные далее.

- Анализ свойств сессионных заявок, определение их легитимности и достоверности и поиск оптимальных стратегий обработки таких заявок. При поступлении сессионных заявок поток событий перестает быть ординарным. Однако процессы возникновения новых сессий и возникновения заявок внутри одной сессии можно приближенно считать ординарными и пуассоновскими, что позволяет частично использовать модели, описанные ранее.
- Исследование ситуации, когда система может взаимодействовать с пользователем с помощью нескольких каналов выдачи данных (как однородных, так и использующих разные модальности). К таким каналам могут относиться видео- и аудиоканалы стационарных компонентов интеллектуального пространства, портативных устройств пользователей, каналы коммуникации с роботами. В этом случае необходимо рассматривать разделы теории массового обслуживания, связанные с многоканальными системами.
- Исследование влияния разных типов задач на эффективность интеллектуального пространства. В частности, это включает классификацию и анализ задач других сервисов МИНОС, помимо Digital Signage (информационно-навигационный сервис, сервис видеоконференцсвязи и т. п.).
- В области безопасности необходимо исследование вопросов доверия системы к контрагенту. В частности, система должна быть устойчива к атакам со стороны контрагента, что может быть обеспечено с помощью интеграции с известными методами борьбы с атаками, актуальными для интеллектуального пространства (например, DoS, спуфинг) [23]. Известны также способы выявления недостоверной (намеренной или ошибочной) информации, применение которых позволит защититься от подобных атак [24].
- Другим направлением в области защиты от ошибочной информации является совместная обработка данных, поступающих от пользователя, посредством разных модальностей [25].

Работа выполнена при финансовой поддержке РНФ (грант 16-19-00044).

Список литературы

1. **Zeng X., Pei H.** Human-Computer Interaction in Ubiquitous Computing Environments // International Conference on Information Computing and Applications 2012. Vol. 308. P. 628—634.

2. **Юсупов Р. М., Ронжин А. Л.** От умных приборов к интеллектуальному пространству // Вестник Российской академии наук. 2010. № 80. С. 63—68.

3. **Levonevskiy D., Vatamaniuk I., Saveliev A.** Processing models for conflicting user requests in ubiquitous corporate smart spaces // MATEC Web Conf. 13th International Scientific-Technical Conference on Electromechanics and Robotics "Zavalishin's Readings". 2018. Vol. 161. Article 03006. DOI: 10.1051/matecconf/201816103006.

4. **Kwon A., Kang J.-M., Seo S.-S., Kim S.-S., Chung J. Y., Strassner J., Hong J. W.-K.** The design of a quality of experience model for providing high quality multimedia services// IEEE International Workshop on Modelling Autonomic Communications Environments. 2010. P. 24—26.

5. **GB923:** Wireless service measurement Handbook. TM Forum, 2010. URL: https://www.tmforum.org/resources/best-practice/gb923-wireless-service-measurement-handbook/ (дата обращения: 14.10.2018).

6. **Cheong F., Lai R.** QoS specification and mapping for distributed multimedia systems: A survey of issues // Journal of Systems and Software. 1999. Vol. 45, No. 2. P. 127—132.

7. **Hossain M. A., Islam A., Le N. T., Lee Y. T., Lee H. W., Jang Y. M.** Performance analysis of smart digital signage system based on software-defined IoT and invisible image sensor communication // International Journal of Distributed Sensor Networks. 2016. Vol. 7, No. 12. P. 1—14.

8. **Wang Z., Zeng K., Rehman A., Yeganeh H., Wang S.** Objective video presentation QoE predictor for smart adaptive video streaming // Applications of Digital Image Processing XXXVIII. 2015. Vol. 9599. 95990Y.

9. **Lin Y. T., Oliveira E. M. R., Jemaa S. B., Elayoubi S. E.** Machine learning for predicting QoE of video streaming in mobile networks // 2017 IEEE International Conference on Communications (ICC). 2017. P. 1—6.

10. **Nam H., Kim K. H., Kim J. Y., Schulzrinne H**. Towards QoE-aware video streaming using SDN // Global Communications Conference (GLOBECOM). 2014. P. 1317—1322.

11. **Xu Y., Elayoubi S. E., Altman E., El-Azouzi R., Yu Y.** Flow-Level QoE of Video Streaming in Wireless Networks // IEEE Transactions on Mobile Computing. 2016. Vol. 15, No. 11. P. 2762—2780.

12. **Oh H., Lee H., Lee I., Lee S.** Cooperative content and radio resource allocation for visual information maximization in a digital signage scenario // Digital Signal Processing. 2015. Vol. 45. P. 24—35.

13. **Sarwar G., Ullah F., Lee H. W., Ryu W., Lee S.** Control framework and services scenarios of provisioning n-screen services in interactive digital signage // Tehnički vjesnik. 2017. Vol. 24, No. 1. P. 177—185.

14. **Shin I., Lee S., Lee E., Lee N. K., Lee H.** S/W Based Frame-Level Synchronization for Irregular Screen Processing System // ETRI Journal. 2016. Vol. 38, No. 5. P. 868—878.
15. **Sarwar G., Ullah F., Lee S.** QoS and QoE Aware N-Screen Multicast Service // Journal of Sensors. 2016. Vol. 2016, Article ID 8040138.
16. **Бабенко Г. В., Белов С. В.** Анализ трафика TCP/IP на основе методики допустимого порога и отклонения как инструмент определения инцидентов информационной безопасности // Технологии техносферной безопасности. 2011. № 5 (39). С. 1—9.
17. **Lu W., Traore I.** An unsupervised approach for detecting DDoS attacks based on traffic-based metrics // In: PACRIM 2005, 2005. P. 462—465.
18. **Bellaïche M., Grégoire J.-C.** SYN flooding attack detection based on entropy computing // GLOBECOM 2009 — 2009 IEEE Global Telecommunications Conference. 30 Nov.— 4 Dec. 2009 Honolulu, HI, USA. IEEE, 2009. doi: 10.1109/GLOCOM.2009.5425454.
19. **Vatamaniuk I., Levonevskiy D., Saveliev A., Denisov A.** Scenarios of Multimodal Information Navigation Services for Users in Cyberphysical Environment // 18th International Conference on Speech and Computer (SPECOM—2016). 2016. LNAI 9811, P. 588—595.
20. **Liu H., Wang L.** Gesture recognition for human-robot collaboration: A review // International Journal of Industrial Ergonomics. In press (2017). URL: https://www.sciencedirect.com/science/article/abs/pii/S0169814117300690
21. **Матвеев В. Ф., Ушаков В. Г.** Системы массового обслуживания. М.: МГУ, 1984. 242 с.
22. **Henty S.** UI Response Times. URL: https://medium.com/@slhenty/ui-response-times-acec744f3157 (дата обращения: 14.10.2018).
23. **Eder-Neuhauser P., Zseby T., Fabini J., Vormayr G.** Cyber attack models for smart grid environments // Sustainable Energy, Grids and Networks. 2017. Vol. 12. P. 10—29.
24. **Budkov V., Vatamaniuk I., Basov V., Volf D.** Investigation of Speech Signal Parameters Reflecting the Truth of Transmitted Information // 18th International Conference on Speech and Computer (SPECOM—2016). 2016. LNAI 9811. P. 419—426.
25. **Margiotoudi K., Kelly S., Vatakis A.** Audiovisual Temporal Integration of Speech and Gesture // Procedia — Social and Behavioral Sciences. 2014. Vol. 126. P. 154—155.

Ensuring the Availability of Services of the Corporate Intellectual Space by Controlling the Flow of Input Data

D. K. Levonevskiy, DLewonewski.8781@gmail.com, **I. V. Vatamaniuk,** vatamaniuk@iias.spb.su, **D. A. Malov,** malovdmitrij@gmail.com, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, St. Petersburg, 199178, Russian Federation

Corresponding author:

Levonevskiy Dmitriy K., Researcher, St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences, St. Petersburg, 199178, Russian Federation,
E-mail: DLewonewski.8781@gmail.com

Received on August 15, 2018
Accepted on August 20, 2018

This paper considers providing availability of the corporate smart space services. Intended and unintended impacts on the smart space, such as conflicting user requests, exhaustion of resources, network attacks are considered availability threats.

In order to estimate availability, it is proposed to measure parameters that determine quality of experience (QoE) and, in particular, quality of service (QoS). Such parameters include, for instance, average delays, average request processing times, denial rates.

As an example, the paper discusses the process of communication between users and the information service of MINOS (Multimodal Information and Navigation Cloud System). Users can send requests to the service using various modalities such as speech, gestures, or control the service using their mobile devices or personal computers. In conditions of availability threats the service may be unable to allocate the required resource for all pending requests. In this case, it is necessary to find the optimal strategy of request processing that will maximize the QoS values. The optimality of strategies depends on a set of parameters (request and resource intensity, number of users, etc.).

A mass queuing model was built to conduct the experiments. These experiments simulate different strategies of processing requests and methods of their filtering. The experiments show that the effect estimate has maximum values depending on the conditions of the applied system operating. Thereby, the task of determining the parameters of control strategies and filtering methods can be represented as an optimization task in a multidimensional space. The changing conditions of operating may be taken into account using adaptive approaches. The conducted research helps improving the quality of service and user experience in human-computer interaction in corporate smart spaces by means of providing the availability of the smart space services.

Keywords: *smart spaces, multimodal interfaces, corporate information systems, information and navigation systems, cloud systems, computer security, availability*

For citation:

Levonevskiy D. K., Vatamaniuk I. V., Malov D. A. Ensuring the Availability of Services of the Corporate Intellectual Space by Controlling the Flow of Input Data*, Programmnaya Ingeneria,* 2019, vol. 10, no. 1, pp. 20—29

DOI: 10.17587/prin.10.20-29.

References

1. **Zeng X., Pei H.** Human-Computer Interaction in Ubiquitous Computing Environments, *International Conference on Information Computing and Applications,* 2012, vol. 308, pp. 628—634.

2. **Yusupov R. M., Ronzhin A. L.** Ot umnykh priborov k intellektual'nomu prostranstvu (From Smart Devices to Smart Space), *Vestnik Rossijskoj akademii nauk*, 2010, no. 80, pp. 63—68 (in Russian).

3. **Levonevskiy D., Vatamaniuk I., Saveliev A.** Processing models for conflicting user requests in ubiquitous corporate smart spaces, *MATEC Web Conf. 13th International Scientific-Technical Conference on Electromechanics and Robotics "Zavalishin's Readings",* 2018, vol. 161, article 03006, DOI: 10.1051/matecconf/201816103006.

4. **Kwon A., Kang J.-M., Seo S.-S., Kim S.-S., Chung J. Y., Strassner J., Hong J. W.-K.** The design of a quality of experience model for providing high quality multimedia services*, IEEE International Workshop on Modelling Autonomic Communications Environments*, 2010, pp. 24—26.

5. **GB923:** Wireless service measurement Handbook / TM Forum, 2010, available at: https://www.tmforum.org/resources/best-practice/gb923-wireless-service-measurement-handbook/ (date of access: 14.10.2018).

6. **Cheong F., Lai R.** QoS specification and mapping for distributed multimedia systems: A survey of issues*, Journal of Systems and Software*, 1999, vol. 45, no. 2, pp. 127—132.

7. **Hossain M. A., Islam A., Le N. T., Lee Y. T., Lee H. W., Jang Y. M.** Performance analysis of smart digital signage system based on software-defined IoT and invisible image sensor communication*, International Journal of Distributed Sensor Networks,* 2016, vol. 7, no. 12, pp. 1—14.

8. **Wang Z., Zeng K., Rehman A., Yeganeh H., Wang S.** Objective video presentation QoE predictor for smart adaptive video streaming*, Applications of Digital Image Processing* XXXVIII, 2015, vol. 9599, 95990Y.

9. **Lin Y. T., Oliveira E. M. R., Jemaa S. B., Elayoubi S. E.** Machine learning for predicting QoE of video streaming in mobile networks*,* 2017 *IEEE International Conference on Communications (ICC),* 2017, pp. 1—6.

10. **Nam H., Kim K. H., Kim J. Y., Schulzrinne H.** Towards QoE-aware video streaming using SDN, *Global Communications Conference (GLOBECOM),* 2014, pp. 1317—1322.

11. **Xu Y., Elayoubi S. E., Altman E., El-Azouzi R., Yu Y.** Flow-Level QoE of Video Streaming in Wireless Networks, *IEEE Transactions on Mobile Computing,* 2016, vol. 15, no. 11, pp. 2762—2780.

12. **Oh H., Lee H., Lee I., Lee S.** Cooperative content and radio resource allocation for visual information maximization in a digital signage scenario, *Digital Signal Processing*, 2015, vol. 45, pp. 24—35.

13. **Sarwar G., Ullah F., Lee H. W., Ryu W., Lee S.** Control framework and services scenarios of provisioning n-screen services in interactive digital signage*, Tehnički vjesnik*, 2017, vol. 24, no. 1, pp. 177—185.

14. **Shin I., Lee S., Lee E., Lee N. K., Lee H.** S/W Based Frame-Level Synchronization for Irregular Screen Processing System, *ETRI Journal*, 2016, vol. 38, no. 5, pp. 868—878.

15. **Sarwar G., Ullah F., Lee S.** QoS and QoE Aware N-Screen Multicast Service, *Journal of Sensors,* 2016, vol. 2016, article ID 8040138.

16. **Babenko G. V., Belov S. V.** Analiz trafika TCP/IP na osnove metodiki dopustimogo poroga i otkloneniya kak instrument opredeleniya intsidentov informatsionnoj bezopasnosti (Analysis of TCP/IP traffic based on the methodology of specified threshold and the deviation as a tool of detecting information security accidents), *Tekhnologii tekhnosfernoj bezopasnosti*, 2011, no. 5 (39), pp. 1—9 (in Russian).

17. **Lu W., Traore I.** An unsupervised approach for detecting DDoS attacks based on traffic-based metrics*, PACRIM* 2005, 2005, pp. 462—465.

18. **Bellaïche M., Grégoire J.-C.** SYN flooding attack detection based on entropy computing, *GLOBECOM 2009 — 2009 IEEE Global Telecommunications Conference,* 30 Nov.—4 Dec., 2009, Honolulu, HI, USA, IEEE, 2009. DOI: 10.1109/GLOCOM.2009.5425454.

19. **Vatamaniuk I., Levonevskiy D., Saveliev A., Denisov A.** Scenarios of Multimodal Information Navigation Services for Users in Cyberphysical Environment, *18th International Conference on Speech and Computer (SPECOM-2016)*, 2016, LNAI 9811, pp. 588—595.

20. **Liu H., Wang L.** Gesture recognition for human-robot collaboration: A review, *International Journal of Industrial Ergonomics.* In press (2017), available at: https://www.sciencedirect.com/science/article/abs/pii/S0169814117300690

21. **Matveev V. F., Ushakov V. G.** *Sistemy massovogo obsluzhivaniya* (Mass queuing systems), Moscow, MGU, 1984, 242 p. (in Russian).

22. **Henty S.** UI Response Times, available at: https://medium.com/@slhenty/ui-response-times-acec744f3157 (date of access 14.10.2018).

23. **Eder-Neuhauser P., Zseby T., Fabini J., Vormayr G.** Cyber attack models for smart grid environments*, Sustainable Energy, Grids and Networks*, 2017, vol. 12, pp. 10—29.

24. **Budkov V., Vatamaniuk I., Basov V., Volf D.** Investigation of Speech Signal Parameters Reflecting the Truth of Transmitted Information, *18th International Conference on Speech and Computer (SPECOM—2016),* 2016, LNAI 9811, pp. 419—426.

25. **Margiotoudi K., Kelly S., Vatakis A.** Audiovisual Temporal Integration of Speech and Gesture, *Procedia — Social and Behavioral Sciences,* 2014, vol. 126, pp. 154—155.

УДК 004.89 DOI: 10.17587/prin.10.30-37

Е. И. Бурлаева, аспирант, e-mail: ekaterina0853@mail.ru, **В. Н. Павлыш,** д-р техн. наук, проф., зав. кафедрой, e-mail: pavlyshvn@mail.ru, Донецкий национальный технический университет

Анализ методов преобразования текстов в форму объектов векторного пространства

Одной из востребованных технологий обработки текстовой информации является автоматическая классификация документов, представленных в текстовом виде. Традиционное представление документа в форме последовательности символов затрудняет работу с ним как с объектом классификации. Большинство алгоритмов машинного обучения работают с такими документами как с элементами векторного пространства, вследствие чего появляется необходимость соответствующего преобразования текстов в форму векторного объекта.

В статье представлен подход к преобразованию текста в форму векторного объекта, использующий композицию методов. На основании проведенных экспериментов, позволяющих сравнивать эффективности методов векторизации и морфологического разбора, выбран подход к сокращению размерности векторов, использующий сочетание методов "стемминг", "стоп-слова" и tf-idf. Такая композиция, как показали эксперименты, позволяет облегчить работу с использованием метода tf-idf, избавляя текст от неинформативных слов и преобразуя слова к общей форме.

Ключевые слова: *векторное представление, текстовый документ, слово, композиция методов, tf-idf, классификация, стемминг, стоп-слова, нижняя граница*

Введение

Тенденция распространения и увеличения объемов информации в электронном виде стимулирует активное развитие автоматических систем обработки информации. В большинстве организаций значительная часть полезной информации содержится в электронных базах данных. Такая ситуация вызывает повышенный интерес к методам Text mining — автоматического извлечения и обработки знаний из текстовых документов.

Поиск и структуризация по содержанию большого количество текстов довольно сложная для эксперта задача. Ее выполнение ограниченным количеством специалистов и затрачиваемым временем в условиях постоянного поступления новой информации практически невозможно. Как следствие, для решения этой задачи все чаще применяют автоматические системы анализа текста [1]. Получение знаний в автоматическом режиме затрудняется слабой структурированностью текстов, качеством словообразования, описывающего тему исходного текста [2].

Извлечение и структуризация знаний имеют своей конечной целью информационную поддержку эксперта или автоматизированной системы при принятии проектных решений, полезных в различных областях деятельности человека. Таким образом, основной функцией систем извлечения и структуризации знания является информационный поиск полезных сведений в документальных базах. Проблема поиска, извлечения и структуризации информации является актуальной для построения систем обработки, анализа, оценивания и понимания информации. Конечной практической целью здесь является поиск наиболее рационального варианта отнесения той или иной текстовой информации в определенную тематическую группу.

Для структуризации текста разработан ряд методов [3—5]. Работа над новыми решениями продолжается, а также повышается эффективность уже существующих подходов [6—8]. Несмотря на определенные различия, большая часть методов имеет одну общую особенность. Ее суть в том, что с использованием некоторой эвристики с их помощью находится группа слов, которая точно определяет темы или описывает информацию, содержащуюся в исходном тексте. Например, расстояние между словами, частоту использования слов или заранее определенные отношения между словами.

С учетом частоты встречаемости общей лексики в текстах разного назначения наиболее естественный путь решения отмеченных выше задач состоит в использовании известной статистической меры *tf-idf* [9]. Ее используют для выделения среди слов исходной фразы общей лексики и слов терминов (в том числе в составе сочетаний).

В настоящей работе анализируются функциональные возможности сокращения размерности векторов, в которые преобразуются тексты для поиска в корпусе описаний близких фрагментов знаний и языковых форм их выражения. Рассмотрим некоторые из методов, способствующих сокращению размерности

такого вектора для автоматической классификации текста. Выделим их преимущества и недостатки, что может послужить отправной точкой для разработки более эффективных подходов.

1. Уменьшение размерности признакового пространства

Одной из сложностей процедур анализа текстов является большое число слов в документе. Если каждое из этих слов подвергать анализу, то время поиска новых знаний резко возрастет. В то же время не все слова в тексте несут полезную информацию. Таким образом, удаление неинформативных слов, а также приведение близких по смыслу слов к единой форме значительно сокращают время анализа текстов [10, 11].

При уменьшении размера текста важно не ухудшить качество классификации. При этом, основываясь на объективных доводах, предполагается, что ряд термов практически неинформативны, ряд термов зависит друг от друга. Таким образом, они с большой вероятностью попадут в один класс или в один документ. Очевидно, что за счет таких признаков можно уменьшить размерность пространства без ухудшения точности классификации.

Уменьшение размерности может проходить локально или глобально. Термин "локально" означает, что процедура проводится для каждой категории отдельно, затем оставшиеся слова объединяются в единое пространство. "Глобально" подразумевает, что участие принимают сразу все документы обучающей выборки, без учета их принадлежности к той или иной категории.

2. Отбор неинформативных признаков

Существуют различные способы определения признаков, не влияющих на качество классификации [12].

Разумно предположить, что если какой-либо терм практически не встречается в документах обучающей выборки, то он не несет специфической информации, определяющей класс объекта. Когда же терм, наоборот, встречается во всех документах и много раз, то он также будет нести мало полезной информации о принадлежности документа к тому или иному классу. Если все классы статистически независимы от какого-либо терма, то слово также неинформативно. Такие простые правила, конечно, нуждаются в формализации.

Набор алгоритмов, которые занимаются сопоставлением отдельных слов и словоформ в словаре (лексиконе, если быть точным) и выяснением грамматических характеристик слов, называется морфологическим анализатором.

Выделяют [13] два типа морфологических анализаторов:

- словарные (точный морфологический анализ);
- бессловарные (приближенный морфологический анализ или аналитические методы).

Словарные анализаторы используют методы, основанные на словарях. В словарных анализаторах преобразование слова в лемму проводится с помощью специальной таблицы (словаря), которая содержит отображение множества слов на множество лемм.

Методы, основанные на словарях, позволяют определять грамматические характеристики, без которых не представляется возможным, например, проводить фрагментацию текстов. В морфологическом словаре приводятся все формы слов, каждой из которых поставлены в соответствие все необходимые признаки (в частности грамматические, включающие число, падеж, лицо, время, наклонение и другие).

Точный подход к морфологическому анализу основывается на использовании словарей, в которых для каждого слова указано правило изменения его формы. Эти процедуры могут применяться только к словам, основы которых включены в словарь.

Главный недостаток словарных анализаторов состоит в том, что получить какую-либо морфологическую информацию об анализируемой словоформе невозможно, если ее нет в словаре. Учитывая, что неизменяемое лексическое ядро естественного языка составляет порядка 80 % словарного запаса, применение только словарных анализаторов не решает проблему определения всех возможных слов в полном объеме [14].

Бессловарные анализаторы используют аналитические методы. Они содержат набор правил морфологических преобразований. Для русского языка это в основном таблицы суффиксов и условий их отсечения, с помощью которых данное слово преобразуется в некоторую нормальную форму.

Выделяют также системы на основе стемминга. В случае стемминга зачастую отбрасывается вся морфологическая информация, а в качестве нормальной формы берется неизменяемая псевдооснова, называемая стем [14].

Морфология с использованием стемминга обладает рядом достоинств. Так, например, за счет упрощения алгоритма и уменьшения объема выдаваемой информации существенно (до нескольких раз) возрастает скорость анализа, а при использовании лишь массива парадигм сокращается объем хранимых баз.

Главным достоинством морфологии на основе стемминга является тот факт, что при отсутствии словаря основ (стемов) фактически получается морфологическая база неограниченного объема, настраиваемая непосредственно на имеющийся текст. Это очень удобно при создании информационно-поисковых систем с нефиксированной лексикой. В этом случае при индексировании текстов получается некоторый набор стемов, которые и заносят в индекс. Однако подобный подход не лишен недостатков. Первым из них является невысокая точность метода. Еще одним недостатком является отсутствие возможности синтеза на морфологической базе без словаря основ.

Следует заметить, что грань между стемминговой морфологией, базирующейся на неизменяемой псевдооснове, и лексической морфологией, выдающей полный набор морфологических параметров и оперирующей с нормальными формами слова, довольно тонка. С одной стороны, лексическая морфология использует неизменяемую основу, т. е. стем. С другой стороны, при хранении полного набора лексической информации стемминг отличается от лемматизации лишь выдаваемой строкой нормальной формы.

Одним из современных вариантов реализации бессловарной морфологии являются аналитические

методы, не использующие словари [15]. Такие методы не решают все задачи морфологического анализа, так как трудно определить часть речи и грамматические признаки словоформы. Однако аналитические алгоритмы оказываются эффективны для задач индексации текстовых массивов, для создания процедур работы со словарями естественных языков.

В целях повышения эффективности решения задачи аналитического выделения основ на практике применяют смешанный подход, т. е. наряду с правилами преобразования аффиксов рассматривают таблицы, например, неправильных глаголов или исключений при образовании форм множественного числа имен существительных. Такие алгоритмы используют частоты *n*-грамм [16, 17], марковские цепи [18] и нейронные сети [19]. Универсальные методы выделения основ, не опирающиеся на грамматику естественного языка, требуют обработки больших объемов текстовой информации для обучения системы или в процессе работы самой информационной системы. Это обстоятельство не позволяет достигать высоких скоростей и качества обработки текстов, что делает их плохо пригодными к использованию в реальных системах.

Целью настоящей работы является полученная на основе анализа результатов экспериментальных исследований и эвристических соображений возможность оценки влияния методов "стемминг", "стоп-слова", "нижняя граница" и *tf-idf* на результат в разных модификациях этих алгоритмов при автоматической классификации текстовой информации. Все информационные коллекции при этом тестировались на наборах документов на русском языке [20] с использованием английских терминов и названий. Тестирование осуществлялось с помощью алгоритмов на характерных наборах данных. Используемые алгоритмы машинного обучения требуют для анализа специально подготовленной обучающей выборки, т. е. специально размеченных документов, в которых указаны слова, являющиеся самыми важными в данном тексте.

2.1. Мера *tf-idf*

В задачах анализа текстов, методах информационного поиска и технологии *tf-idf* для оценки важности слова в контексте документа, входящего в некоторый текстовый корпус, используется статистическая мера. Согласно определению, данная мера представляет собой произведение *tf*-меры (отношения числа вхождений слова к общему числу слов документа) и инверсии частоты встречаемости слова в документах корпуса (*idf*). Таким образом, *tf-idf* [21] — это статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса.

Вес некоторого слова пропорционален числу употребления этого слова в документе и обратно пропорционален частоте употребления слова в других документах коллекции [22]. Вес слова вычисляют по формуле

$$tf - idf\,(w,\, d,\, D) = tf\,(w,\, d) \times idf\,(W,\, D),$$

где *tf* (*term frequency* — частота терма) — отношение числа вхождения некоторого терма к общему количеству термов документа. Таким образом, оценивается важность терма w в пределах отдельного документа d [13], D — число документов в корпусе. Частота слова оценивает важность слова w_i в пределах отдельного документа:

$$tf\,(w,\, d) = \frac{n_i}{\Sigma_k n_k},$$

где n_i — число вхождений слова i в документ; $\Sigma_k n_k$ — общее число слов в данном документе; *idf* (*inverse document frequency* — обратная частота документа) — инверсия частоты, с которой некоторое слово встречается в документах коллекции. Учет *idf* уменьшает вес широкоупотребительных слов. Для каждого уникального слова в пределах конкретной коллекции документов существует только одно значение *idf* [22]:

$$idf\,(w,\, D) = \frac{|D|}{\left|(d_i \supset w_i)\right|},$$

где $|D|$ — число документов в корпусе; $\left|(d_i \supset w_i)\right|$ — число документов, в которых встречается слово w_i.

Когда *tf-idf*-функция применяется ко всем словам во всех документах корпуса, слова можно отсортировать по полученным весам. Более высокий вес *tf-idf* свидетельствует о том, что слово важно для данного документа, и в то же время достаточно редко употребляется в других документах корпуса. Этот факт зачастую можно интерпретировать как знак того, что слово является важным для данного конкретного документа и может быть использовано, чтобы более точно описать документ. Мера *tf-idf* предоставляет хорошую эвристику для определения кандидатов в ключевые слова, и этот метод (и многие его модификации) за годы исследований показал свою эффективность [9]. Данный метод в связи с эффективностью и простотой продолжает активно применяться и в настоящее время [23].

2.2. Метод "нижняя граница"

В качестве нижней границы при применении метода "нижняя граница" используется алгоритм, возвращающий первые 100 слов с начала текста. Это количество является оптимальным для средних размеров текста [24]. Данный алгоритм является весьма эффективным, так как требует наименьшего количества вычислительных ресурсов, не требует моделей и при этом его точность и полнота зачастую превышают соответствующие показатели сложных алгоритмов [24].

2.3. Метод "стоп-слова" ("шумовые слова")

Стоп-слова (шумовые слова) — это слова, являющиеся вспомогательными, которые несут малую смысловую нагрузку о содержании документа [25]. Для проведения экспериментальных исследований будет использоваться заранее составленный список по каждому корпусу текстов вручную. В процессе предварительной обработки шумовые слова удаляют из текста. К стоп-словам относят предлоги, союзы, местоимения и т. д.

2.4. Метод "стемминг"

Стеммингом является морфологический поиск [26], который заключается в преобразовании каждого слова и приведении его к общей форме. В этом подходе главным образом используется отбрасывание окончаний.

Это технология алгоритма морфологического разбора, учитывающая языковые особенности, вследствие чего данный алгоритм является языково-зависимым.

3. Метрики качества

Основным подходом к оценке качества использования методов автоматической классификации текстов является сравнение результатов их тестовых испытаний с теми оценками, которые дают этим методам эксперты. При этом "идеальным" алгоритмом считается тот, для которого выводы, сделанные по результатам тестирования, согласуются с мнением экспертов-оценщиков [27].

Основными показателями критерием при оценке качества, которое реализуется с использованием программной реализации метода (классификатора), являются точность и полнота. Точность (*precision*) классификации в пределах класса — это доля найденных классификатором документов, действительно принадлежащих данному классу, относительно всех документов, которые система отнесла к этому классу. Полнота (*recall*) классификации — это доля найденных классификатором документов, действительно принадлежащих классу, относительно всех документов этого класса в тестовой выборке [28].

Оценка качества работы классификатора проводится на тестовой выборке. Одновременно результаты его использования оценивает эксперт (табл. 1).

В табл. 1 приняты следующие условные обозначения:

TP — истинно положительное решение; *TN* — истинно отрицательное решение; *FP* — ложно положительное решение; *FN* — ложно отрицательное решение.

Теперь точность определяется следующим образом:

$$p = \frac{TP}{TP + FP}.$$

Полнота вычисляется по формуле

$$r = \frac{TP}{TP + FN}.$$

Таблица 1

Оценка качества методов уменьшения размерности текста классификации (выбранного слова)

Класс c_i		Экспертная оценка	
		Положительная	Отрицательная
Оценка системы	Положительная	*TP*	*FP*
	Отрицательная	*FN*	*TN*

Сравнение классификаторов при использовании в их составе различных методов является довольно сложной задачей по причине того, что разные входные данные могут приводить к различным результатам. Чтобы провести сравнение различных классификаторов, необходимо выполнить их построение и вычисление эффективности на одинаковых наборах документов для обучения и тестирования.

4. Экспериментальные исследования

Для апробации предложенных композиций методов был создан тестовый корпус, состоящий из 5000 текстовых документов, который описывает факты узкой предметной области.

В табл. 2 приведены дополняющие друг друга комбинации методов классификации текстов на основе их векторного представления.

В качестве исходных данных для приведения текста в векторный вид используются документы на русском языке. Документ подвергается обработке и анализу, в результате чего получаем вектор признаков, который используется для дальнейшей классификации.

В качестве примера приведем сравнение значимости слова по весу, используя работу [29]. Для сравнения в табл. 3 и 4 приведен вес 40 наиболее часто встречающихся слов в документе. Нулем в таблице обозначены слова, отсутствующие и не принимающие участия в создании вектора для определенной композиции.

Таблица 2

Композиции методов векторного представления текста

Сокращение	Композиция методов	Краткое описание
Т-I	*tf-idf*	Использование метода *tf-idf* для преобразования текста в вектор без какой-либо дополнительной обработки
Ш+Т-I	Шумовые слова + *tf-idf*	Удаление слов, не несущих смысловой нагрузки, и использование алгоритма *tf-idf* для приведения текста к векторному виду
С+Т-I	Стемминг + *tf-idf*	Приведение слов в тексте к единой основе и преобразование текста с помощью алгоритма *tf-idf*
С+Ш+Т-I	Стемминг + шумовые слова + *tf-idf*	Приведение слов в тексте к одинаковой основе, удаление шумовых слов, не несущих в себе смысловую нагрузку, и использование алгоритма *tf-idf*
НГ+Т-I	Нижняя граница + *tf-idf*	Используются сразу два алгоритма: «нижняя граница» и *tf-idf* в виде линейной комбинации
Ш+НГ+Т-I	Шумовые слова + нижняя граница + *tf-idf*	Удаление шумовых слов и использование сразу двух алгоритмов: «нижняя граница» и *tf-idf*
С+Ш+НГ+Т-I	Стемминг + шумовые слова + нижняя граница + *tf-idf*	Приведение слов в тексте к общей основе, затем удаление слов, которые самостоятельно не несут никакой смысловой нагрузки. Использование сразу двух алгоритмов: «нижняя граница» и *tf-idf*

Таблица 3

Изменение веса слов при использовании композиций

Слово	Композиция			
	Т-I	НГ+Т-I	Ш+Т-I	Ш+НГ+Т-I
цветом	0,03238	0	0,03238	0
для	0,00877	0	0	0
компьютерной	0,008994	0,090909	0,008994	0
или	0,006133	0	0	0
графики	0,012782	0,033333	0,012782	0
иллюстрации	0,017989	0,052632	0,017989	0,052632
что	0,003925	0	0	0
при	0,003998	0	0	0
слой	0,006746	0	0,006746	0
как	0,003257	0	0	0
изображения	0,004293	0	0,004293	0
графика	0,003445	0	0,003445	0,038462
вид	0,004906	0	0	0
the	0,053967	0	0	0
это	0,003855	0	0	0
изображение	0,011564	0	0,011564	0
текста	0,004261	0	0,004261	0
кисти	0,009524	0	0,009524	0
цвет	0,007359	0	0,007359	0
book	0,012265	0	0,012265	0
искусства	0,010378	0	0,010378	0,038462
смысле	0,006425	0	0,006425	0
его	0,003291	0	0	0
слоев	0,010378	0	0,010378	0
компьютерная	0,009812	0	0,009812	0,011111
книжной	0,00771	0	0,00771	0,015625
позволяет	0,004497	0	0,004497	0
иллюстрация	0,003175	0	0,003175	0
graphics	0,013492	0	0,013492	0
компьютерных	0,007196	0	0,007196	0
проблемы	0,008994	0,009091	0,008994	0
данной	0,004693	0,015152	0,004693	0
этом	0,003482	0,011111	0	0
методы	0,003084	0,045455	0,003084	0
этот	0,002249	0	0	0
процесс	0,004906	0,015152	0,004906	0
наброска	0,009812	0	0,009812	0
создается	0,00771	0	0,00771	0
заполнения	0,004906	0	0,004906	0
слоя	0,005681	0	0,005681	0

Таблица 4

Изменение веса слов при использовании композиций

Слово	Композиция		
	С+Т-I	С+Ш+Т-I	С+Ш+НГ+Т-I
цветом	0,08095	0,08095	0
для	0,00877	0	0
компьютерной	0,024285	0,024285	0
или	0,006746	0	0
графики	0,045446	0,045446	0
иллюстрации	0,058464	0,058464	0,052632
что	0,004906	0	0
при	0,003998	0	0
слой	0,021081	0,021081	0
как	0,003722	0	0
изображения	0,017171	0,017171	0
графика	0	0	0,038462
вид	0,008994	0,008994	0
the	0,062961	0	0
это	0,010279	0	0
изображение	0	0	0
текста	0,008521	0,008521	0
кисти	0,01746	0,01746	0
цвет	0	0	0
book	0,012265	0,012265	0
искусства	0,016605	0,016605	0,038462
смысле	0,006425	0,006425	0
его	0,003291	0	0
слоев	0	0	0
компьютерная	0	0	0,011111
книжной	0,038548	0,038548	0,015625
позволяет	0,00787	0,00787	0
иллюстрация	0,014285	0,014285	0
graphics	0,013492	0,013492	0
компьютерных	0	0	0
проблемы	0,013492	0,013492	0
данной	0,010559	0,010559	0
этом	0,008704	0	0
методы	0,004626	0,004626	0
этот	0	0	0
процесс	0,009812	0,009812	0
наброска	0,019624	0,019624	0
создается	0,023129	0,023129	0
заполнения	0,008586	0,008586	0
слоя	0	0	0

Таблица 5

Метрики качества методов уменьшения размерности вектора

Композиция	Точность, %	Полнота, %
Т-I	46	48
Ш+Т-I	75	73
С+Т-I	68	59
С+Ш+Т-I	96	90
НГ+Т-I	40	42
Ш+НГ+Т-I	45	38
С+Ш+НГ+Т-I	48	50

В зависимости от того, какая композиция будет использована, будет меняться и длина вектора, а также и значимость слова в документе (см. табл. 3, 4).

В табл. 5 приведена оценка качества работы композиций системы. При этом наилучшим алгоритмом считается тот, для которого выводы, сделанные системой, согласуются с мнением экспертов-оценщиков.

В результате проведения серии экспериментов и основываясь на метриках качества, можно прийти к перечисленным далее выводам.

- Установка "нижней границы" достаточно неоднозначна. Для нее сложно определить *n,* так как задача этого метода состоит в уменьшении размерности получаемого вектора. Однако при слишком большом уменьшении числа слов можно потерять смысловую нагрузку текста.
- Удаление "стоп-слов" всегда приводит к улучшению результатов работы алгоритмов.
- Использование "стемминга" также приводит к улучшению ситуации, так как при приведении к единой форме слов изменяется их вес и число документов, использующих данное слово.

Оптимальным будет сочетание методов "стемминг" + "стоп-слова" + *tf-idf*. Эта композиция позволит облегчить работу метода *tf-idf*, избавив текст от неинформативных слов и преобразовав слова к общей форме.

Заключение

В результате проведенных исследований найдена форма наиболее эффективной композиции методов предварительной обработки текстовых документов и приведения документов к векторному виду для их дальнейшего использования классификатором. Установлено, что необходима предварительная обработка текста для уменьшения размерности с условием сохранения смысловой нагрузки документа.

В ходе исследования выявлена рациональная композиция, использующая такие вспомогательные процедуры, как удаление стоп-слов из документов, стемминга, определения важности терма в корпусе документов с помощью характеристик *tf-idf*, работа которых позволит ускорить получение необходимого результата.

Определено направление дальнейшего продолжения исследований, направленных на улучшение векторизации текстовых документов, способствующих уменьшению размерности вектора и, вместе с тем сведению к минимуму потери важной информации.

Список литературы

1. **LeCun X. Z. Y.** Text understanding from scratch. Computer Science Department. Courant Institute of Mathematical Sciences, New York University. 2016. P. 10. URL: https://arxiv.org/pdf/1502.01710.pdf (дата обращения: 10.06.2018).
2. **Коваль С. А.** Лингвистические проблемы компьютерной морфологии. СПб.: Из-во СПбГУ, 2005. 152 с.
3. **Агеев М. С., Добров Б. В., Лукашевич Н. В.** Автоматическая рубрикация текстов: методы и проблемы // Ученые записки Казанского государственного университета. Серия Физико-математические науки. 2008. Том 150, Книга 4. С. 25—40.
4. **Поляков И. В., Соколова Т. В., Чеповский А. А., Чеповский А. М.** Проблема классификации текстов и дифференцирующие признаки // Вестник Новосибирского Государственного Университета. Серия: Информационные технологии. 2015. Том 13, № 2, С. 55—63.
5. **Yang Y.** An evaluation of statistical approaches to text categorization//Information Retrieval Jour. 1999. Vol. 1, Iss. 1, P. 69—90.
6. **Ju Ronghui, Zhou Pan, Hua Li Cheng, Liu Lijun.** An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis // 2015 IEEE Intern. Conf. on Comp. and Inform. Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing. Liverpool, UK, 2015. P. 2276—2283.
7. **Араева Л. А.** (Общ. ред.). Актуальные проблемы современного словообразования. Сборник научных статей. Вып. 4. Кемерово, 2011. 535 с.
8. **Nolasco D., Oliveira J.** Detecting Knowledge Innovation through Automatic Topic Labeling on Scholar Data // 49th Hawaii International Conference on System Sciences (HICSS). Koloa, HI: IEEE Computer Society, 2016. P. 358—367.
9. **Joachims T.** A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization // Joachims_97a. 1997. P. 9 URL: http://www.cs.cornell.edu/People/tj/publications/joachims_97a.pdf (дата обращения: 10.06.2018).
10. **Загорулько Ю. А., Кононенко И. С., Костов Ю. В., Сидорова Е. А.** Классификация деловых писем в системе документооборота // Материалы международной научно-технической конференции "Информационные системы и технологии" (ИСТ'2003). Новосибирск: Издательство НГТУ, 2003. Т. 3. С. 141—145.
11. **Некрестьянов И. С., Добрынин В. Ю., Клюев В. В.** Оценка тематического подобия текстовых документов // Труды второй всероссийской научной конференции "Электронные библиотеки". Протвино, Россия, сентябрь 2000. С. 204—210.
12. **Baerman M., Brown D.** Understanding and Measuring Morphological Complexity. Oxford University Press, 2015. 238 p.
13. **Болховитянов А. В., Чеповский А. М.** Алгоритмы морфологического анализа компьютерной лингвистики: учеб. пособие. М.: МГУП имени Ивана Федорова, 2013. 198 с.
14. **Николаев И. С., Митренина О. В., Ландо Т. М.** (ред.) Прикладная и компьютерная лингвистика. М.: Ленанд, 2016. 316 с.
15. **Большакова Е. И., Воронцов К. В., Ефремова Н. Э.** и др. Автоматическая обработка текстов на естественном языке и анализ данных: учеб. пособие. М.: Изд-во НИУ ВШЭ, 2017. 269 с.
16. **Sidorov G., Velasquez F., Stamatatos E., Gelbukh A., Chanona-Hernández L.** Syntactic Dependency-based N-grams as Classification Features // LNAI 7630. 2012. P. 1—11. http://www.cic.ipn.mx/~sidorov/sn_grams_MICAI2012.pdf (дата обращения: 10.06.2018).
17. **Sidorov G.** Syntactic Dependency Based N-grams in Rule Based Automatic English as Second Language Grammar Correction // International Journal of Computational Linguistics and Applications. 2013. Vol. 4, No. 2. P. 169—188.
18. **Кельберт М. Я., Сухов Ю. М.** Вероятность и статистика в примерах и задачах. Том 2. Марковские цепи как отправная точка теории случайных процессов и их приложения М.: МЦНМО, 2017. 43 с.
19. **Горожанина Е. И.** Нейронные сети: учеб. пособие. Поволж. гос. ун-т телекоммуникаций и информатики. Самара: Изд-во ПГУТИ, 2017. 84 с.
20. **Тузов В. А.** Компьютерная семантика русского языка. СПб.: Изд-во СПбГУ, 2004. 400 с.

21. **Jones K. S.** A statistical interpretation of term specificity and its application in retrieval // Journal of Documentation. 1972. Vol. 28, No. 1. P. 11—21.
22. **Некрестьянов И. С., Кураленок И. Е.** Оценка систем текстового поиска// Программирование. 2002. Том 28, № 4. С. 226—242.
23. **Недильченко О. С.** Этапы и методы автоматического извлечения ключевых слов // Молодой ученый. 2017. № 22. С. 60.
24. **Воронцов К. В.** Лекции по машинному обучению. URL: http://www.machinelearning.ru (дата обращения: 02.09.2018).
25. **Гращенко Л. А.** О модельном стоп-словаре // Известия Академии наук Республики Таджикистан. Отделение физико-математических, химических, геологических и технических наук. 2013. № 1 (150). С. 40—46.
26. **Губин М. В., Морозов А. Б.** Влияние морфологического анализа на качество информационного поиска. Консорциум "Кодекс". 2006. С. 16.
27. **Агеев М. С., Кураленок И. Е.** Официальные метрики РОМИП'2004 // Российский семинар по Оценке Методов Информационного Поиска (РОМИП 2004) Пущино, 2004. С. 142—150.
28. **Жизняков А. Л., Зуев В. В.** Влияние изменения размерности вектор-контуров изображений на их меру близости // Вопросы радиоэлектроники. 2010. Том 1, № 1. С. 165—170.
29. **Макарова И. О.** Компьютерная графика в книжной иллюстрации // Вестник Адыгейского государственного университета. Серия 2: Филология и искусствоведение 2011. С. 5. URL: https://cyberleninka.ru/article/n/kompyuternaya-grafika-v-knizhnoy-illyustratsii.pdf (дата обращения: 02.09.2018).

Analysis of Methods for Converting Texts into the Form of Objects in a Vector Space

E. I. Burlaeva, ekaterina0853@mail.ru, **V. N. Pavlysh,** pavlyshvn@mail.ru, Donetsk National Technical University, Donetsk, 83008, Donetsk region

Corresponding author:

Burlayeva Ekaterina I., Graduate Student, Donetsk National Technical University, Donetsk, 83008, Donetsk region,
E-mail: ekaterina0853@mail.ru

Received on June 26, 2018
Accepted on September 05, 2018

In the modern world, the amount of information is constantly growing. A large part of it is unstructured text data. It is difficult for a person to independently process them. Moreover, manual analysis is ineffective for large volumes of text, since it is limited by speed, errors and errors due to the human factor. Therefore, methods that can automatically handle such data are required. One of the technologies for processing textual information is the automatic classification of text documents. The traditional representation of a document in the form of a sequence of symbols makes it difficult to work with it as an object of classification. Most machine learning algorithms work with documents as elements of a vector space, which determines the need for a corresponding transformation of texts into a vector object form.

In this paper, we consider the possibilities of reducing the dimension of vectors for searching in the textual body of descriptions of close fragments of knowledge and linguistic forms of their expression.

Some of the methods that reduce the dimensionality of the vector for automatic classification of the text are considered, their advantages and disadvantages are highlighted, which can serve as a starting point for the development of more effective approaches. So, to reduce the dimension of vectors, combinations of text preprocessors.

Keywords: *vector representation, text document, word, composition of methods, tf-idf, classification, "Stemming", "Stop words", "Lower boundary"*

For citation:

Burlaeva E. I., Pavlysh V. N. Analysis of Methods for Converting Texts into the Form of Objects in a Vector Space*, Programmnaya Ingeneria,* 2019, vol. 10, no. 1, pp. 30—37

DOI: 10.17587/prin.10.30-37

References

1. **LeCun X. Z. Y.** *Text understanding from scratch,* Computer Science Department, Computer Science Department. Courant Institute of Mathematical Sciences, New York University. 2016. 10 p., available at: https://arxiv.org/pdf/1502.01710.pdf (date of access 10.06.2018).
2. **Koval' S. A.** *Lingvisticheskie problemy komp'juternoj morfologii* (Linguistic problems of computer morphology), Saint Petersburg, Izdatel'stvo SPbGU, 2005. 152 p. (in Russian).
3. **Ageev M. S., Dobrov B. V., Lukashevich N. V.** Avtomaticheskaja rubrikacija tekstov: metody i problem (Automatic rubrication of texts: methods and problems), *Uchenye zapiski Kazanskogo gosudarstvennogo universiteta. Serija Fiziko-matematicheskie nauki,* 2008, vol. 150, book 4, pp. 25—40 (in Russian).
4. **Poljakov I. V., Sokolova T. V., Chepovskij A. A., Chepovskij A. M.** Problema klassifikacii tekstov i differencirujushhie priznaki (Text classification problem and features set.), *Vestn. NGU. Ser.: Informacionnye tehnologii,* 2015, vol. 13, no. 2, pp. 55—63 (in Russian).

5. **Yang Y.** An evaluation of statistical approaches to text categorization, *Information Retrieval, Jour.,* 1999, vol. 1, iss. 1, pp. 69—90.

6. **Ju Ronghui, Zhou Pan, Hua Li Cheng, Liu Lijun.** An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis, *2015 IEEE Intern. Conf. on Comp. and Inform. Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing,* Liverpool, UK, 2015, pp. 2276—2283.

7. **Araeva L. A. (Eds.).** *Aktual'nye problemy sovremennogo slovoobrazovanija* (Actual problems of modern word formation), Sbornik nauchnyh statej, issue 4, Kemerovo, 2011, 535 p. (in Russian).

8. **Nolasco D., Oliveira J.** Detecting Knowledge Innovation through Automatic Topic Labeling on Scholar Data, *49th Hawaii International Conference on System Sciences (HICSS).* Koloa, HI: IEEE Computer Society, 2016, pp. 358—367.

9. **Joachims T.** A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, Joachims_97a, 1997, 9 p., available at: http://www.cs.cornell.edu/People/tj/publications/joachims_97a.pdf (date of access 10.06.2018).

10. **Zagorul'ko Ju. A., Kononenko I. S., Kostov Ju. V., Sidorova E. A.** Klassifikacija delovyh pisem v sisteme dokumentooborota (Classification of business letters in the workflow system), *Materialy mezhdunarodnoj nauchno-tehnicheskoj konferencii "Informacionnye sistemy i tehnologii" (IST'2003),* Novosibirsk, Izdatel'stvo NGTU, 2003, vol. 3, pp. 141—145 (in Russian).

11. **Nekrest'janov I. S., Dobrynin V. Ju., Kljuev V. V.** Ocenka tematicheskogo podobija tekstovyh dokumentov (Evaluation of thematic similarity of text documents.), *Trudy vtoroj vserossijskoj nauchnoj konferencii "Jelektronnye biblioteki",* Protvino, 2000, pp. 204—210 (in Russian).

12. **Baerman M., Brown D.** *Understanding and Measuring Morphological Complexity*, Oxford University Press, 2015, 238 p.

13. **Bolhovitjanov A. V., Chepovskij A. M.** *Algoritmy morfologicheskogo analiza komp'juternoj lingvistiki* (Algorithms for morphological analysis of computational linguistics), ucheb. posobie, Moscow, MGUP imeni Ivana Fedorova, 2013, 198 p. (in Russian).

14. **Nikolaev I. S., Mitrenina O. V., Lando T. M.** (Eds.) *Prikladnaja i komp'juternaja lingvistika* (Applied and computational linguistics), Moscow, Lenand, 2016, 316 p. (in Russian).

15. **Bol'shakova E. I., Voroncov K. V., Efremova N. Je.** et al. *Avtomaticheskaja obrabotka tekstov na estestvennom jazyke i analiz dannyh* (Automatic processing of natural language texts and data analysis), uchebnoe posobie, Moscow, Izd-vo NIU VShJe, 2017, 269 p. (in Russian).

16. **Sidorov G., Velasquez F., Stamatatos E., Gelbukh A., Chanona-Hernández L.** Syntactic Dependency-based N-grams as Classification Features, *LNAI 7630,* 2012, pp. 1—11, available at: http://www.cic.ipn.mx/~sidorov/sn_grams_MICAI2012.pdf (date of access 10.06.2018).

17. **Sidorov G.** Syntactic Dependency Based N-grams in Rule Based Automatic English as Second Language Grammar Correction, *International Journal of Computational Linguistics and Applications,* 2013, vol. 4, no. 2, pp. 169—188.

18. **Kel'bert M. Ja., Suhov Ju. M.** *Verojatnost' i statistika v primerah i zadachah, Tom 2. Markovskie cepi kak otpravnaja tochka teorii sluchajnyh processov i ih prilozhenija* (Probability and statistics in examples and problems), Moscow, Izdatel'stvo MCNMO, 2017, 43 p. (in Russian).

19. **Gorozhanina E. I.** *Nejronnye seti* (Neural networks), ucheb. posobie, Povolzh. gos. un-t telekommunikacij i informatiki, Samara, Izd-vo PGUTI, 2017. 84 p. (in Russian).

20. **Tuzov V. A.** *Komp'juternaja semantika russkogo jazyka* (Computer semantics of the Russian language), Saint Petersburg, Izd-vo SPbGU, 2004, 400 p. (in Russian).

21. **Jones K. S.** A statistical interpretation of term specificity and its application in retrieval*, Journal of Documentation,* 1972, vol. 28, no. 1, pp. 11—21.

22. **Nekrest'janov I. S., Kuralenok I. E.** Ocenka sistem tekstovogo poiska (Evaluation of text search systems.), *Programmirovanie*, 2002, vol. 28, no. 4, pp. 226—242 (in Russian).

23. **Nedil'chenko O. S.** Jetapy i metody avtomaticheskogo izvlechenija kljuchevyh slov (Steps and methods for automatic extraction of keywords), *Molodoj uchenyj*, 2017, no. 22, p. 60. (in Russian).

24. **Voroncov K. V.** Lekcii po mashinnomu obucheniju (Lectures on machine learning), available at: http://www.machinelearning.ru (date of access 02.09.2018) (in Russian).

25. **Grashhenko L. A.** O model'nom stop-slovare (About the model stop dictionary), *Izvestija Akademii nauk Respubliki Tadzhikistan. Otdelenie fiziko-matematicheskih, himicheskih, geologicheskih i tehnicheskih nauk,* 2013, no. 1 (150), pp. 40—46. (in Russian).

26. **Gubin M. V., Morozov A. B.** Vlijanie morfologicheskogo analiza na kachestvo informacionnogo poiska (Impact of morphological analysis on the quality of information retrieval), Konsorcium "Kodeks", 2006, 16 p. (in Russian).

27. **Ageev M. S., Kuralenok I. E.** Oficial'nye metriki ROMIP'2004 (Official metrics ROMIP 2004), *Rossijskij seminar po ocenke metodov informacionnogo poiska (ROMIP 2004)*, Pushhino, 2004, pp. 142—150 (in Russian).

28. **Zhiznjakov A. L., Zuev V. V.** Vlijanie izmenenija razmernosti vektor-konturov izobrazhenij na ih meru blizosti (The effect of changing vector dimensions — the contours of images on their measure of proximity), *Voprosy radiojelektroniki,* 2010, vol. 1, no. 1, pp. 165—170 (in Russian).

29. **Makarova I. O.** Komp'juternaja grafika v knizhnoj illjustracii (Computer graphics in the book illustration), *Vestnik Adygejskogo gosudarstvennogo universiteta. Serija 2: Filologija i iskusstvovedenie*, 2011, p. 5, available at: https://cyberleninka.ru/article/n/kompyuternaya-grafika-v-knizhnoy-illyustratsii.pdf (date of access 02.09.2018) (in Russian).

УДК 81'32+004.8 DOI: 10.17587/prin.10.38-46

М. В. Плетнёва, ассистент, e-mail: pletneva.mv.kirov@gmail.com, Вятский государственный университет, г. Киров

Программная система анализа тональности текстов на основе словарей оценочной лексики

Рассмотрена задача автоматического анализа текстов в целях определения их тональности. Предложена структура программной системы анализа тональности текстов, представлен состав ее подсистем и их назначение. С помощью диаграммы классов показан вариант программной реализации системы. Эксперименты, проведенные на основе общедоступных текстовых корпусов, подтверждают эффективность разработанной системы.

Ключевые слова: *обработка естественного языка, извлечение мнений, автоматический анализ тональности, словарь оценочной лексики, программная система, диаграмма классов*

Введение

Многие текстовые документы, размещаемые пользователями в сети Интернет, являются источником субъективной информации, т. е. содержат авторскую эмоциональную оценку объекта высказывания. Например, в отзывах о фильмах это могут быть следующие фразы: "хороший, отлично поставленный фильм, но максимально предсказуемый", "шедевр, сокровище всех времен", "скукота, только зря потрачено время". Эмоциональное отношение автора, выраженное в тексте, называют *тональностью* текста [1]. Как правило, тональность представляется в виде значения на одномерной шкале, которое называется *классом тональности*. В зависимости от количества значений выделяют бинарную шкалу (два класса тональности — позитивный и негативный) [2], тернарную шкалу (три класса тональности — позитивный, негативный, нейтральный) [3] и многозначную шкалу (более трех классов тональности) [4]. Автоматическое определение тональности текстов называют *анализом тональности* (*sentiment analysis*) или *классификацией по тональности* (*sentiment classification*) — в настоящее время эти термины взаимозаменяемы. Качественное решение проблемы анализа тональности позволит вывести на новый уровень исследования в области автоматического понимания текста и существенно расширить возможности систем искусственного интеллекта. В качестве примеров практического применения можно указать маркетинговые исследования сайтов отзывов в интересах поставщиков и потребителей товаров и услуг, анализ мнений избирателей в социальных сетях о кандидатах на выборах, определение тональности ответов пользователей в человеко-машинных интерфейсах для повышения качества диалога.

Задачу анализа тональности можно представить как частный случай проблемы текстовой категоризации [5]: дано множество текстовых документов $D = \{d_1,..., d_{|D|}\}$ и множество классов тональности $S = \{s_1,..., s_{|S|}\}$. Требуется построить такую функцию (классификатор) Ф, которая каждой паре $(d_i, e_j) \in D \times E$ присваивает булево значение T или F ("истина" и "ложь" соответственно):

$$\Phi: D \times E \rightarrow \{T, F\}. \quad (1)$$

При этом значение T, назначенное паре (d_i, e_j), свидетельствует о том, что в документе d_i выражена тональность e_j; значение F указывает, что данная тональность отсутствует.

Следует отметить, что задача анализа тональности осложняется сильной зависимостью выражений тональности от контекста и предметной области, а также наличием сарказма и иронии.

Проблема автоматического анализа тональности активно исследуется с начала 2000-х годов. Для решения данной задачи использовались следующие основные подходы [6]: подход на основе знаний; машинное обучение с учителем и без учителя; комбинированный подход.

В *подходе на основе знаний* анализ тональности текстов осуществляется с помощью правил, заданных вручную экспертами. Существуют два основных вида правил, основанных на грамматике и на фрагментах. Грамматические правила подразумевают применение контекстно-свободных грамматик, как это реализовано, например, в анализаторе Томита [7]. В другом виде правил используются специальные фрагменты, на которые разбивается текст, а правила описывают операции, которые проводятся над этими фрагментами [8]. Составление правил является достаточно трудоемкой задачей, однако, несмотря на это, данный подход реализован во многих работах [9—12].

Впервые для задачи анализа тональности текста подход на основе *машинного обучения* с *учителем* был применен в исследовании, результаты которого представлены в работе [13]. Данный подход предпо-

лагает наличие корпуса размеченных документов, при этом эффективность анализа тональности напрямую зависит от качества корпуса. Наиболее часто используются такие методы машинного обучения, как метод опорных векторов (*support vector machines*, SVM) [3, 4, 13], метод максимизации энтропии (*maximum entropy method*) [13—15], комитеты классификаторов (*AdaBoost*) [16, 17], наивный байесовский классификатор (*Naïve Bayes classifier*) [13—15, 18]. Метод опорных векторов в большинстве экспериментов показывал лучшие результаты по сравнению с другими методами.

При *обучении без учителя* не требуется наличия обучающего корпуса. Первым исследованием по анализу тональности с использованием обучения без учителя была работа P. Turney [19]. В ней для определения тональности в тексте сначала выделяли синтаксические шаблоны, а затем для каждого слова в шаблоне оценивали близость к словам "excellent" и "poor" на основе результатов поисковой системы. После этого каждый шаблон получал вес, равный разности оценок близости к "excellent" и "poor". Вычисляли средний вес всех шаблонов и вывод о тональности текста делали на основании знака среднего веса. Методы с использованием поисковых систем применяли также в работах [20, 21].

Другие исследования в рамках обучения без учителя использовали различные лингвистические ресурсы — словари или тезаурусы [2, 22, 23]. В работе M. Taboada и др. [23] использовались словари, в которых для каждого слова указана семантическая ориентация (полярность и сила тональности). На основе встречаемых в тексте слов из словаря, слов усиления и отрицания осуществлялся вывод о тональности текста.

В работах [10, 24, 25] применяли *комбинированный подход*, сочетающий классификаторы, построенные на основе разных принципов. Например, в работе [10] на основе обучающей коллекции данных автоматически выделялись текстовые шаблоны, которые представляли собой упорядоченную последовательность элементов словаря. В дальнейшем они предоставлялись эксперту, который отбирал среди них шаблоны, содержащие достаточную информацию для определения тональности документа на основе самого шаблона. Строился двухэтапный классификатор следующим образом: сначала проверяли, соответствует ли неклассифицированный документ любому из выбранных шаблонов. В случае соответствия документу присваивали тональность на основе шаблона. В противном случае тональность назначалась классификатором на основе метода машинного обучения, который был обучен на тех же данных, на основе которых выделялись шаблоны. В работе [25] последовательно применяли классификаторы на основе правил и машинного обучения.

В настоящей работе предлагается структура системы анализа тональности текстов на основе словарей оценочной лексики и вариант ее программной реализации. Система использует комбинированный подход, сочетающий создание словаря оценочной лексики при участии эксперта и определение тональности текстов на основе методов машинного обучения с учителем. Методы, реализованные в системе и описанные в работах [26, 27], позволяют:

- формировать словарь оценочной лексики для заданной предметной области;
- настраивать оптимальные параметры для полученного словаря;
- автоматически определять тональность текста.

При минимальном объеме экспертных знаний система позволяет на высоком уровне выявлять мнения, что доказано рядом экспериментов на общедоступных русскоязычных корпусах документов [28, 29].

Общая структура системы

Система анализа тональности текстов на основе словарей оценочной лексики состоит из нескольких подсистем, схема взаимодействия которых показана на рис. 1. Массив текстовых документов на входе системы разделен на **обучающий** и **контрольный** корпуса. Обучающий корпус используется как для создания словарей оценочной лексики, так и для классификации текстов по тональности. Контрольный корпус применяется для оценки качества классификации текстов по тональности.

Все тексты, с которыми работает система, подвергаются предобработке в **подсистеме первичного анализа текста**. Сначала тексты поступают в блок графематического анализа и преобразуются из исходного вида в формат, принятый в системе: текст очищают от знаков форматирования, выделяются отдельные слова, все слова приводят к единому регистру. В следующем блоке осуществляется морфологический анализ [30], в результате которого документ представляется в виде списка слов в нормальной форме с грамматическими характеристиками (частями речи). Кроме того, удаляются слова, не несущие смысловую нагрузку (стоп-слова), и в зависимости от текущих значений параметров системы удаляются неиспользуемые части речи. Наиболее вероятными оценочными словами являются прилагательные, существительные, глаголы, наречия и междометия, поскольку слова этих частей речи несут наибольшую эмоциональную нагрузку [31]. Таким образом, результатом работы подсистемы первичного анализа является представление каждого документа в виде списка терминов — наиболее значимых слов.

Подсистема формирования словаря реализует функции построения словаря оценочной лексики. Она действует на основе подготовленного в предыдущей подсистеме корпуса обучающих документов, а результатом ее работы является словарь оценочной лексики, включающий слова позитивной и негативной тональностей. Например, для отзывов о фильмах позитивными могут быть следующие оценочные слова: *захватывающий*, *отличный*, *супер*, *нравиться*, *обожать*; а негативными — *скучно*, *ерунда*, *нудно*, *спать*, *ужасный*. Подсистема основана на алгоритме форми-

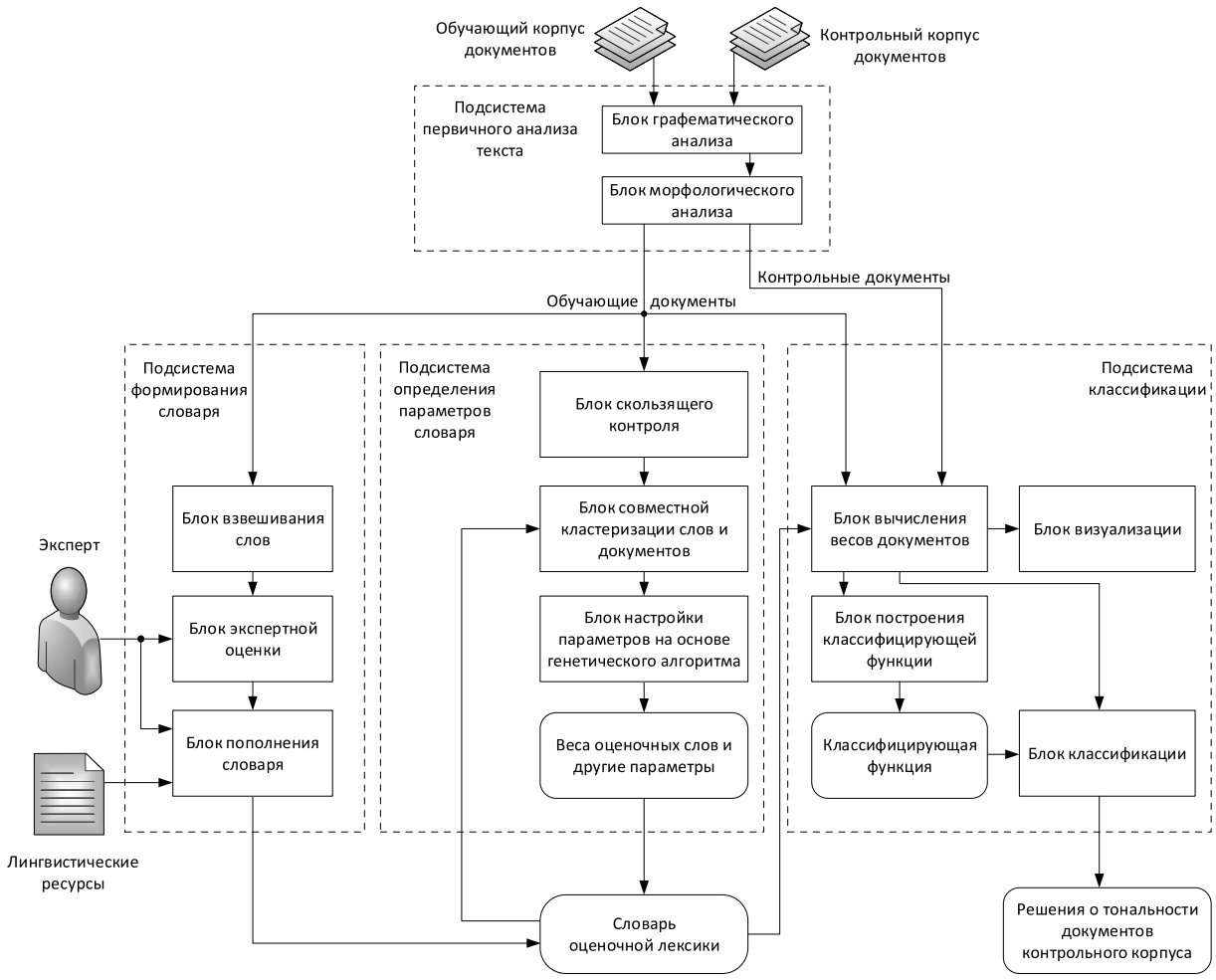


Рис. 1. Структура системы анализа тональности текстов

рования словаря оценочной лексики [26], в котором сначала вычисляются веса всех слов обучающего корпуса отдельно для позитивного и негативного классов тональности на основе модифицированной релевантной частоты [32]. Далее эксперты отбирают из обоих списков оценочные слова, имеющие наиболее яркую эмоциональную окраску. К отобранным словам добавляются отрицание (частица *не*) и модификаторы (*сильно*, *очень*, *совсем* и др.) под контролем эксперта. Отрицания меняют эмоциональную оценку следующего за ним оценочного слова на противоположную. Так, в словосочетании "***не*** *нравится фильм*" оценочное слово *нравится* имеет позитивную тональность, частица ***не*** меняет тональность на негативную. Модификаторы усиливают или ослабляют эмоциональную оценку следующего за ними оценочного слова. Например, в словосочетании "***очень*** *хороший*" модификатор ***очень*** усиливает позитивную оценку оценочного слова *хороший*.

Каждое слово, входящее в сформированный словарь, имеет весовой коэффициент, характеризующий интенсивность выражения тональности: чем больше значение коэффициента, тем сильнее данное слово выражает тональность. Значения данных коэффициентов для оценочных слов, отрицаний и модификаторов, оптимальные для решения задачи анализа тональности, вычисляются в **подсистеме определения параметров словаря** на основе метода, предложенного в работе [26]. В данном методе задача вычисления оптимальных весов оценочных слов рассматривается как задача оптимизации многомерной функции, в которой в качестве аргументов выступают искомые веса, а значение функции совпадает со значением выбранной метрики качества классификации.

Для оценки качества анализа тональности может служить одна из следующих метрик: точность (*precision*), полнота (*recall*), F_1-мера (F_1-*measure*) или правильность (*accuracy*) [5]. Выбор конкретной метрики зависит от особенностей решаемой задачи, например, в случае сильной несбалансированности контрольного корпуса (число позитивных текстов значительно отличается от числа негативных) рекомендуется выбирать F_1-меру. Такая мера является гармоническим средним двух важных показателей эффективности системы — точности и полноты. Оптимальными весами оценочных слов являются

такие значения, при которых достигается максимум выбранной метрики качества с учетом используемых контрольного корпуса и метода классификации.

Оптимальные веса оценочных слов находят на основе генетического алгоритма. Для сокращения пространства поиска оптимальных весов оценочных слов и, соответственно, времени поиска, используют алгоритм совместной кластеризации слов и документов [33]. В нем осуществляется кластеризация документов на основании распределения в них слов и также кластеризация слов, определяемая их совместной встречаемостью в документах. В результате формируются кластеры, каждый из которых включает в себя обучающие документы и слова из словаря оценочной лексики. В дальнейшем каждый кластер обрабатывается независимо, а именно — находятся оптимальные веса оценочных слов на основе генетического алгоритма с использованием только тех документов, которые принадлежат данному кластеру. Для исключения эффекта переобучения в подсистеме используется процедура скользящего контроля по N блокам (*N-fold cross-validation*) [23], где N, как правило, принимает значения 5 или 10.

Таким образом, совместное применение генетического алгоритма и совместной кластеризации слов и документов позволяет выявить компактный и эффективно взвешенный словарь оценочной лексики.

Взвешенный словарь, а также корпус обучающих документов используются **подсистемой классификации** для автоматического определения тональности контрольных документов. Работа подсистемы включает два основных этапа — этап обучения и этап классификации.

На этапе обучения вычисляются веса текстовых документов из обучающего корпуса для позитивной и негативной тональностей с использованием словаря оценочной лексики по формуле

$$W_d^s = \frac{k_s \sum_{i=1}^{N_s} W_i^s}{N_s}, \qquad (2)$$

где W_d^s — вес документа d для тональности s; W_i^s — вес оценочного слова i, относящегося к тональности s; N_s — число оценочных слов тональности s в документе d. В связи с преобладанием в текстах позитивной лексики вводится коэффициент k_s для негативных текстов, который позволяет компенсировать данный эффект. Указанный коэффициент определяется автоматически в подсистеме определения параметров словаря. Для позитивных текстов k_s равно единице.

Далее обучающие тексты помещают в двумерное "эмоциональное" пространство (позитивная тональность — негативная тональность) в соответствии с вычисленными весами. Наглядное представление текстов на двумерной плоскости позволяет отслеживать и при необходимости корректировать процесс работы классификатора. На основе обучающих текстов на двумерной плоскости строится бинарная классифицирующая функция. Для построения такой функции используется один из наиболее мощных методов машинного обучения — метод опорных векторов (SVM) [13]. В качестве реализации метода опорных векторов используется библиотека LIBSVM [34]. В ходе исследования проводился выбор ядра и подбор оптимальных параметров. Наилучшие результаты показало использование линейного ядра с параметром регуляризации $C = 1$.

На этапе классификации вычисляются веса контрольных документов по формуле (2), так же как и на этапе обучения. Затем контрольные документы помещают в двумерное эмоциональное пространство в соответствии с полученными весами. Классификация контрольных документов осуществляется на основе классифицирующей функции. В зависимости от знака функции для контрольного документа принимается решение об его отнесении к одному из классов тональности. Следует отметить, что в случае необходимости повышения производительности оба этапа работы подсистемы классификации легко распараллеливаются, поскольку зависимости по данным отсутствуют.

В случае наличия готового взвешенного словаря оценочной лексики в системе используются только две подсистемы — первичного анализа текста и классификации.

Программная реализация системы

Проектирование программной системы выполнено на основе принципов объектно-ориентированного программирования с помощью языка моделирования UML [35]. Основные классы системы представлены на рис. 2. Некоторые служебные и вспомогательные классы, например, класс *Preprocessor*, реализующий подсистему первичного анализа текстов, не представлены на диаграмме. Для простоты оставлены лишь ключевые методы классов, сигнатуры методов опущены.

Между классами можно выделить два основных вида отношений.

Первый из них — ассоциативный (⟵), он отражает направленную связь. Например, класс, **GeneticAlgorithm** использует класс **Cluster** в качестве параметра операции.

Второй вид — агрегирующий (◇—), он представляет отношение "целое—часть". Например, класс **Document** включает в себя (агрегирует) класс **Word**.

Функции подсистемы первоначального анализа текстов выполняют служебные классы, не отраженные в данной схеме. В результате их работы формируются текстовые корпуса данных, для хранения которых используется структура **DocCollection**. Обучающий корпус и контрольный различаются с помощью параметра **TypeCollection**. После создания (метод Create) данные корпуса можно сохранить (метод Save), а при следующем использовании — загрузить из файла (метод Load).

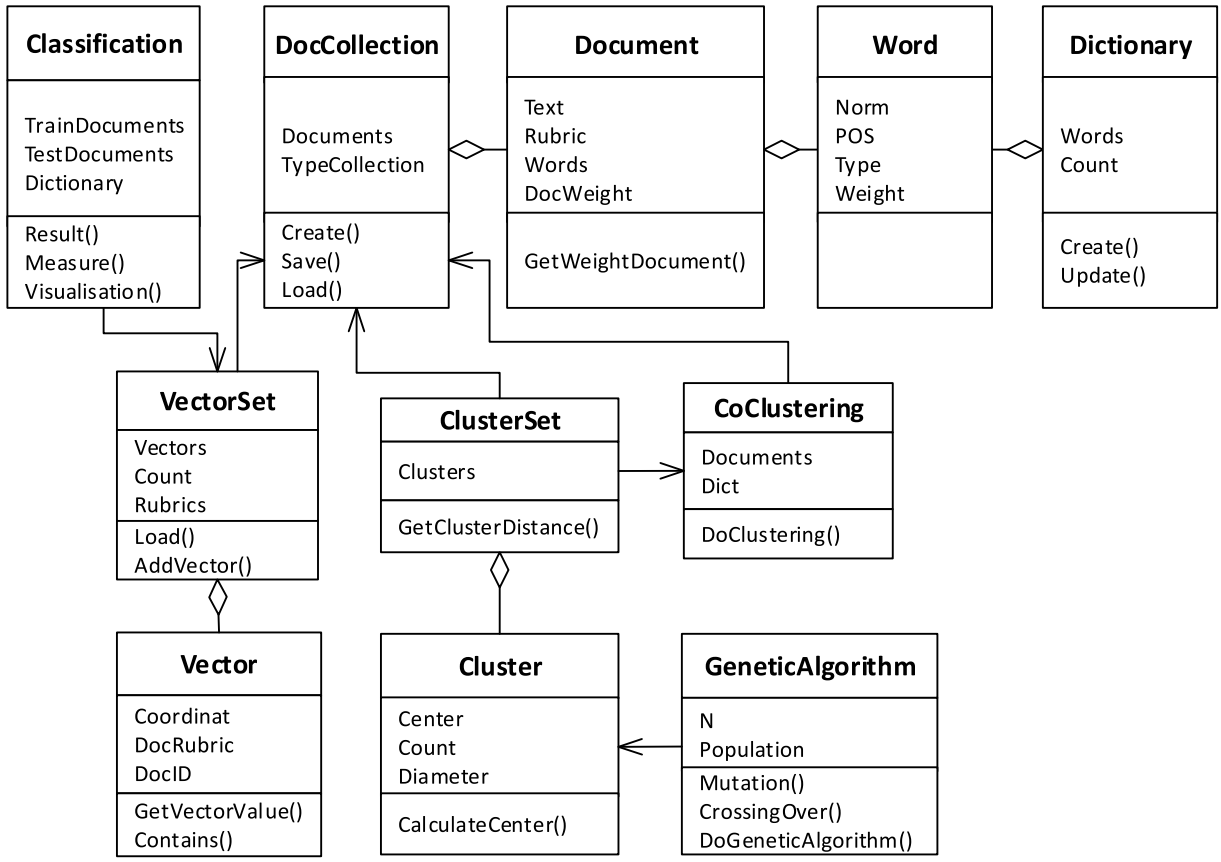


Рис. 2. Диаграмма основных классов системы

При формировании корпуса отдельно обрабатывается каждый документ, для их хранения используется структура **Document**. Для каждого документа определяются следующие свойства: Rubric — класс тональности, к которому относится документ; Text — содержимое документа, которое разбивается на отдельные слова Words; DocWeight — вес документа, вычисляемый в результате работы метода классификации.

Для каждого слова документа определяются его нормальная форма Norm, часть речи POS (*part of speech*), тип слова Type (оценочное, модификатор, отрицание), вес слова Weight, который вычисляется в результате работы подсистемы определения параметров словаря.

Словарь хранится в структуре данных **Dictionary** и формируется на основе обучающей коллекции документов. Функциональные возможности подсистемы формирования словаря реализуется в методе Create.

Реализация подсистемы определения параметров словаря выполнена с использованием двух основных классов — **GeneticAlgorithm** и **CoClustering**. Класс **CoClustering** управляет процессом совместной кластеризации слов из словаря оценочной лексики и документов обучающей коллекции. Сформированные в результате кластеры в свою очередь формируют набор кластеров **ClusterSet**. Кластеры располагаются в двумерном евклидовом пространстве (позитивная—негативная тональности). Каждый кластер **Cluster** при этом описывается центром — Center, числом элементов в кластере Count, диаметром — Diameter и собственно набором векторов, входящих в кластер. Для определения центра кластера используется метод CalculateCenter. Класс **GeneticAlgorithm** управляет процессом определения оптимальных параметров для каждого кластера, основанным на генетическом алгоритме.

Подсистема классификации реализуется в классе **Classification**. Здесь вычисляются веса документов для каждого класса тональности. Документы представляются на плоскости в соответствии со своим весом (метод Visualisation). Класс **Vector** описывает представление документа на координатной плоскости, где осями координат являются классы тональности. Класс **VectorSet** описывает представление корпуса документов. Также здесь вычисляются параметры, отвечающие за качество классификации в методе Measure.

Программная реализация рассмотренных классов выполнена в среде Microsoft Visual Studio 2015 на языке C#.

Экспериментальные результаты

Эффективность разработанной программной системы оценивали с использованием текстовых корпусов, предоставленных организаторами семи-

Таблица 1

Корпуса документов

Документы	Фильмы		Книги	
	Обучающий корпус	Контрольный корпус	Обучающий корпус	Контрольный корпус
Всего:	10 000	720	10000	395
Позитивная тональность	7347	594	7898	356
Негативная тональность	2653	126	2102	39

наров по оценке методов информационного поиска РОМИП-2011 [28] и РОМИП-2012 [29]. Указанные корпуса включают отзывы интернет-пользователей о фильмах и книгах (табл. 1). В качестве обучающего корпуса для системы были использованы случайные 10 000 отзывов для каждой предметной области корпуса РОМИП-2011, для получения метрик качества использованы контрольные корпуса РОМИП-2011 и РОМИП-2012. В отзывах пользователями дается оценка по десятибалльной шкале. Для классификации на два класса тональности оценки были преобразованы следующим образом: негативная тональность — оценка от 1 до 5, позитивная тональность — оценка от 6 до 10.

В разработанной системе для негативных текстов используется коэффициент k_s для компенсирования преобладания в корпусах позитивных текстов. Приводится сравнение результатов работы системы с учетом данного коэффициента и без него (табл. 2).

Результаты, полученные с использованием представленной в настоящей работе системы, сравнивали с результатами, полученными с помощью методов SVM, Naïve Bayes и AdaBoost, которые реализованы в библиотеке машинного обучения scikit-learn [36]. Для сравнения использовался также простейший классификатор (*baseline*), который классифицировал все объекты как позитивные. Для представления документов использовалась векторная модель [37], согласно которой каждый документ представлен своим вектором. Размерность векторов при этом совпадает с размером словаря, а каждое слово соответствует определенному элементу вектора. Значение элемента вектора характеризует интенсивность выражения тональности данным словом.

Вследствие сильного преобладания позитивных текстов в корпусах для оценки использовалась F_1-мера, для которой выполнялось макроусреднение [37].

В табл. 2 представлены результаты предложенной системы и методов машинного обучения при классификации текстов на два класса тональности.

Предложенная система показала лучший результат для обеих предметных областей. Превосходство над ближайшим лучшим методом для фильмов составило 5,6 % (SVM), для книг — 5,8 % (SVM).

На рис. 3 и 4 (см. четвертую сторону обложки) представлено распределение текстов на эмоциональной плоскости без учета и с учетом коэффициента k_s.

Таблица 2

Результаты экспериментов, F_1-мера, %

Метод	Книги	Фильмы
baseline	47,4	45,2
SVM	62,3	60,9
Naïve Bayes	58,6	55,2
AdaBoost	60,5	58,5
Предложенная система без учета k_s	62,7	61,4
Предложенная система с учетом k_s	**68,1**	**66,5**

На рис. 3 и 4 видно, что применение коэффициента k_s, используемого в системе при подсчете весовых значений для негативных текстов, позволило сгруппировать негативные документы, что улучшило результаты работы системы на 5,4 % для отзывов о книгах и 5,1 % для отзывов о фильмах.

Заключение

С учетом изложенного выше можно сделать следующие выводы. Разработанная автором и представленная в настоящей работе программная система демонстрирует высокое качество анализа тональности, превосходящее другие методы машинного обучения.

Другие преимущества предложенной системы заключаются в следующем:

- в системе имеется возможность формирования и автоматической настройки параметров словаря оценочной лексики в зависимости от предметной области;
- при наличии словаря оценочной лексики система обладает независимостью от обучающих данных;
- средства визуализации представления текстов на двумерной эмоциональной плоскости позволяют наглядно объяснять полученные результаты классификации.

Предлагаемая система может быть использована как самостоятельный инструментальный комплекс для резюмирования мнений по исследуемому объекту на основе корпуса документов. Возможна также интеграция основной части системы в поисковые

сервисы, ориентированные на подготовку рекомендаций по результатам анализа текстовых данных.

Работа выполнена при финансовой поддержке Министерства образования и науки РФ, государственное задание ВятГУ № 34.2092.2017/4.6, проект "Разработка и исследование словарей оценочной лексики для анализа тональности текстов" (2017—2019 гг.).

Список литературы

1. **Taboada M.** Sentiment Analysis: An Overview from Linguistics // Annual Review of Linguistics. 2016. No. 2. P. 325—347.

2. **Godbole N. Srinivasaiah M., Skiena S.** Large-Scale Sentiment Analysis for News and Blogs // Proceedings of the International Conference on Weblogs and Social Media (ICWSM), 2007. URL https://icwsm.org/papers/3-Godbole-Srinivasaiah-Skiena.pdf.

3. **Agarwal A., Xie B., Vovsha I., Rambow O., Passonneau R.** Sentiment Analysis of Twitter Data // Proceedings of the Workshop on Language in Social Media (LSM 2011). 2011. P. 30—38.

4. **Gamon M.** Sentiment classification on customer feedback data: Noisy data, large feature vectors, and the role of linguistic analysis // Proceedings of the International Conference on Computational Linguistics (COLING). 2004. P. 841—847.

5. **Sebastiani F.** Machine learning in Automated Text Categorization // ACM Computing Surveys. 2002. Vol. 34. P. 1—47.

6. **Liu B.** Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers. 2012. 168 p.

7. **Tomita** parser. URL: https://tech.yandex.ru/tomita/.

8. **Vasilyev V. G., Denisenko A. A., Solovyev D. A.** Aspect Extraction and Twitter Sentiment Classification by Fragment Rules // Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". 2015. No. 14 (21). P. 76—87.

9. **Пазельская А. Г., Соловьев А. Н.** Метод определения эмоций в текстах на русском языке // Компьютерная лингвистика и интеллектуальные технологии: по материалам ежегодной междунар. конф. "Диалог". Вып. 10 (17). М.: Изд-во РГГУ. 2011. С. 510—522.

10. **König A. C., Brill E.** Reducing the human overhead in text categorization // Proceedings of the 12th ACM SIGKDD conference on knowledge discovery and data mining. 2006. P. 598—603.

11. **Kuznetsova E. S., Loukachevitch N. V., Chetviorkin I. I.** Testing rules for a sentiment analysis system // Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". 2013. No. 12 (19). P. 71—80.

12. **Panicheva P. V.** ATEX: a rule-based sentiment analysis system processing texts in various topics // Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". 2013. No. 12 (19). P. 101—112.

13. **Pang B., Lee L., Vaithyanathan S.** Thumbs up? Sentiment Classification using Machine Learning Techniques // Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). 2002. P. 79—86.

14. **Boiy E., Hens P., Deschacht K., Moens M.-F.** Automatic Sentiment Analysis in On-line Text // Proceedings of Conference on Electronic Publishing (ELPUB'2007). 2007. P. 349—360.

15. **Go A., Bhayani R., Huang L.** Twitter Sentiment Classification using Distant Supervision // Association for Computational Linguistics. 2009. P. 30—38.

16. **Bhowmick P. K., Basu A., Mitra P.** Classifying Emotion in News Sentences: When Machine Classification Meets Human Classification // International Journal on Computer Science and Engineering. 2010. Vol. 2 (1). P. 98—108.

17. **Kouloumpis E., Wilson T., Moore J.** Twitter Sentiment Analysis: The Good the Bad and the OMG! // Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM—2011). 2011. P. 538—541.

18. **Tutubalina E. V., Zagulova M. A., Ivanov V. V., Malykh V. A.** A Supervised Approach for SentiRuEval Task on Sentiment Analysis of Tweets about Telecom and Financial Companies // Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". 2015. No. 14 (21). P. 65—75.

19. **Turney P.** Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews // Proceedings of the Association for Computational Linguistics (ACL). 2002. P. 417—424.

20. **Chaovalit P., Zhou L.** Movie review mining: A comparison between supervised and unsupervised classification approaches // Proceedings of the Hawaii International Conference on System Sciences (HICSS). 2005. P. 112c—112c.

21. **Kozareva Z., Navarro B., Vazquez S., Montoyo A.** Uazbsa: A headline emotion classification through web information. // Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval—2007), Prague, Czech Republic, Association for Computational Linguistics. 2007. P. 334—337.

22. **Котельников Е. В., Клековкина М. В.** Автоматический анализ тональности текстов на основе методов машинного обучения // Компьютерная лингвистика и интеллектуальные технологии: по материалам ежегодной междунар. конф. "Диалог". Вып. 11 (18). М.: Изд-во РГГУ, 2012. С. 753—762.

23. **Taboada M., Brooke J., Tofiloski M., Voll K., Stede M.** Lexicon-Based Methods for Sentiment Analysis // Computational Linguistics. 2011. Vol. 37 (2). P. 267—307.

24. **Котельников Е. В.** Комбинированный метод автоматического определения тональности текста // Программные продукты и системы. 2012. № 3. С. 189—195.

25. **Prabowo R., Thelwall M.** Sentiment analysis: A combined approach // Journal of Informetrics. 2009. Vol. 3, No. 2. P. 143—157.

26. **Котельников Е. В., Плетнева М. В**. Анализ тональности текстов на основе генетического алгоритма и совместной кластеризации слов и документов // Известия РАН. Теория и системы управления. 2016. № 1. С. 115—123.

27. **Клековкина М. В., Котельников Е. В.** Метод автоматической классификации текстов по тональности, основанный на словаре эмоциональной лексики // Труды XIV Всероссийской научной конференции "Электронные библиотеки: перспективные методы и технологии, электронные коллекции" (RCDL—2012), Переславль-Залесский: изд-во "Университет города Переславль", 2012. С. 118—123.

28. **Chetviorkin I., Braslavskiy P., Loukachevitch N.** Sentiment Analysis Track at ROMIP 2011 // Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". 2012. Vol. 2, No. 11 (18). P. 1—14.

29. **Chetviorkin I., Loukachevitch N.** Sentiment Analysis Track at ROMIP 2012 // Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2013", Bekasovo. 2013. Vol. 2, No. 12 (19). P. 40—50.

30. **Kotelnikov E. V., Razova E. V., Fishcheva I. N.** A close look at Russian morphological parsers: which one is the best? // In: A. Filchenkov, L. Pivovarova, J. Žižka (eds.) Artificial Intelligence and Natural Language. AINL 2017. Communications in Computer and Information Science, Vol. 789. Springer, 2018. P. 131—142.

31. **Kotelnikov E. V., Bushmeleva N. A., Razova E. V., Peskisheva T. A., Pletneva M. V.** Manually Created Sentiment Lexicons: Research and Development // Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue". 01—04 June 2016. No. 15 (22). P. 281—295.

32. **Lan M., Tan C. L., Su J., Lu Y.** Supervised and Traditional Term Weighting Methods for Automatic Text Categorization // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2009. Vol. 31, No. 4. P. 721—735.

33. **Dhillon I. S.** Co-clustering Documents and Words using Bipartite Spectral Graph Partitioning // Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001), 2001. P. 269—274.

34. **LIBSVM** — A Library for Support Vector Machines. URL: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

35. **Унифицированный** язык моделирования UML. URL: http://www.uml.org.

36. **Pedregosa F., Varoquaux G., Gramfort A.** et al. Scikit-learn: Machine Learning in Python // Journal of Machine Learning Research. 2011. Vol. 12. P. 2825—2830.

37. **Manning C. D., Raghavan P., Schütze H.** Introduction to information. Cambridge University Press, NY, USA, 2008. 496 p.

Software System for Text Sentiment Analysis Based on Sentiment Lexicons

M. V. Pletneva, pletneva.mv.kirov@gmail.com, Vyatka State University, Kirov, 610000, Russian Federation

Corresponding author:

Pletneva Maria V., Assistant, Vyatka State University, Kirov, 610000, Russian Federation, E-mail: pletneva.mv.kirov@gmail.com

Received on July 03, 2018
Accepted on July 30, 2018

The article is devoted to the problem of automatic text sentiment analysis. Under the text sentiment the author's emotional attitude, expressed in the text is meant. There is a wide range of areas where sentiment analysis is applied, including political, sociological and marketing research, the search engines, the human-computer interfaces.

The task of sentiment analysis is complicated by the strong dependence of the sentiment expressions on the context and domain, the presence of sarcasm and irony. To solve this problem the following approaches are used: a knowledge-based approach; machine learning and combined approach. Each of these approaches has advantages and disadvantages. In this article a combined approach is used.

The article proposes the structure of the software system for text sentiment analysis based on sentiment lexicons. The structure includes four main subsystems, namely text preprocessing, sentiment lexicon generation, sentiment lexicon parameters determination and classification subsystems. The implementation of this system is considered in accordance with the object-oriented approach. Main classes and their relations are described with UML class diagram.

The results of developed system's experimental research are presented. The research is performed using the text corpora of movies and books reviews, provided by organizers of ROMIP-2011 and ROMIP-2012 seminars. These results show the superiority of the proposed system over widespread machine learning methods.

The developed system can be used as a standalone application for summarizing opinions on the text corpora. It is possible to integrate the main part of the system into search services, focused on the preparation of recommendations based on the text sentiment analysis.

***Keywords: natural** language processing, opinion mining, text sentiment analysis, sentiment lexicons, software system, class diagram.*

Acknowledgments: *This work was supported by the Ministry of Education and Science of the Russian Federation (the project No. 34.2092.2017/4.6 "Research and Development of Sentiment Lexicons for Text Sentiment Analysis").*

For citation:

Pletneva M. V. Software System for Text Sentiment Analysis Based on Sentiment Lexicons*, Programmnaya Ingeneria,* 2019, vol. 10, no. 1, pp. 38—46

DOI: 10.17587/prin.10.38-46

References

1. **Taboada M.** Sentiment Analysis: An Overview from Linguistics, *Annual Review of Linguistics*, 2016, no. 2, pp. 325—347.

2. **Godbole N. Srinivasaiah M., Skiena S.** Large-Scale Sentiment Analysis for News and Blogs, *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, 2007, available at: https://icwsm.org/papers/3-Godbole-Srinivasaiah-Skiena.pdf.

3. **Agarwal A., Xie B., Vovsha I., Rambow O., Passonneau R.** Sentiment Analysis of Twitter Data, *Proceedings of the Workshop on Language in Social Media (LSM 2011)*, 2011, pp. 30—38.

4. **Gamon M.** Sentiment classification on customer feedback data: Noisy data, large feature vectors, and the role of linguistic analysis, *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2004, pp. 841—847.

5. **Sebastiani F.** Machine learning in Automated Text Categorization, *ACM Computing Surveys*, 2002, vol. 34, pp. 1—47.

6. **Liu B.** *Sentiment Analysis and Opinion Mining*, Morgan & Claypool Publishers, 2012, 168 p.

7. **Tomita** parser, available at: https://tech.yandex.ru/tomita/

8. **Vasilyev V. G., Denisenko A. A., Solovyev D. A.** Aspect Extraction and Twitter Sentiment Classification by Fragment Rules, *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, 2015, No. 14 (21), pp. 76—87.

9. **Pazel'skaia A. G., Solov'ev A. N.** Metod opredelenija emocij v tekstah na russkom jazyke (A method of sentiment analysis in Russian texts), *Komp'juternaja lingvistika i intellektual'nye tehnologii: po materialam ezhegodnoj mezhdunarodnoj konferencii "Dialog"*, 2011, no. 10 (17), pp. 510—522 (in Russian).

10. **König A. C., Brill E.** Reducing the human overhead in text categorization, *Proceedings of the 12th ACM SIGKDD conference on knowledge discovery and data mining*, 2006, pp. 598—603.

11. **Kuznetsova E. S., Loukachevitch N. V., Chetviorkin I. I.** Testing rules for a sentiment analysis system, *Computational Linguis-*

tics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue", 2013, no. 12 (19), pp. 71—80.

12. **Panicheva P. V.** ATEX: a rule-based sentiment analysis system processing texts in various topics, *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, 2013, no. 12 (19), pp. 101—112.

13. **Pang B., Lee L., Vaithyanathan S.** Thumbs up? Sentiment Classification using Machine Learning Techniques, *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002, pp. 79—86.

14. **Boiy E., Hens P., Deschacht K., Moens M.-F.** Automatic Sentiment Analysis in On-line Text, *Proceedings of Conference on Electronic Publishing (ELPUB'2007)*, 2007, pp. 349—360.

15. **Go A., Bhayani R., Huang L.** Twitter Sentiment Classification using Distant Supervision, *Association for Computational Linguistics*, 2009, pp. 30—38.

16. **Bhowmick P. K., Basu A., Mitra P.** Classifying Emotion in News Sentences: When Machine Classification Meets Human Classification, *International Journal on Computer Science and Engineering*, 2010, vol. 2 (1), pp. 98—108.

17. **Kouloumpis E., Wilson T., Moore J.** Twitter Sentiment Analysis: The Good the Bad and the OMG!, *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media (ICWSM-2011)*, 2011, pp. 538—541.

18. **Tutubalina E. V., Zagulova M. A., Ivanov V. V., Malykh V. A.** A Supervised Approach for SentiRuEval Task on Sentiment Analysis of Tweets about Telecom and Financial Companies, *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, 2015, no. 14 (21), pp. 65—75.

19. **Turney P.** Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews, *Proceedings of the Association for Computational Linguistics (ACL)*, 2002, pp. 417—424.

20. **Chaovalit P., Zhou L.** Movie review mining: A comparison between supervised and unsupervised classification approaches, *Proceedings of the Hawaii International Conference on System Sciences (HICSS)*, 2005, pp. 112c—112c.

21. **Kozareva Z., Navarro B., Vazquez S., Montoyo A.** Ua-zbsa: A headline emotion classification through web information, *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval—2007)*, Prague, Czech Republic, Association for Computational Linguistics, 2007, pp. 334—337.

22. **Kotelnikov E. V., Klekovkina M. V.** Avtomaticheskij analiz tonal'nosti tekstov na osnove metodov mashinnogo obuchenija (The automatic sentiment text classification based on machine learning methods), *Komp'juternaja lingvistika i intellektual'nye tehnologii: po materialam ezhegodnoj mezhdunarodnoj konferencii "Dialog"*, 2012, no. 11 (18), pp. 753—762 (in Russian).

23. **Taboada M., Brooke J., Tofiloski M., Voll K., Stede M.** Lexicon-Based Methods for Sentiment Analysis, *Computational Linguistics*, 2011, vol. 37 (2), pp. 267—307.

24. **Kotelnikov E. V.** Kombinirovannyj metod avtomaticheskogo opredelenija tonal'nosti teksta (Combined method of text tonality automatic identifying), *Programmnye produkty i sistemy*, 2012, no. 3, pp. 189—195 (in Russian).

25. **Prabowo R., Thelwall M.** Sentiment analysis: A combined approach, *Journal of Informetrics*, 2009, vol. 3, no. 2, pp. 143—157.

26. **Kotelnikov E. V., Pletneva M. V.** Analiz tonal'nosti tekstov na osnove geneticheskogo algoritma i sovmestnoj klasterizacii slov i dokumentov (Text sentiment classification based on a genetic algorithm and word and document co-clustering), *Izvestija RAN. Teorija i sistemy upravlenija*, 2016, no. 1, pp. 115—123 (in Russian).

27. **Klekovkina M. V., Kotelnikov E. V.** Metod avtomaticheskoj klassifikacii tekstov po tonal'nosti, osnovannyj na slovare emocional'noj leksiki (The automatic sentiment text classification method based on emotional vocabulary), *Trudy XIV Vserossijskoj nauchnoj konferencii "Elektronnye biblioteki: perspektivnye metody i tehnologii, elektronnye kollekcii" (RCDL—2012)*, Pereslavl'-Zalesskij, Izd. "Universitet goroda Pereslavl'", 2012, pp. 118—123 (in Russian).

28. **Chetviorkin I., Braslavskiy P., Loukachevitch N.** Sentiment Analysis Track at ROMIP 2011, *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, 2012, vol. 2, no. 11 (18), pp. 1—14.

29. **Chetviorkin I., Loukachevitch N.** Sentiment Analysis Track at ROMIP 2012, *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2013"*, Bekasovo, 2013, vol. 2, no. 12 (19), pp. 40—50.

30. **Kotelnikov E. V., Razova E. V., Fishcheva I. N.** A close look at Russian morphological parsers: which one is the best? // A. Filchenkov, L. Pivovarova, J. Žižka (eds.), *Artificial Intelligence and Natural Language. AINL 2017. Communications in Computer and Information Science*, vol. 789. Springer, 2018, pp. 131—142.

31. **Kotelnikov E. V., Bushmeleva N. A., Razova E. V., Peskisheva T. A., Pletneva M. V.** Manually Created Sentiment Lexicons: Research and Development, *Computational Linguistics and Intellectual Technologies: Papers from the Annual International Conference "Dialogue"*, 01—04 June 2016, no. 15 (22), pp. 281—295.

32. **Lan M., Tan C. L., Su J., Lu Y.** Supervised and Traditional Term Weighting Methods for Automatic Text Categorization, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009, vol. 31, no. 4, pp. 721—735.

33. **Dhillon I. S.** Co-clustering Documents and Words using Bipartite Spectral Graph Partitioning, *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001)*, 2001, pp. 269—274.

34. **LIBSVM** — A Library for Support Vector Machines, available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

35. **Unified** Modeling Language, UML, available at: http://www.uml.org.

36. **Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos A., Cournapeau D., Brucher M., Perrot M., Duchesnay E.** Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, 2011, vol. 12, pp. 2825—2830.

37. **Manning C. D., Raghavan P., Schütze H.** *Introduction to information retrieval*, Cambridge University Press, NY, USA, 2008, 496 p.

Указатель статей, опубликованных в журнале "Программная инженерия" в 2018 г.

Index of articles published in the journal "Software Engineering " in 2018

ООО "Издательство "Новые технологии". 107076, Москва, Стромынский пер., 4
Технический редактор *Е. М. Патрушева*. Корректор *Н. В. Яшина*

Сдано в набор 07.11.2018 г. Подписано в печать 20.12.2018 г. Формат 60×88 1/8. Заказ PI119
Цена свободная.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1. Сайт: **www.aov.ru**